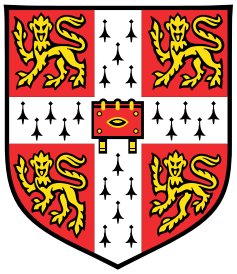
Analysing and Mitigating Classification Bias for Text-based Foundation Models



Adian Liusie

Department of Engineering
University of Cambridge

This dissertation is submitted for the degree of
Doctor of Philosophy

Pembroke College

October 2024

Declaration

This thesis is the result of my own work and includes nothing which is the outcome of work done in collaboration except as declared in the preface and specified in the text. It is not substantially the same as any work that has already been submitted, or is being concurrently submitted, for any degree, diploma or other qualification at the University of Cambridge or any other University or similar institution except as declared in the preface and specified in the text. It does not exceed the prescribed word limit for the relevant Degree Committee.

Adian Liusie
October 2024

Abstract

Examining and Mitigating Classification Bias for Text-based Foundation Models
Adian Liusie

The objective of text classification is to categorise texts into one of several pre-defined classes. Text classification is a standard natural language processing (NLP) task with various applicability in many domains, such as analysing the evolving sentiment of users on a platform, identifying and filtering fraudulent reviews, or extracting useful features in a pipeline. While text classification has traditionally been performed manually, the rapid advancement of deep learning approaches has sparked considerable interest in developing automatic text classifiers. This has been accelerated by the introduction of pre-trained large language models (LLMs), models trained on vast quantities of digital textual data, enabling unprecedented capabilities across a diverse range of natural language understanding and generation tasks. These NLP foundation models are typically leveraged within two main methodologies: the "pre-train and fine-tune" paradigm or by prompting instruction-following LLMs. While these approaches have been widely applied and have demonstrated state-of-the-art performance across NLP benchmarks, there remain concerns about their reliability, particularly regarding their susceptibility to spurious correlations and implicit model bias.

This thesis analyses the forms of bias and spurious correlations that are present when NLP foundation models are applied to text classification tasks. We analyse particular biases present in the systems, determine what impact these biases have on the predictions, and examine whether mitigation techniques can reduce the influence of the biases. The first part of the thesis focuses on the pre-train and fine-tune methodology, where we analyse the risk of systems learning spurious correlation by examining two popular NLP tasks: sentiment classification and multiple choice reading comprehension (MCRC). For sentiment classification, we demonstrate that fine-tuned systems may leverage spurious stopword relationships based on the stopword distribution in the training data. For MCRC, we find that models are susceptible to ignoring the contextual information and may use world knowledge to solve the task, which we leverage to assess question quality.

In the second part of the thesis, we examine biases present when instruction-following LLMs are prompted zero-shot for text classification. We analyse the prompt-based classifier setup and examine the biases present in these systems when applied to text classification tasks, including text classification and multiple choice question answering (MCQA) tasks. For text classification, we demonstrate that the choice of label words can cause an implicit prior that may favour particular classes over others, severely impacting the performance of these systems. However, by considering reweighting debiasing schemes, we demonstrate that both zero-resource and our proposed unsupervised reweighting debiasing yield more robust performance and reduce the sensitivity to the choice of label words. Further, we illustrate that when instruction-following LLMs are prompted for MCQA, they can exhibit considerable permutation bias. Here, the systems are sensitive to the ordering of the input options, which also negatively impacts task performance. We show that permutation debiasing improves performance significantly, and we propose a simple distillation framework to make this inefficient process more efficient.

The thesis concludes by considering biases for new tasks and domains. We propose LLM comparative assessment, a novel way of performing general, zero-shot, and effective NLG assessment by prompting LLMs to make pairwise decisions. For LLM comparative assessment, we find that position bias is also present and demonstrate that averaging the probabilities over both permutations can result in more accurate decisions and final rankings. We extend this approach to the product-of-experts framework for LLM comparative assessment, enabling faster convergence with fewer comparisons. Further, we investigate accounting for bias in the experts, which can result in better performance when a low number of comparisons are used. Finally, we conclude the thesis by examining whether our debiasing approaches generalise into other modalities, particularly the audio domain. We propose a novel way to leverage the emergent abilities of ASR foundation models for zero-shot audio classification and demonstrate that our proposed reweighting debiasing approaches remain effective for tasks in the audio modality as well.

Acknowledgements

First, I would like to thank my PhD supervisor, Professor Mark Gales, for his crucial support, insight and invaluable guidance over the last 4 years. I also want to thank Vyas Raina, Vatsal Raina, Potsawee Manakul, Yassir Fathullah, Rao Ma and Kate Knill, as well as the entire ALTA group, for the collaborations, enjoyable discussions and for making the last few years very memorable. I also want to express my gratitude to Cambridge University Press & Assessment for sponsoring my PhD. Finally, I would like to thank my family and friends for their unconditional support over all these years.

I would like to dedicate this thesis to my teacher …

Table of contents

List of figures

List of tables

Notation

Acronyms and Abbreviations

ASR Automatic Speech Recognition

CNN Convolutional Neural Network

ENO Effective Number of Options

KL Kullback–Leibler (divergence)

LLM Large Language Model

MLM Masked Language Model

MCQA Multiple Choice Question Answering

MCRC Multiple Choice Reading Comprehension

NLG Natural Language Generation

NLI Natural Language Inference

NLL Negative Log Likelihood

NLP Natural Language Processing

PCC Pearson Correlation Coefficient

PoE Product of Experts

RNN Recurrent Neural Network

SCC Spearman Correlation Coefficient

SSL Self-Supervised Learning

Roman Symbols

a	Scalar
$\mathbf{a}$	Vector
$\mathbf{A}$	Matrix
$\mathcal{C}$	Set of pairwise comparisons
$\mathcal{D}$	Dataset
$\mathcal{L}$	Loss function
M	Dataset size
$\mathcal{P}$	Prompt
$\boldsymbol{x}$	Text input
$x_{1:N}$	Text input (represented as sequence of N tokens)

Greek Symbols

$\boldsymbol{\theta}$	Model parameters
$\phi(\cdot)$	Activation function
ω_k	Class k

Probability and Statistics

$\mathtt{p}(\cdot)$	Probability density function
$\mathtt{P}(\cdot)$	Probability mass function
$\mathtt{P}(\cdot;\boldsymbol{\theta})$	Probability mass function parametrised by $\boldsymbol{\theta}$
$\mathcal{N}(\cdot)$	Gaussian distribution
$\mathcal{H}(\cdot)$	Shannon entropy of probability mass function
$\mathbb{E}$	Expectation

Functions and Operators

$\mathbb{1}(\cdot)$	Indicator function

$[\cdot\,;\,\cdot]$ Vector concatenation

$\oplus$ Text concatenation

Superscripts

(i) i-th example of the dataset

T Transpose

$^{-1}$ Matrix inverse

Subscripts

i General index

$i : j$ Indices from i to j

Chapter 1

Introduction

The recent advancements in text-based foundation models [24] have revolutionised the field of natural language processing (NLP) [51, 30, 241]. These NLP foundation models possess robust task representations, which has led to their widespread adoption and impressive state-of-the-art performance across a diverse range of language-based tasks. Text-based foundation models typically operate under two popular paradigms; the first is the "pre-train and fine-tune" methodology [51, 185, 45], where the foundation model serves as a robust starting point that can then be adapted to target tasks by further training on bespoke labelled task data. The second is "prompting" [30, 314], where the foundation models are designed to understand natural language instructions, and the system is then prompted to perform particular NLP tasks. These two approaches have been widely adopted to apply foundation models across diverse NLP tasks [51, 185, 314, 2]. One popular use case is automatic text classification, which enables convenient, cost effective, and scalable classification decisions. Foundation models have demonstrated impressive performance for NLP tasks, leading to increased deployment of automatic text-classifiers in critical real-world applications [317]. As these models become more prevalent, it's important for developers to be aware of both the capabilities and limitations of systems and to examine whether these models exhibit biases that can impact the reliability and accuracy of the predictions.

Although both methodologies have enabled excellent performance, they are not without limitations. For the pre-train and fine-tune methodology, by training the system on limited supervised data, the system may learn ineffective features that inadvertently pick up on spurious correlations. Learning superficial shortcuts [85] may severely impact the generalisability of the systems, as though learned spurious correlations may hold within the training domain, the spurious relationships will limit performance in other domains of the task [210]. For example, within a given product review platform, negative reviews may typically be longer

than positive reviews due to the tendency of angry reviewers to vent. Although this learned relationship may provide useful information when classifying reviews on the platform, the rule will fail when applied to a different domain, such as when classifying the sentiment of headlines. On the other hand, prompting is less susceptible to domain-specific spurious correlations. However, prompted systems may manifest biases such as positional bias or label word bias [355, 247] and suffer from systematic biases learned from training, including those that resemble human cognitive biases [1, 288, 178].

This thesis aims to explore the forms of classification bias that can emerge when utilizing NLP foundation models and methodologies. This thesis introduces no new architectures, foundation models, or tasks; instead, it analyses existing NLP models and approaches applied to standard tasks, examining the nature of resulting biases. The contributions of the thesis are in identifying previously unknown spurious correlations and biases and, further, in proposing potential mitigation techniques to minimise their impact and enable better performance and reliability. We also investigate the generalisability of our analysis and examine whether similar observations hold for more recently proposed approaches as well as other modalities. We extend our study to zero-shot NLG assessment and propose "LLM comparative assessment", a novel approach to better leverage the abilities of instruction-following LLMs for text quality assessment, where an LLM is used to judge which of two texts is better. We investigate the influence of bias in this approach, particularly positional bias, and consider how to make the approach more efficient. We also provide an initial investigation into other modalities and consider whether prompted ASR foundation models demonstrate similar biases and whether they can be mitigated with the same debiasing schemes as in NLP.

1.1 Thesis Outline

Chapters 2-4 provide background content describing the pre-requisite knowledge of deep learning, foundation models and the main text tasks studied in the thesis, namely text classification and text quality assessment. Chapters 5-8 then provide experimental analysis and contain the significant contributions of the thesis, considering both the pre-train and fine-tune methodology (Chapter 5) as well as the zero-shot prompting methodology (Chapters 6-8). The thesis has the following structure:

- **Chapter 2** provides the pre-requisite deep learning information and details some standard networks, starting with Feed-Forward Networks and building up to transformer architectures, which is the fundamental architecture used throughout the thesis. The

chapter also contains a discussion on the practical optimisation of the networks and inference methods.

- **Chapter 3** introduces NLP foundation models, discussing typical pre-training self-supervised tasks, the details of various popular pre-trained Large Language Models (LLMs), and concludes with details of model alignment, including instruction-tuning and RLHF.

- **Chapter 4** provides background for the main text tasks analysed throughout the thesis: text classification and text quality assessment. We detail various methodologies of applying pre-trained LLMs to classification, provide details of baseline NLG assessment approaches, and also provide background on comparative judgement, which supports the theory of Chapter 7.

- **Chapter 5** is the first of the experimental sections. In this chapter, we examine spurious correlation in NLP and discuss how fine-tuned LLMs may be at risk of learning spurious features. We highlight two novel types of spurious correlations, one based on stopword distribution and the other on using world knowledge as a shortcut for comprehension tasks. We demonstrate the presence of spurious correlations and their impact on real systems, and use shortcut-based approaches to propose metrics for assessing question quality.

- **Chapter 6** discusses biases present in zero-shot LLMs and introduces several mitigation schemes. We first consider label-word bias, where the selection of the label words can yield implicit priors that result in class biases, and formalise the reweighting debiasing scheme and demonstrate its efficacy. We further consider permutation bias for Multiple Choice Question Answering (MCQA) tasks and demonstrate the effectiveness of permutation debiasing while proposing a framework to train efficient debiased students to make the process more efficient.

- **Chapter 7** introduces LLM comparative assessment, a text quality assessment approach where LLMs make pairwise judgements on which of two texts is better. We further propose the Product of Experts framework for comparative assessment, which alleviates computational expense concerns and enables comparative assessment to operate effectively when only using a subset of all the comparisons. LLM comparative assessment is demonstrated to be an effective zero-shot approach that outperforms many existing baselines, and though it does suffer from permutation bias, this can be accounted for by using both permutations or accounting for bias in the experts.

- **Chapter 8** expands the debiasing approaches to the speech modality and introduces a novel way to prompt ASR foundation models for zero-shot audio classification. We observe that, similar to NLP models, these prompted ASR models exhibit label word bias, and we show that applying reweighting debiasing can mitigate such bias and lead to performance improvements.

- **Chapter 9** presents the conclusions of the thesis, summarises the key contributions and discusses future works as well as limitations of the analysis.

1.2 Published Work

Sections of my original PhD research have been published in peer-reviewed conferences over the course of this doctoral program. The papers discussed in this thesis are listed below, as well as my contributions for each piece of work. Unless explicitly stated otherwise, it can be assumed that I was the core contributor of the project, where as the core contributor, I set up the code base, ran the main experiments, collected results and led paper writing.

- [192] **Adian Liusie**, Vyas Raina, Vatsal Raina, Mark Gales, "Analyzing biases to spurious correlations in text classification tasks", In Proceedings of the 2nd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 12th International Joint Conference on Natural Language Processing (**IJCNLP-AACL 2022**).
  → This work was done in Collaboration with Vyas Raina and Vatsal Raina. Vyas and Vatsal helped brainstorm components of the project and executed ablation analysis.
  → This paper is used as the basis for the spurious stopword correlation analysis in Chapter 5.

- [191] **Adian Liusie***, Vatsal Raina*, Mark Gales, "World knowledge in multiple choice reading comprehension", In Proceedings of the Sixth Fact Extraction and VERification Workshop **(FEVER 2023)**
  → This work was done with equal contribution by Vatsal Raina, who set up the baseline question-answering framework and was involved in ideation and paper writing. I adapted his system to run experiments on the 'shortcut inputs', performed the analysis and led the paper-writing.
  → This paper forms the basis for the spurious questions section in Chapter 5.

- [187] **Adian Liusie**, Potsawee Manakul, Mark Gales, "Mitigating word bias in zero-shot prompt-based classifiers", In Findings of the Association for Computational

Linguistics: (**IJCNLP-AACL 2023 Findings**)
→ This work was done in Collaboration with Potsawee Manakul, who provided feedback throughout the project and helped review the final paper.
→ This paper forms the basis of the weight reweighting approach of Chapter 6.

- [188] **Adian Liusie**, Potsawee Manakul, Mark Gales, "LLM comparative assessment: Zero-shot nlg evaluation through pairwise comparisons using large language models", In Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics **(EACL 2024)**
  → This work was done in Collaboration with Potsawee Manakul, who helped with discussing the initial concept and running the baseline experiments as well as experiments for podcast assessment (omitted from the thesis).
  → This paper proposed standard LLM comparative Assessment, which is used in Chapter 7.

- [199] Rao Ma*, **Adian Liusie***, Kate Knill, Mark Gales, "Investigating the emergent audio classification ability of ASR foundation models", In Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics **(NAACL 2024)**
  → This work was done with equal contribution by Rao Ma, who initially proposed the project and was responsible for all speech aspects of the project, including running the ASR decoding. I was responsible for all aspects related to debiasing, performing the experimental analysis and led the paper writing.
  → This paper forms the basis of all audio experiments of Chapter 8.

- [186] **Adian Liusie**, Yassir Fathullah, M.J.F. Gales, "Teacher-Student Training for Debiasing: General Permutation Debiasing for Large Language Models", In Findings of the Association for Computational Linguistics: (**ACL 2024 Findings**)
  → This work was done in collaboration with Yassir Fathullah, who implemented the error correction module and assisted in ideation and paper writing.
  → This paper introduced the student-teacher distillation framework, which is discussed in both Chapter 6 and Chapter 7.

- [190] **Adian Liusie**, Vatsal Raina, Yassir Fathullah, Mark Gales, "Efficient LLM Comparative Assessment: a Product of Experts Framework for Pairwise Comparisons", Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: (**EMNLP 2024**)
  → This work was done in collaboration with Yassir Fathullah and Vatsal Raina, who

both assisted in discussing the project. Yassir ran experiments using Laplacian experts (omitted from the thesis) and optimised the code for the soft-Zermelo, while Vatsal ran initial experiments on the CMCQRD dataset.

- This paper introduced the PoE framework of comparative assessment and is the basis of the second half of Chapter 7.

Chapter 2

Deep Learning Fundamentals

Deep learning [164], a branch of machine learning inspired by the structure of biological neurons [125], has revolutionised approaches to complex tasks. This transformative paradigm is centred around deep neural networks [211], systems where layers of interconnected neurons process the signal, which it transforms into structured representations [119]. By stacking multiple layers, these networks yield systems capable of generating powerful representations directly from raw data. These systems exhibit impressive capabilities in a wide range of complex tasks, such as natural language processing (NLP) [47, 233], computer vision [43, 323] and speech processing [220, 226].

This chapter provides the necessary background for deep learning to support the rest of the thesis. It provides the technical details of key architectural components, learning algorithms and inference methods that underpin their capabilities. Although many deep learning architectures have been proposed, including vanilla Feed-Forward Neural Networks (FNNs) [281] , Recurrent Neural Networks (RNNs) [63], bidirectional Long-Short Term Memories (BiLSTMs) [122] and Convolutional Neural Networks (CNNs) [165], over the last few years, one architecture has become dominant in many areas: the transformer architecture [320]. Section 2.1 first discusses the evolution of deep learning up to the transformer block, while Section 2.2 builds on this and discusses the various standard transformer architectures. Section 2.3 then discusses standard optimisation techniques to learn parameters for tasks, while Section 2.4 discusses how the models are leveraged in inference to generate responses.

2.1 Deep Neural Networks

In deep learning, computational networks learn robust representations of data through multiple layers of non-linear transformations. As these networks scale up, they can capture complex and abstract patterns in the data, enabling impressive capabilities while avoiding

manual feature engineering. These networks, often referred to as neural networks, are parametric models f that map an input $\mathbf{u} \in \mathbb{R}^d$ to an output $\mathbf{v} \in \mathbb{R}^n$,

$$\mathbf{v} = f(\mathbf{u}; \boldsymbol{\theta}) \tag{2.1}$$

where $\boldsymbol{\theta}$ is used to represent the parameters of the model. Although Equation 2.1 represents the inputs and outputs as vectors, depending on the task, the inputs/outputs can take different forms, such as a scalar, a vector or a sequence of vectors. The remainder of this section will provide the details of the basic neural network blocks.

2.1.1 Feed-Forward Neural Networks

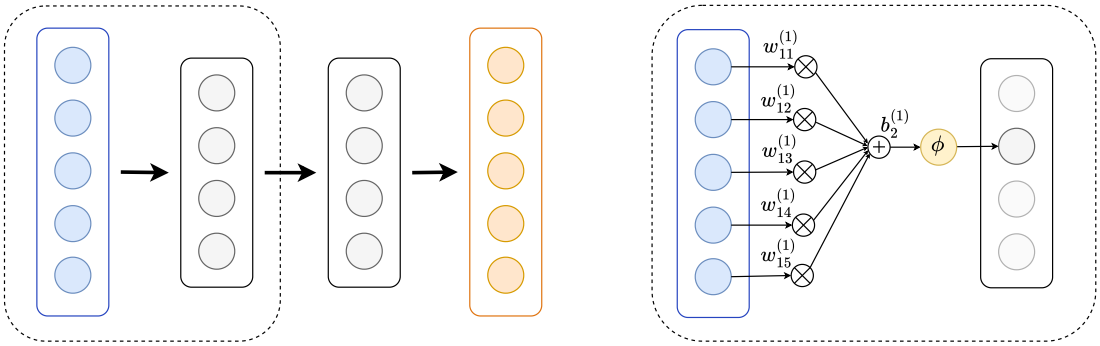


Fig. 2.1 Depiction of a 3-layer Feed-Forward Network.

One of the fundamental deep learning architectures is the feed-forward neural network (FNN) [281]. These networks comprise of multiple layers of perceptrons [278], such that in every layer, each input node is connected to each output node. A series of mathematical calculations enable the unidirectional flow of information from input to output, such that for layer l, the output representation $\mathbf{h}^{(l)} \in \mathbb{R}^n$ is a transformation of the previous layer representation $\mathbf{h}^{(l-1)} \in \mathbb{R}^d$ following,

$$\mathbf{h}^{(l)} = \phi(\mathbf{W}^{(l)}\mathbf{h}^{(l-1)} + \mathbf{b}^{(l)}) \tag{2.2}$$

where $\mathbf{W}^{(l)} \in \mathbb{R}^{n \times d}$ is the weight matrix for layer l, $\mathbf{b}^{(l)} \in \mathbb{R}^n$ is the bias term for layer l, and ϕ is a non-linear activation function. The network f is then defined as the sequential composition of all layers, where each layer l has its own associated parameters, $(\mathbf{W}^{(l)}, \mathbf{b}^{(l)})$. The layers can have different input and output dimensions, d and n, which are hyperparameters of the system, and the learnable parameters of each layer together form the parameters of the model. Figure 2.1 illustrates a Feed-Forward Network with an input and output size of

5 and 3 intermediate layers, each of size 4. The left diagram shows the complete network, while the right diagram illustrates the computation for determining the value of a neuron in the next layer.

Activation Functions

Each layer in the FNN can be decomposed into the linear function, $(\mathbf{W}^{(l)}\mathbf{h}^{(l-1)} + \mathbf{b}^{(l)})$, followed by the non-linear activation function, ϕ. The non-linear activation function is crucial for deep learning applications; without it, the composition of multiple linear transformations will be equivalent to applying a single linear transformation. Therefore, to approximate a wider class of functions and model complex relationships (while being constrained to a finite number of nodes [167]), non-linear activations are required. The non-linear activations enable a model to exploit its depth, learn hierarchical representations, and increase its capacity to model more complex functions [196]. Standard non-linear activation functions are shown in Table 2.1, with ReLU [225] and GeLU [112] largely utilised due to their simplicity, promotion of sparsity, and ability to mitigate vanishing gradient issues [90]. Figure 2.2 plots the characteristics of the different non-linear activation functions.

Activation, $\phi(\cdot)$	**Expression**
Sigmoid	$\frac{1}{1+\exp(-u)}$
Tanh	$\frac{\exp(u)-\exp(-u)}{\exp(u)+\exp(-u)}$
ReLU	$\max(0,u)$
GeLU	$\frac{u}{2}\left(1+\text{erf}\left(\frac{u}{\sqrt{2}}\right)\right)$
Softmax	$\frac{\exp(u_i)}{\sum_j \exp(u_j)}$

Table 2.1 Activation functions and their mathematical expressions.

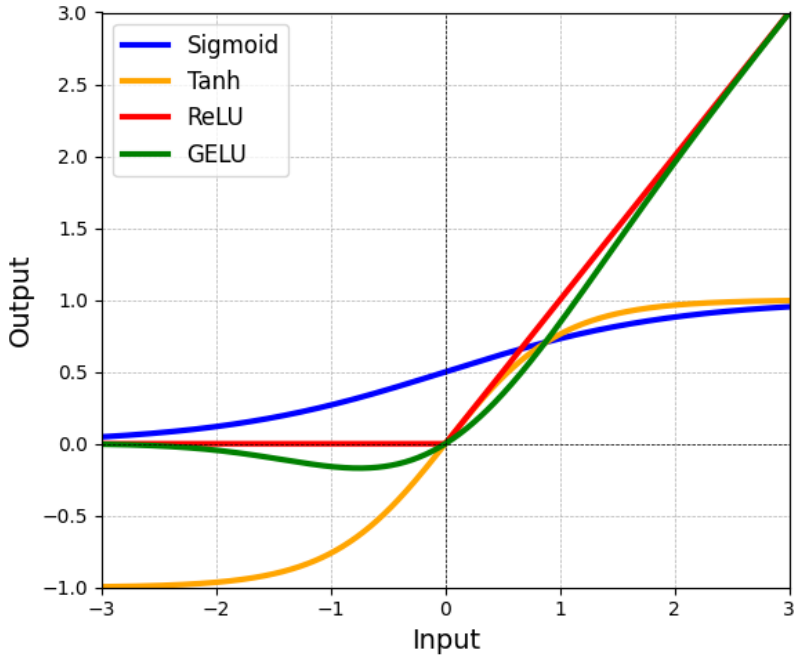


Fig. 2.2 Plot of various non-linear activation functions.

2.1.2 Convolutional Neural Networks

While FNNs are effective for many tasks, they process inputs independently, ignoring spatial relationships. Convolutional Neural Networks (CNNs) [165] address this limitation by using convolutional layers to capture translation invariances. In a CNN, the activations of the l-th hidden layer are a set of feature maps $\mathbf{H}_i^{(l)} \in \mathbb{R}^{h^{(l)} \times w^{(l)}}$, where $i \in \{1, \ldots, c^{(l)}\}$ is the channel number and $h^{(l)}$, $w^{(l)}$, and $c^{(l)}$ represent the height, width, and number of channels for layer l respectively. The CNN applies a set of convolutional filters $\mathbf{W}_{i,j}^{(l)} \in \mathbb{R}^{k_h \times k_w}$ to the feature maps, where k_h and k_w are the height and width of the filter. In each layer, the output for the next hidden layer is computed using convolution:

$$\mathbf{H}_i^{(l)} = \phi\left(\left(\sum_{j=1}^{c^{(l-1)}} \mathbf{H}_j^{(l-1)} * \mathbf{W}_{i,j}^{(l)}\right) + \mathbf{B}_i^{(l)}\right), \quad \forall i \in \{1, \ldots, c^{(l)}\} \tag{2.3}$$

where ϕ is an activation function, $*$ denotes the 2D convolution operation, and $\mathbf{B}_i^{(l)} \in \mathbb{R}^{h^{(l)} \times w^{(l)}}$ is a bias term for the i-th filter of layer l.

2.1.3 Recurrent Neural Networks

Further, FNNs are only suitable for tasks with fixed-size inputs and outputs and have limitations when applied to tasks with sequential inputs where the temporal dependencies between data points are crucial. In such scenarios, typical of NLP and speech processing tasks, recurrent neural networks (RNN) [63] may be used. RNNs handle sequential data by updating the hidden state at each timestep to reflect the information gained from all previous timesteps. Given sequential data $\mathbf{u}_{1:N}$, where $\mathbf{u}_i \in \mathbb{R}^d$ is the vector at position i, the hidden state $\mathbf{h}_i \in \mathbb{R}^n$ is updated from the previous hidden state $\mathbf{h}_{i-1}$ given the current input $\mathbf{u}_i$,

$$\mathbf{h}_i = f(\mathbf{u}_i, \mathbf{h}_{i-1}) \tag{2.4}$$

A vanilla RNN processes information by computing the hidden state and output at each time step as follows:

$$\mathbf{h}_i = \phi_h(\mathbf{W}_u \mathbf{u}_i + \mathbf{W}_h \mathbf{h}_{i-1} + \mathbf{b}_h) \tag{2.5}$$

$$\mathbf{z}_i = \phi_z(\mathbf{W}_z \mathbf{h}_i + \mathbf{b}_z) \tag{2.6}$$

where $\mathbf{W}_u, \mathbf{W}_\mathbf{h}, \mathbf{b}_h, \mathbf{W}_z$ and $\mathbf{b}_z$ are trainable parameters of the model [63], ϕ_h and ϕ_z are non-linear activation functions (§2.1), and $\mathbf{z}_i$ is the output at position i. Further, multiple recurrent units can be stacked to form deep RNNs with multiple layers, which provides extra model capacity [97].

Although vanilla RNNs allow one to process sequential data of variable length and can model time-related dependencies, RNNs do have significant drawbacks. One limitation is that they leverage only unidirectional information when computing hidden states and only consider past information, not incorporating any future context. To overcome this, bidirectional RNNs [290] can be used. The two RNN hidden states, one operating forward in time and the other backward, are combined at each timestep to produce representations that incorporate both past and future information,

$$\mathbf{h}_i^f = f(\mathbf{u}_i, \mathbf{h}_{i-1}^f) \tag{2.7}$$

$$\mathbf{h}_i^b = f(\mathbf{u}_i, \mathbf{h}_{i+1}^b) \tag{2.8}$$

$$\mathbf{h}_i = [\mathbf{h}_i^f; \mathbf{h}_i^b] \tag{2.9}$$

Which is illustrated in Figure 2.3. RNNs are typically trained using backpropagation through time [335], which unfolds the recurrent network over all time steps and propagates errors backwards through the temporal sequence. Consequently, a limitation of RNNs is the issue

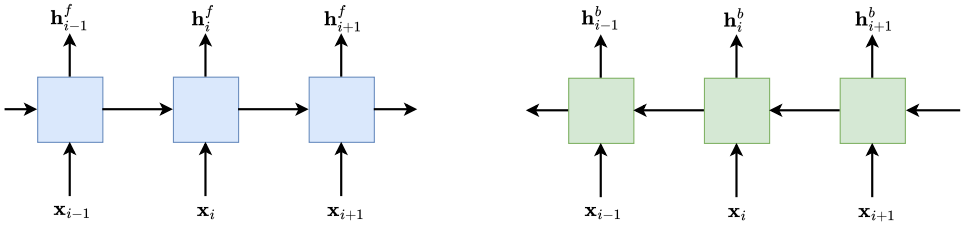


Fig. 2.3 RNN architecture unrolled in time, illustrating both forward in time (left) and backwards in time (right). Bidirectional RNNs concatenate these representations.

of the vanishing gradient [20], where very small gradients can be encountered in backpropagation through time, making it difficult for the network to learn long-range dependencies. Extensions of the RNN, such as the long short-term memory (LSTM) [122] and gated recurrent unit (GRU) [38], have been proposed to address this issue. These networks use gating mechanisms to better learn long-term dependencies, however, can also struggle to learn long-range dependencies effectively, especially for long sequences [240]. Further, the inherent sequential nature of RNNs can make training computationally inefficient.

2.1.4 Attention Mechanism

The preceding subsection (§2.1.3) introduced recurrent neural networks and detailed their drawbacks, including the computational inefficiency caused by their sequential nature and their difficulty in capturing long-range dependencies. To address the limitations, researchers drew inspiration from how humans selectively focus on different regions of an image or words in a sentence [334] and proposed the attention mechanism. The attention mechanism, as proposed by Bahdanau et al. [13], operates on a sequence of vectors $\mathbf{h}_{1:N}$ using a query vector $\mathbf{q}$. It computes a context vector $\mathbf{c}$ by weighting each input vector based on its relative importance to the query,

$$\alpha_i = \frac{\exp(f_\alpha(\mathbf{q}, \mathbf{h}_i))}{\sum_{j=1}^{N} \exp(f_\alpha(\mathbf{q}, \mathbf{h}_j))} \tag{2.10}$$

$$\mathbf{c} = \sum_{j=1}^{T} \alpha_i \mathbf{h}_i \tag{2.11}$$

where α_i is the attention weight associated with the i-th input vector. The attention function f_α computes a score based on the input query and vector, typically the dot product or scaled dot-product [320, 334], with common forms of attention functions presented in Table 2.2. This approach enables the model to dynamically focus on relevant parts of the input sequence,

thereby enhancing its ability to capture long-range dependencies and improving overall performance [340]. It is also an essential part of the transformer architecture,•which will be discussed in the next subsection.

Attention Type	**Attention Function** $f_\alpha(\mathbf{q}_i, \mathbf{h}_j)$
Dot-Product	$\mathbf{q}_i \cdot \mathbf{h}_j$
Scaled Dot-Product	$\frac{\mathbf{q}_i \cdot \mathbf{h}_j}{\sqrt{d}}$
Content-Based	$\frac{\mathbf{q}_i \cdot \mathbf{h}_j}{\|\|\mathbf{q}_i\|\|_2 \|\|\mathbf{h}_j\|\|_2}$
Additive	$\mathbf{w}_a^\top \tanh(\mathbf{W}_q \mathbf{q}_i + \mathbf{W}_k \mathbf{h}_j)$
General	$\mathbf{q}_i^\top \mathbf{W}_a \mathbf{h}_j$

Table 2.2 Common forms of attention scores. $\mathbf{W}_a, \mathbf{W}_k, \mathbf{W}_v, \mathbf{w}_a$ are parameters of the score function, and d is the dimensionality of the vectors.

2.1.5 Transformer Block

The previous section (§2.1.4) introduced the attention mechanism, which was initially proposed as an effective mechanism to aggregate the final encoded representations to better capture contextual information. Shortly after the proposal of attention, Vaswani et al. [320] demonstrated that the capabilities of attention extend significantly beyond simple representation aggregation. The paper introduced the transformer architecture, which was composed solely of attention mechanisms and feed-forward neural networks (FNNs) yet demonstrated substantial performance improvements over existing state-of-the-art methods. This innovation marked the beginning of a new era in the field of NLP, where transformer-based models began to dominate task leaderboards, a trend that continues to this day. This subsection details the components of the basic transformer block (illustrated in Figure 2.4), which forms the foundation of the transformer architecture.

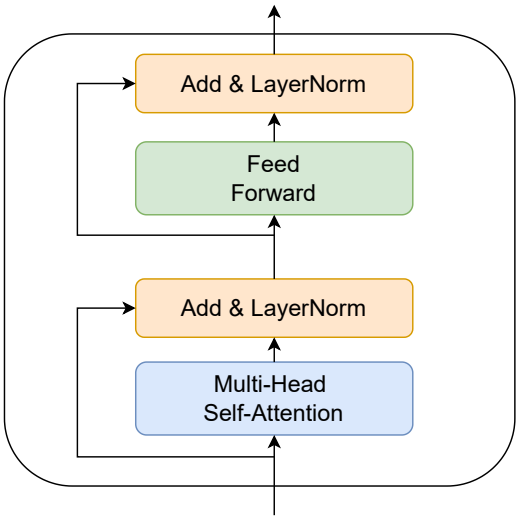


Fig. 2.4 Depiction of the transformer block.

Multi-Head Self-Attention

The first module of the transformer block, multi-head self-attention, builds upon the self-attention mechanism, a particular configuration of the attention mechanism. Given a sequence of vectors, $\mathbf{u}_{1:N}$, for each vector $\mathbf{u}_i \in \mathbb{R}^d$, self-attention first computes the corresponding query, key and value vectors for each position of the sequence:

$$\mathbf{q}_i = \mathbf{W}_q \mathbf{u}_i \tag{2.12}$$

$$\mathbf{k}_i = \mathbf{W}_k \mathbf{u}_i \tag{2.13}$$

$$\mathbf{v}_i = \mathbf{W}_v \mathbf{u}_i \tag{2.14}$$

where $\mathbf{W}_q, \mathbf{W}_k, \mathbf{W}_v \in \mathbb{R}^{d \times d}$ are learnable weight matrices and $\mathbf{q}_i, \mathbf{k}_i, \mathbf{v}_i \in \mathbb{R}^d$ are the queries, keys and values respectively. The input and output dimensions are matched to enable residual connections and subsequent layer normalisation (discussed later in the subsection), though it is possible for the input and output dimensionalities to differ, $\mathbf{W}_q, \mathbf{W}_k, \mathbf{W}_v \in \mathbb{R}^{n \times d}$. The self-attention weights α_{ij} are then computed using the scaled dot-product between pairs of queries and keys,

$$\alpha_{ij} = \text{softmax}\left(\frac{\mathbf{q}_i \cdot \mathbf{k}_j}{\sqrt{d}}\right) = \frac{\exp\left(\frac{\mathbf{q}_i \cdot \mathbf{k}_j}{\sqrt{d}}\right)}{\sum_{j'=1}^{N} \exp\left(\frac{\mathbf{q}_i \cdot \mathbf{k}_{j'}}{\sqrt{d}}\right)} \tag{2.15}$$

The state representation for position i can then be set as a weighted summation of the values, where the weighting is based on the self-attention weights,

$$\mathbf{h}_i = \sum_{j=1}^{N} \alpha_{ij} \mathbf{v}_j \tag{2.16}$$

This yields the sequence of output vector representations, $\mathbf{h}_{1:N}$, with each vector $\mathbf{h}_i \in \mathbb{R}^d$. Unlike RNNs, which are inherently sequential, the attention block allows for parallel training, as the representations at each position are computed independently. As shown in Vaswani et al. [320], the attention block can be compactly represented as,

$$[\mathbf{h}_{1:N}] = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d}}\right)\mathbf{V} \tag{2.17}$$

where $\mathbf{Q}, \mathbf{K}, \mathbf{V}$ is the matrix of all the queries, keys and values, respectively, such that $\mathbf{Q} = [\mathbf{q}_{1:N}], \mathbf{K} = [\mathbf{k}_{1:N}], \mathbf{V} = [\mathbf{v}_{1:N}]$. Self-attention can also be extended to multi-head self-attention by running several independent self-attention mechanisms in parallel and then concatenating the outputs. For each of the H heads, separate projection matrices are learned for the queries, keys, and values,

$$\mathbf{q}_i^{(m)} = \mathbf{W}_q^{(m)} \mathbf{u}_i \tag{2.18}$$

$$\mathbf{k}_i^{(m)} = \mathbf{W}_k^{(m)} \mathbf{u}_i \tag{2.19}$$

$$\mathbf{v}_i^{(m)} = \mathbf{W}_v^{(m)} \mathbf{u}_i \tag{2.20}$$

where $\mathbf{u}_i \in \mathbb{R}^d$ is the input vector at position i, $\mathbf{W}_q^{(m)}, \mathbf{W}_k^{(m)}, \mathbf{W}_v^{(m)} \in \mathbb{R}^{d_h \times d}$ are independent parameter matrices learned for the m-th head, and $d_h = d/H$ is the dimensionality of each head, $\mathbf{q}_i^{(m)}, \mathbf{k}_i^{(m)}, \mathbf{v}_i^{(m)} \in \mathbb{R}^{d_h}$. The output of each head is computed following:

$$\mathbf{h}_i^{(m)} = \sum_{j=1}^{N} \alpha_{ij}^{(m)} \mathbf{v}_j^{(m)} \qquad \text{where} \quad \alpha_{ij}^{(m)} = \text{softmax}\left(\frac{\mathbf{q}_i^{(m)} \cdot \mathbf{k}_j^{(m)}}{\sqrt{d_h}}\right) \tag{2.21}$$

Finally, the outputs from all heads are concatenated to generate the final representation:

$$\mathbf{h}_i = [\mathbf{h}_i^{(1)}; \mathbf{h}_i^{(2)}; \ldots; \mathbf{h}_i^{(M)}] \tag{2.22}$$

Feed-Forward Neural Network

The outputs of the multi-head self-attention sub-layer are subsequently passed to a fully connected feed-forward network. This consists of a two-layer FNN, applied independently to each position of the input vector,

$$\text{FFN}(\mathbf{h}_i) = \mathbf{W_2}(\phi(\mathbf{W}_1\mathbf{h}_i + \mathbf{b}_1)) + \mathbf{b}_2 \tag{2.23}$$

where in the original transformer, the non-linear activation function is the ReLU [225], while some recent transformer models alternatively use the GeLU [112].

Residual Connection + Layer Normalisation

Each sublayer also contains a skip connection and layer normalisation. First, the residual connection [109] adds the original input (i.e. the input representation to the sublayer) to the output of the sublayer. The residual connection helps to mitigate the risk of vanishing gradients and enables deeper networks by ensuring better gradient flow during backpropagation,

$$\text{ResidualConnection}(\mathbf{u}) = \mathbf{u} + f(\mathbf{u}) \tag{2.24}$$

Where f is the sublayer function (e.g. FNN or Multi-head self-attention). For residual connections, the output of the sublayer must have the same dimensionality as the input, $n = d$, as previously assumed within multi-head self-attention.

The residual connection is followed by layer normalisation; in neural networks, gradients in one layer can be highly dependent on the outputs of the previous layer. To stabilise training and reduce covariate shifts in the representations, layer normalisation [8] is applied. This independently normalises each vector $\mathbf{u}_i \in \mathbb{R}^d$ of the vector sequence $\mathbf{u}_{1:N}$ over all its dimensions, as follows:

$$\mu_i = \frac{1}{d}\sum_{j=1}^{d} u_{ij} \tag{2.25}$$

$$\sigma_i^2 = \frac{1}{d}\sum_{j=1}^{d} (u_{ij} - \mu_i)^2 \tag{2.26}$$

$$\hat{u}_{ij} = \frac{u_{ij} - \mu_i}{\sqrt{\sigma_i^2 + \varepsilon}} \tag{2.27}$$

where ε is a small constant added for numerical stability. The output of layer normalisation is then,

$$\text{LayerNorm}(\mathbf{u}_i) = \gamma\,\hat{\mathbf{u}}_i + \beta \tag{2.28}$$

where γ and β are learnable scalar parameters for the layer. This approach differs from batch normalisation [132], which normalises each dimension based on the average over all data samples in the batch, which can constrain the size of mini-batches.

2.2 Transformer Architecture

The previous section introduced various fundamental deep-learning blocks and culminated in the discussion of the transformer block (§2.1.5). The advent of the transformer [320] marked a paradigm shift in NLP; transformers [320, 51] rapidly established itself as the dominant architecture across various benchmarks, consistently demonstrating state-of-the-art performance for diverse tasks. These models have been successfully adapted for a wide range of tasks and domains and have become the default choice for NLP solutions, typically outperforming all other architectures. This section provides a detailed overview of the three transformer architectures: the encoder-only transformer [51], the encoder-decoder transformer [320], and the decoder-only transformer [264]. This section will only focus on providing the architectural details, while the next chapter (§3) will highlight the importance of pre-training and introduce the various pre-trained transformers used to conduct experiments in this thesis.

2.2.1 Encoder-Only Transformer

Figure 2.5 illustrates the architecture of the encoder-only transformer [51], a system designed to learn powerful representations of input data. The architecture has been applied effectively to tasks where text generation abilities are not required, such as text classification. The core of the architecture is a stack of multiple transformer blocks (§2.1.5), where the outputs from one layer serve as inputs to the next. This layered structure enables the model to build increasingly abstract and nuanced representations of the input data, enabling it to capture complex relationships of the input sequence. Beyond the transformer block discussed in subsection 2.1.5, this section lists other components that are also essential to the transformer architecture.

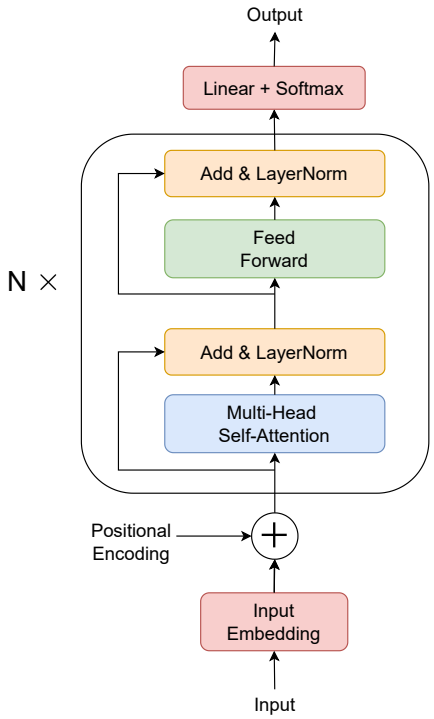


Fig. 2.5 Encoder-only transformer architecture.

Tokenization

An important practical consideration of the system is the process of converting the input data into vector representations that the transformer layers can process. The raw text input is first converted into input tokens, $x_{1:N}$, through a process known as tokenization. This process segments the text into smaller sub-units, which standardises the input for the model. The choice of sub-units may be words, subwords, or characters and depends on the specific tokenization strategy selected. Common tokenization strategies widely used in natural language processing, depicted in Figure 2.6, include:

- **Word-Level Tokenization**: Word-level tokenization [144, 218] uses whitespace as a delimiter to separate texts into words, treating each individual word as a unique token. Word-level tokenization strategies may also perform further processing, such as lowercasing, stemming and separating punctuation, which enables better grouping of words and reduces the vocabulary size [291]. If at inference, the system encounters a previously unseen word, during tokenization the word can be mapped to a special 'unknown' token [144, 22].

- **Byte-Pair Encoding**: A drawback of word-level tokenization is its inability to handle out-of-vocabulary words. Byte-pair encoding (BPE) [294] operates by starting with a

comprehensive set of tokens that cover all possible characters. Using a training corpus, it then iteratively expands the set of tokens by merging the most common pairs of tokens encountered. The vocabulary is fixed once the set of tokens reaches the desired size, and any new text sequence can be tokenized by greedily matching the text to the longest token in the vocabulary. This algorithm enables the process to tokenize any text into a set of existing tokens while maintaining a compact vocabulary size and enabling words to benefit from the knowledge of similar roots (e.g. "runner" and "running").

- **WordPiece**: WordPiece [339] is a tokenization algorithm that follows a similar principle to BPE but differs in its merging rules. Instead of merging pairs of tokens based on frequency, WordPiece merges tokens to maximise the likelihood of the training data. This is performed by selecting the pair that maximises the pair's joint frequency divided by the product of the individual token frequencies. Further, to differentiate between a subword at the start of a word and a subword that continues from a previous subword, WordPiece typically uses '##' as a special prefix to denote a continuation.

- **SentencePiece**: The previous methods make the assumption that whitespace splits texts into words, which in WordPiece and BPE is information implicitly used during detokenization. However, some languages, such as Chinese, Thai and Hindi, do not use whitespace to separate words. SentencePiece [156] addresses this limitation by using a lossless tokenization approach where all the information to reproduce the text is preserved, including whitespace. It treats the text as a raw input stream of Unicode characters and then applies a subword tokenization algorithm (either unigram language model or BPE) to create the token vocabulary. SentencePiece typically uses '_' to denote whitespace, which often indicates the start of a new word, separating word beginnings and continuations.

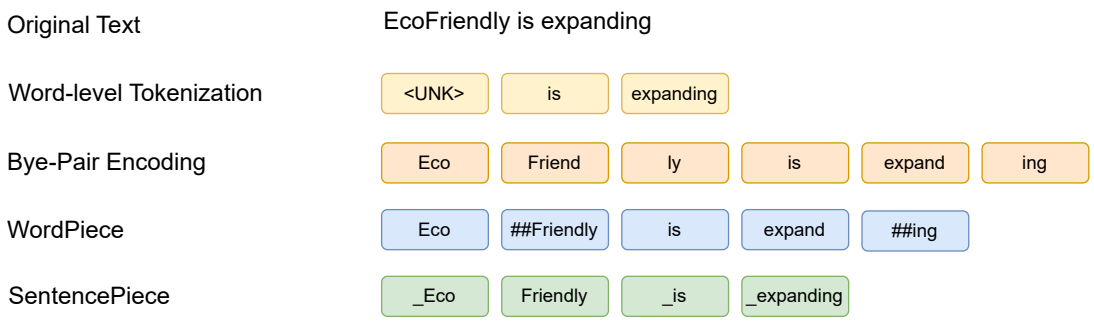


Fig. 2.6 Illustration of different tokenization schemes. Word-level tokenization splits the sentence at spaces and replaces out-of-vocabulary words with the 'unknown' token. The other approaches split the text into subwords and differ in their strategies for subword segmentation.

Embedding

The tokenized input $x_{1:N}$ is converted to the sequence $\mathbf{u}_{1:N}$ of vector embeddings through one-hot encoding followed by embedding lookup. Let x_i represent the token at position i; first, each token is represented as a one-hot vector, $\mathbf{e}_i \in \{0,1\}^{|\mathcal{V}|}$, where $\mathcal{V}$ denotes the full set of tokens. The embedding matrix $\mathbf{W}_e \in \mathbb{R}^{d\times|\mathcal{V}|}$ contains a d-dimensional vector for each token in the vocabulary. The embedding for a token is obtained by multiplying its one-hot vector with $\mathbf{W}_e$, which selects the corresponding row:

$$\mathbf{u}_i = \mathbf{W}_e\mathbf{e}_i \tag{2.29}$$

This operation is performed for each position of the input, $i \in \{1,\ldots,N\}$. Although original word representations, such as Word2vec [218] and GLoVe [245], were learned in an unsupervised fashion and then transferred to new tasks, transformers treat $\mathbf{W}_e$ as parameters of the model and learn the embedding matrix jointly with all model parameters during training.

Positional Encoding

Attention blocks compute the attention weights α_{ij} purely based on the attention score between pairs of input vectors, making all attention-based approaches position-independent. As a result, permuting the input vectors will have no influence on the output. However, positional information is important for many tasks, and for example, permuting words in a sentence can largely alter its meaning. Therefore, encoding positional information is important for language-based tasks. To incorporate positional information, the original transformer model [320] proposed **absolute positional embeddings** where positional vectors $\mathbf{p}_i \in \mathbb{R}^d$ are added to the input embedding vectors $\mathbf{u}_i \in \mathbb{R}^d$, generating positional aware input vectors $\mathbf{u}_i^p \in \mathbb{R}^d$,

$$\mathbf{u}_i^p = \mathbf{u}_i + \mathbf{p}_i \tag{2.30}$$

In the original paper, these positional encodings were fixed sinusoidal embeddings, where each dimension had a different frequency.

$$p_{ij} = \begin{cases} \sin\left(i/10000^{\frac{j}{d}}\right) & \text{if } j \mod 2 = 0 \\ \cos\left(i/10000^{\frac{j-1}{d}}\right) & \text{otherwise} \end{cases} \tag{2.31}$$

Later work proposed keeping these as learnable parameters that are learned during training [51]. As an alternative to absolute positional embeddings, one can use **relative positional**

embeddings. Here, instead of adding a fixed or learned vector to each input, the model considers the relative positions between elements and adds a bias when computing attention. This is typically done by modifying the attention mechanism to incorporate a vector $\mathbf{r}_{ij} \in \mathbb{R}^d$, such that the attention weights are computed as:

$$\alpha_{ij} = \text{softmax}\left(\frac{\mathbf{q}_i \cdot (\mathbf{k}_j + \mathbf{r}_{ij})}{\sqrt{d}}\right) \tag{2.32}$$

where $\mathbf{r}_{ij}$ represents the 'relative position' encoding between positions i and j. Positions i and j that have the same distance between them will share the same learned vectors, and these vectors are typically learned during training [266]

Linear-Layer + Softmax

The final component of the architecture is to convert the output representations back to a distribution of predicted tokens. Let $\mathbf{h}_i \in \mathbb{R}^d$ be the output of the final transformer block for the i-th token. The linear layer transforms this vector to logits as follows:

$$\mathbf{z}_i = \mathbf{W}_o \mathbf{h}_i + \mathbf{b}_o \tag{2.33}$$

where $\mathbf{W}_o \in \mathbb{R}^{|\mathcal{V}| \times d}$ is a weight matrix, $\mathbf{b}_o \in \mathbb{R}^{|\mathcal{V}|}$ is a bias vector, and $|\mathcal{V}|$ is the vocabulary size. The softmax function is then applied to convert the logits to a probability distribution over the tokens. Note that in practice, the weights of the embedding matrix $\mathbf{W}_e$ are often shared with the output layer $\mathbf{W}_o$. This technique, known as weight tying, reduces the total number of parameters and can improve performance in language modelling tasks [256].

2.2.2 Decoder-Only Transformer

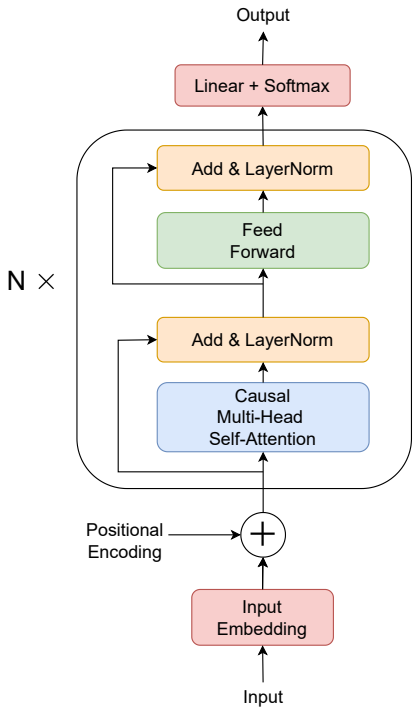


Fig. 2.7 Decoder-only transformer architecture.

The previous encoder-only transformer was proposed as an effective architecture for hierarchical representation learning, particularly for classification and regression tasks. However, a crucial limitation is that the architecture is bidirectional, meaning that the representations at each position can attend to all other positions. This bidirectionality prevents the architecture from performing auto-regressive decoding, making it unsuitable for generative tasks.

Masked Multi-Head Self-Attention

The decoder-only transformer (illustrated in Figure 2.7) addresses this limitation by introducing a causal mask to the multi-head self-attention mechanism, ensuring that each position can only attend to previous positions. This way, the output tokens can be fed back into the system and used sequentially to predict the next token. The self-attention is made causal by

explicitly setting all non-causal attention weights to $-\infty$:

$$e_{ij} = \begin{cases} \frac{\mathbf{q}_i \cdot \mathbf{k}_j}{\sqrt{d}} & \text{if } j \leq i \\ -\infty & \text{if } j > i \end{cases} \tag{2.34}$$

$$\alpha_{ij} = \text{softmax}\left(e_{ij}\right) \tag{2.35}$$

where d is the dimensionality of the vectors, $\mathbf{q}_i, \mathbf{k}_i \in \mathbb{R}^d$, which forces all 'future' attention scores to be 0.

2.2.3 Encoder-Decoder Transformer

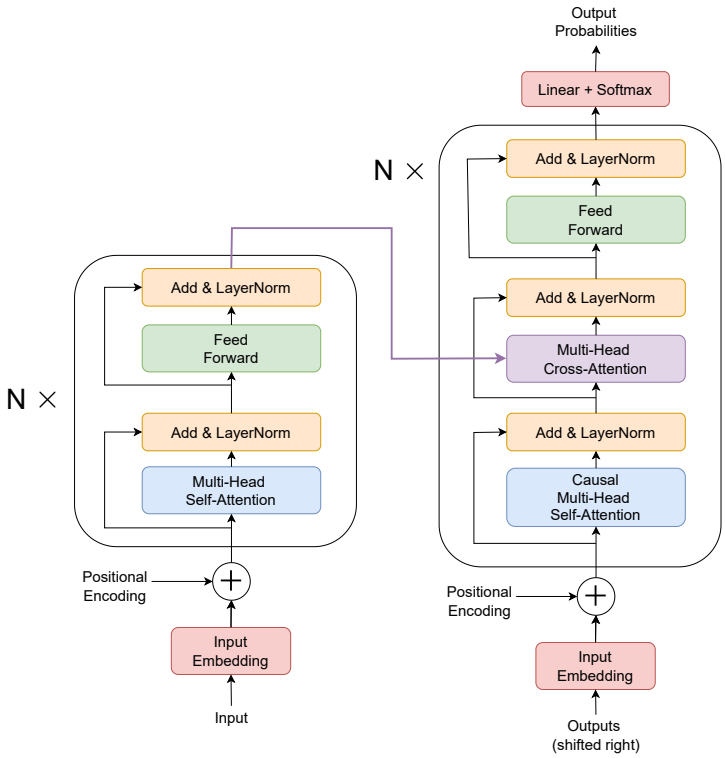


Fig. 2.8 Encoder-decoder transformer architecture.

Although the previous two sections introduced the encoder-only [51] and decoder-only transformer [264], the transformer architecture was initially proposed within the encoder-decoder framework [320]. This framework includes both an encoder stack and a decoder stack (as depicted in Figure 2.8) and leverages the abilities of each individual stack. The encoder takes the input tokens and generates contextualised sequence representations (with

the same length as the input), while the decoder autoregressively generates the output sequence, conditioned on the encoder's representations. However, the self-attention blocks in the decoder cannot directly utilise the encoder's output. To enable this, another attention mechanism is introduced for encoder-decoder transformers: multi-head cross-attention.

Multi-Head Cross-Attention

In the encoder-decoder framework, given an input $x_{1:N}$, the encoder generates a sequence of contextual representations $\mathbf{h}_{1:N}$ which are used by the decoder to generate a sequence of output representations $\mathbf{y}_{1:L}$. These representations are generated autoregressively from all previous representations, achieved using multi-head cross attention. Here, given the encoder representations $\mathbf{h}_{1:N}$ and the input decoder representations of the current layer $\mathbf{h}^{y}_{1:k}$, the representation of the next position is calculated using multi-head cross attention. Multi-head cross-attention computes the keys and values from the output of the encoder, $\mathbf{h}_{1:N}$, and computes the queries using the decoder representations in the current layer, $\mathbf{h}^{\mathrm{y}}_{1:k}$:

$$\mathbf{q}_i = \mathbf{W}_q \mathbf{h}^{\mathrm{y}}_i \tag{2.36}$$

$$\mathbf{k}_i = \mathbf{W}_k \mathbf{h}_i \tag{2.37}$$

$$\mathbf{v}_i = \mathbf{W}_v \mathbf{h}_i \tag{2.38}$$

Similar to the previous attention mechanisms, the attention weights are calculated by a scaled dot-product between the queries and keys:

$$\alpha_{ij} = \mathrm{softmax}\left(\frac{\mathbf{q}_i \cdot \mathbf{k}_j}{\sqrt{d}}\right) \tag{2.39}$$

$$\mathbf{y}_k = \sum_{j=1}^{k} \alpha_{kj} \mathbf{v}_j \tag{2.40}$$

Therefore, the contextualised vectors $\mathbf{y}_k$ are a linear combination of projected representations from the encoder, enabling the decoder to directly leverage the final transformer encoder representations. The process is causal as the decoder only attends to previous positions during self-attention, ensuring that future tokens are not attended to during generation. The use of caching helps improve efficiency by storing past decoder self-attention outputs, avoiding redundant computation. Additionally, the process described represents the operation of a single head; multi-head cross-attention can be achieved by running independent heads in parallel, each with its own set of learned projections, and concatenating their outputs to form the final representation.

2.3 Optimisation

The previous section introduced the typical networks used for deep learning in NLP and culminated in the transformer architectures that underpin modern foundation models. The section further described how neural networks are complex mapping functions characterised by the parameters of the model, which take the form of weight matrices and bias vectors. However, the previous section did not consider how to find the right parameters to enable systems to achieve the task of interest. Therefore, this section considers the optimisation techniques required to learn the trainable parameters of the network, $\boldsymbol{\theta}$, and discusses typical deep-learning training procedures.

2.3.1 Training Criterion

Machine learning algorithms are data-driven, and given training data $\mathcal{D}$, the objective is for the system to identify patterns and learn mapping functions that generalise to new, unseen data. Deep neural networks are often trained in an end-to-end fashion, where the system jointly learns features as well as predictions. Essential to machine learning methods is the definition of the training criterion, known as the loss function. This loss function provides the system with an explicit objective to train the model parameters with during training. The system, parameterised by model parameters $\boldsymbol{\theta}$, typically models an underlying probability distribution for a particular target task. The goal is to find the optimal parameters $\hat{\boldsymbol{\theta}}$ that minimise the loss $\mathcal{L}(\boldsymbol{\theta})$, where the loss encapsulates the specific task, such that the system is trained to learn:

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}) \tag{2.41}$$

where for convenience, the dependence on the dataset $\mathcal{D}$ is dropped. For autoregressive models, the parameters of the models typically estimate the language model probability associated with generating a particular token. For example, given inputs $x_{1:N}$ and the partially generated output $y_{1:j-1}$, the system models the probability of generating the next token, $y \in \mathcal{V}$,

$$\hat{y}_j \sim \mathtt{P}\left(y | y_{1:j-1}, x_{1:N}; \boldsymbol{\theta}\right) \tag{2.42}$$

The training objective is then to find the parameters $\boldsymbol{\theta}$ that best model the distribution. Assuming access to a labelled dataset, $\mathcal{D} = \{(x^{(i)}_{1:N}, y^{(i)}_{1:L})\}_{i=1}^{M}$, of paired input sequence $x_{1:N}$ and corresponding gold-standard output sequence $y_{1:L}$ (where for simplicity, the dependence of N and L on the indices is dropped), a common training approach is maximum likelihood

training. For this loss function, the objective is to search for the parameters that maximise the likelihood of the training data, often practically achieved by minimising the negative log-likelihood,

$$\mathcal{L}(\boldsymbol{\theta}) = -1 \cdot \frac{1}{M} \cdot \log \left(\prod_{i=1}^{M} \mathrm{P}\left(y_{1:L}^{(i)} | x_{1:N}^{(i)}; \boldsymbol{\theta} \right) \right) \tag{2.43}$$

$$= -1 \cdot \frac{1}{M} \cdot \log \left(\prod_{i=1}^{M} \prod_{j=1}^{L} \mathrm{P}\left(y_{j}^{(i)} | x_{1:N}^{(i)}, y_{1:j-1}^{(i)}; \boldsymbol{\theta} \right) \right) \tag{2.44}$$

$$= -1 \cdot \frac{1}{M} \cdot \sum_{i=1}^{M} \sum_{j=1}^{L} \log \left(\mathrm{P}\left(y_{j}^{(i)} | x_{1:N}^{(i)}, y_{1:j-1}^{(i)}; \boldsymbol{\theta} \right) \right) \tag{2.45}$$

This loss uses teacher forcing [338], where the correct history $y_{1:j-1}^{(i)}$ is used when generating probabilities for the next token. Teacher forcing stabilises training in the early training stages when the model has poor generative abilities and enables parallelised training, as there is no need for a sequential generation. However, during inference, the model operates without reference tokens and is thus free-running. This can lead to exposure bias [270], where errors cause the model to enter parts of the output space it has not encountered during training, potentially leading to poor outputs. One can alternatively use scheduled sampling [19], where during training, the model is gradually exposed to its own predictions, bridging the gap between training and inference. For transformers, scheduled sampling is less effective and often unnecessary since the transformers attend to all previous positions, which makes the system less prone to accumulating errors over time steps [217]. Additionally, since transformers process input tokens in parallel and not sequentially, scheduled sampling can be computationally inefficient.

2.3.2 Stochastic Gradient Descent

The previous section discussed training networks by minimising the loss function. However, since $\boldsymbol{\theta}$ parametrises deep complex non-linear functions, it is typically not possible to find a closed-form solution of the parameters that exactly minimise the loss. To enable practical optimisation of networks, deep learning systems are intentionally designed to be differentiable, and gradient descent [273, 279] methods along with gradient backpropagation [280] can be used to iteratively update the parameters. Let $\boldsymbol{\theta} \in \mathbb{R}^P$ be the vectorised model parameters, where P represents the total number of individual learnable scalar parameters of the system. Gradient descent iteratively estimates the local gradients and then updates the parameters by

taking a step in the direction that minimises the loss locally,

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta \left. \frac{\partial \mathcal{L}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \right|_{\boldsymbol{\theta}_t} \tag{2.46}$$

where η is the learning rate and $\boldsymbol{\theta}_t$ represents the model parameters at iteration t. A drawback of standard gradient descent, though, is that the loss is calculated over the entire dataset, which makes each update step very computationally expensive and impractical for real-world scenarios. Therefore a more popular optimisation approach is stochastic gradient descent (SGD) [25], an extension of gradient descent, where instead of calculating the gradients with respect to the entire training data, the gradients are calculated using a subset of randomly drawn samples. However, a known issue of SGD is its slow convergence due to sudden large changes in gradients caused by the noisy estimates from the mini-batches [258, 305]. Therefore, optimisation approaches typically incorporate a *momentum* term that helps to stabilise the updates and accelerate convergence. The momentum term [254] accumulates past gradients and adds them to the current gradient,

$$\mathbf{m}_t = \gamma \mathbf{m}_{t-1} + \eta \left. \frac{\partial \mathcal{L}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \right|_{\boldsymbol{\theta}_t} \tag{2.47}$$

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \mathbf{m}_t \tag{2.48}$$

where $\mathbf{m}_t$ is the momentum at iteration t and γ is the momentum coefficient that controls how much momentum is carried forward to the current step (typically between 0.9 and 0.99). Improved optimisers that enhance the basic momentum updates include AdaGrad [59], RMSprop [116], Adam [152] and AdamW [194]. For example, AdamW, which has gained considerable traction and has shown to yield good convergence, has updates that take form,

$$\mathbf{m}_t = \beta_1 \mathbf{m}_{t-1} + (1-\beta_1) \left. \frac{\partial \mathcal{L}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \right|_{\boldsymbol{\theta}_t} \qquad \hat{\mathbf{m}}_t = \frac{\mathbf{m}_t}{1-\beta_1} \tag{2.49}$$

$$\mathbf{v}_t = \beta_2 \mathbf{v}_{t-1} + (1-\beta_2) \left(\left. \frac{\partial \mathcal{L}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \right|_{\boldsymbol{\theta}_t} \right)^2 \qquad \hat{\mathbf{v}}_t = \frac{\mathbf{v}_t}{1-\beta_2} \tag{2.50}$$

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta \left(\frac{\hat{\mathbf{m}}}{\sqrt{\hat{\mathbf{v}}_t} + \varepsilon} \right) - \eta \lambda \boldsymbol{\theta}_t \tag{2.51}$$

where β_1 and β_2 are scalar hyperparameters that control the rates of decay for momentum and velocity, respectively, λ is a hyperparameter that controls the weight decay factor, and ε is a small constant added for numerical stability. Note that AdamW decouples the weight

decay from the gradient updates, and without the $\eta\lambda\boldsymbol{\theta}_t$ term, the optimiser would function as the Adam optimiser [152], another widely used optimiser.

2.3.3 Regularisation

The parameters of deep neural networks are known to be prone to overfitting training data, where the model may fit various peculiarities of the training data rather than learning a general predictive rule [52]. Regularisation is a common approach used to counteract this and to incentivise learning simpler, more general solutions. Traditional approaches constrain the model parameters. One such method is L2 regularisation [311], which encourages small parameter values by adding an L2 penalty to the loss function,

$$\mathcal{L}_{reg}(\boldsymbol{\theta}) = \mathcal{L}(\boldsymbol{\theta}) + ||\boldsymbol{\theta}||_2 \tag{2.52}$$

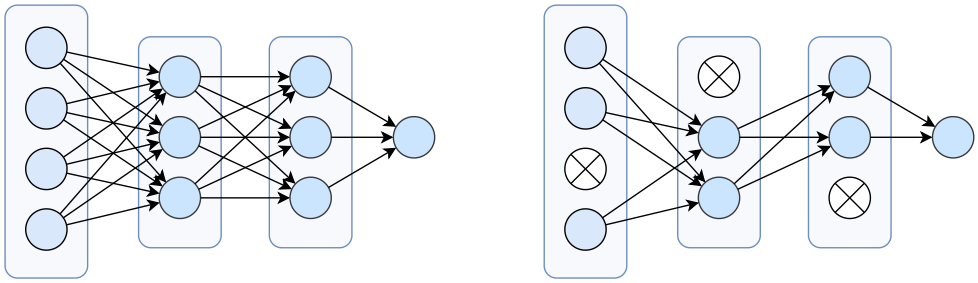
Here, $\boldsymbol{\theta}$ represents the parameters in vector form and $||\mathbf{u}||_n$ represents the Lp norm of vector $\mathbf{u} \in \mathbb{R}^d$, defined as:

$$||\mathbf{u}||_p = \left(\sum_{i=1}^{d} |u_i|^p\right)^{\frac{1}{p}} \tag{2.53}$$

where u_i are the individual elements of the vector $\mathbf{u}$. Alternatively, L1 regularisation [229] can be used to promote sparsity in the network,

$$\mathcal{L}_{reg}(\boldsymbol{\theta}) = \mathcal{L}(\boldsymbol{\theta}) + ||\boldsymbol{\theta}||_1 \tag{2.54}$$

Instead of acting directly on the parameters, one of the main regularisation techniques for transformers is dropout [300]. During training, a model using dropout will randomly set a percentage of the model's neurons to zero, and the model proceeds with the remaining connections. During inference, all neurons are active, but their outputs are scaled to maintain the expected value of the activations, as illustrated in Figure 2.9. Dropout reduces overfitting by encouraging the model to learn effective representations across different subnetworks, preventing the network from relying on any single neuron or feature. The approach has shown to be an effective regularisation technique that aids model training and improves overall system performance and generalisation [120, 77, 320].



(a) Standard DNN (b) DNN after applying dropout

Fig. 2.9 **left:** A FNN with two hidden layers. **right**: The FNN with dropout applied, where crossed-out neurons are randomly sampled and dropped during training.

2.4 Inference Methods

The previous sections discuss common architectures for DNN systems, as well as how training is performed to find optimised parameters $\hat{\boldsymbol{\theta}}$ for LLMs of particular tasks. When autoregressive LLMs are used for real-world generative applications, given an input text $x_{1:N}$, the task is often to return an output text $\hat{y}_{1:L}$. Given the learned model distribution, $\mathtt{P}(\boldsymbol{y}|\boldsymbol{x};\hat{\boldsymbol{\theta}})$, where $\boldsymbol{x}$ refers to the entire text input, $\boldsymbol{x} = x_{1:N}$, and $\hat{\boldsymbol{y}}$ denotes the entire output sequence, $\hat{\boldsymbol{y}} = \hat{y}_{1:L}$, inference aims to find the most likely output sequence,

$$\hat{\boldsymbol{y}} = \arg\max_{\boldsymbol{y}} \mathtt{P}(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\theta}) \tag{2.55}$$

Though for sequence-to-sequence tasks, for a particular back history, probabilities are typically estimated at the token level, $\mathtt{P}(y_j|\hat{y}_{1:j-1},\boldsymbol{x};\boldsymbol{\theta})$. It is infeasible to search over the entire space of possible output sequences and find the maximum likelihood solution. Further, traditional dynamic programming approaches [322] rely on conditional independence assumptions and are inapplicable since small differences in the back-history will cause different token-level output PMFs, even with similar previous tokens. Therefore, in practice, the maximum likelihood search is approximated by one of the following methods:

2.4.1 Greedy Search

Greedy search is a decoding process where at each step, the system makes locally optimal decisions. Given the input and current (incomplete) output, the system selects the token with

the highest associated probability as the next token:

$$\hat{y}_j = \arg\max_{y \in \mathcal{V}} \mathtt{P}(y|\hat{y}_{1:j-1}, \boldsymbol{x}; \boldsymbol{\theta}) \tag{2.56}$$

This process continues until the end-of-sequence token is generated. While computationally efficient, the method does not consider potential future continuations of the sequence, and by sticking to locally optimal decisions, the method may miss out on better overall sequences.

2.4.2 Beam Search

To address the limitations of greedy search, beam search instead maintains a set of k most probable partial sequences, called beams. At each step, each beam is expanded by considering all possible next tokens for the beam, and then the k most probable beams form the new beam, where the probabilities are that of the entire sequence and not just the current token, as shown in Algorithm 1. Beam search offers a trade-off between output quality and computational complexity. A larger beam width achieves a more comprehensive search, though with increased computation time.

Algorithm 1 Beam Search

Initialise beam $B_1 = \{(\texttt{<START>}, 0)\}$
for $j = 2$ to L **do**
    $C_j \leftarrow \emptyset$ ▷ Candidates
    for $(y_{1:j-1}, s)$ in B_{j-1} **do**
        for y in $\mathcal{V}$ **do**
            $s' \leftarrow s + \log \mathtt{P}(y|y_{1:j-1}, \boldsymbol{x}; \boldsymbol{\theta})$
            $C_j \leftarrow C_j \cup \{(y_{1:j-1} \oplus y, s')\}$
        end for
    end for
    $B_j \leftarrow$ top-$k(C_j)$ ▷ Select top k candidates
    if <END> $\in y_{1:j}$ for any $(y_{1:j}, s)$ in B_j **then**
        break
    end if
end for
return top-$1(B_j)$

2.4.3 Sampling

A limitation of deterministic methods such as greedy and beam search is the lack of diversity in responses. As an alternative, sampling introduces randomness into the generation process

by sampling tokens:

$$\hat{y}_j \sim \mathrm{P}(y|\hat{y}_{1:j-1}, \boldsymbol{x}; \boldsymbol{\theta}) \tag{2.57}$$

Sampling can produce more diverse outputs, which can be beneficial for generative NLP tasks where creativity and novelty are required, such as story generation [68] or chatbot response generation [215]. However, sampling may sometimes yield low-quality or incoherent results, especially when sampling from low-probability tokens. To mitigate this, variants like top-K [68] and top-p (nucleus) sampling [124] restrict the sampling space to more likely options. Top-K sampling considers only the K highest probability tokens, while top-p sampling dynamically selects the smallest set of top tokens whose cumulative probability exceeds a threshold p.

2.5 Chapter Summary

This chapter discussed the fundamental building block for deep learning architectures. First, the basic deep learning blocks were introduced, motivating the development of networks from feed-forward networks to transformers. The transformer architecture was then examined in detail, highlighting its various components and architectural variants. The chapter also examined network optimisation and how stochastic gradient descent can be applied to update model weights, searching for the parameters that minimise the training criterion. Lastly, the chapter presented various inference methods for generating responses using trained autoregressive LLMs.

Chapter 3

NLP Foundation Models

Traditionally, deep learning systems were initialised from scratch and trained using supervised data [359, 155]. Supervised training, though, is constrained by the need for large-scale labelled training data, which can be time-consuming and expensive to manually annotate. Transfer learning [309, 235] can be applied to overcome this issue, where instead of training a model from scratch, knowledge from one or more source tasks can be transferred to a target task. Extending this idea further, foundation models [24] are machine learning models that have been trained on a large quantity of diverse data and have gained general abilities over a wide range of use cases. These models acquire powerful representation abilities which can be used as a backbone, providing favourable initialisation for fast convergence to downstream tasks or, in some cases, immediate zero-shot abilities. The concept of a foundation model is not new; it builds on deep neural networks and self-supervised learning, ideas that are decades old [281, 166]. However, with the availability of faster computational resources and the increasing scale of available digital data, the current generation of foundation models has revolutionised many different fields, including Natural Language Processing (NLP) [51, 30, 2], Computer Vision (CV) [55, 262] and Audio Processing [11, 92].

The previous chapter introduced the transformer (§2.2), which has become the dominant model within NLP [241]. In particular, the success of transformers is largely attributed to their effective transfer learning abilities, where pre-training them as Large Language Models (LLMs) unlocks strong language understanding capabilities across a wide range of NLP tasks [51, 2, 227]. This chapter discusses various NLP foundation models, focused on those utilised in this thesis. This chapter first discusses self-supervised learning (SSL) and common objectives used to pre-train the systems (§3.1), details various pre-trained LLMs and their typical downstream use-cases (§3.2), and discusses system alignment (§3.3)

3.1 Self-Supervised Learning

Most data is unlabelled, but the absence of supervised labels makes extracting meaningful insights from unlabelled data challenging. Traditional unsupervised learning approaches aim to discover inherent patterns and structure in data through techniques like clustering and dimensionality reduction, but these methods can be limited in their ability to learn rich, transferable representations. Self-supervised learning (SSL) addresses this limitation by automatically generating supervisory signals from the unlabelled data itself, using pretext tasks to create supervisory signals [53, 98], which can enable models to learn meaningful feature representations that can benefit downstream tasks. The term "pretext" indicates that it does not solve the primary task but serves as a pre-training task for the model to capture better representations within the domain.

This section will describe common pretext tasks used in NLP, highlighting typical objectives such as causal language modelling [264] and masked language modelling (MLM) [51], which are commonly used for pre-training LLMs.

3.1.1 SSL objectives within NLP

Within NLP, there are several forms of self-supervised tasks, most of which harness the strong contextual information present in texts. In particular, the dominant self-supervised tasks are often language modelling based, where during pre-training the system is first trained to model the underlying distribution of words given unidirectional or bidirectional context. This can be a powerful pretext task, as in order to determine which words are likely in particular contexts, the system needs to develop a good understanding of language, linguistic structure and simple world knowledge [51, 276]. Further self-supervised classification objectives such as next sentence prediction (NSP) have also been introduced, however usually with the intention to develop an additional ability that may also be useful for downstream tasks. The following are common self-supervised objectives within NLP:

Causal Language Modelling

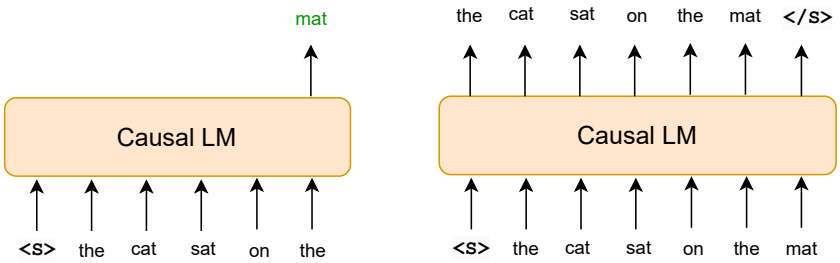


Fig. 3.1 Causal Language Modelling, at both the token-level and parallelised training.

The objective of causal language modelling is to predict the correct next token given the previous context. The input tokens $x_{1:N}$ include the addition of special tokens, such as the start of sequence token, $x_1 =$ <s>, and the end of sequence token $x_N =$ <\s>. For each position, the system predicts the probability distribution of the next token causally, such that for each token in the input sequence, only the preceding tokens are used when predicting the current token. Each position uses the *true* back-history, enabling efficient parallelism of the process via the causal self-attention within the transformer architecture. This process is illustrated in Figure 3.1, where the left diagram illustrates the autoregressive nature of the model, while the right diagram demonstrates that due to the model's causal structure, each position can be trained in parallel using teacher forcing [338]. Given an unlabelled dataset $\mathcal{D}_x = \{\boldsymbol{x}^{(i)}\}_{i=1}^{M}$ of text passages, where $\boldsymbol{x}$ is used to refer to the entire text input, $\boldsymbol{x}^{(i)} = x_{1:N}^{(i)}$, the loss function for causal language modelling takes the form,

$$\mathcal{L}(\boldsymbol{\theta}) = -1 \cdot \frac{1}{M} \cdot \sum_{i=1}^{M} \sum_{k=1}^{N-1} \log \mathrm{P}(x_{k+1}^{(i)} | x_{1:k}^{(i)}; \boldsymbol{\theta}) \tag{3.1}$$

$\boldsymbol{\theta}$ denotes the model parameters, and the dependence of the length of texts N with i is dropped for simplicity. As the loss is often averaged only across samples and not normalised by N, each token contributes equally to the loss, resulting in sequences of different lengths contributing unevenly to the overall loss.

Masked Language Modeling

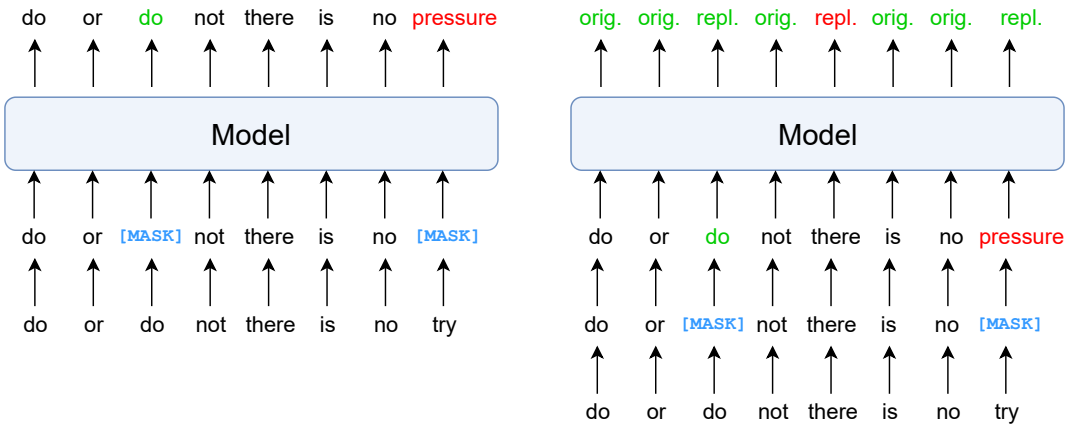


Fig. 3.2 **left:** Masked Language Modelling, where input tokens are randomly masked and the model predicts the original token. **right:** Replaced Token Detection, where a model randomly replaces tokens and the model predicts whether the token was replaced.

The previous causal language modelling learns the unidirectional context of the word, using only information from previous words. The resulting token representations may, therefore, not have full contextual awareness and not account for all available token information. Masked Language Modeling (MLM) [51, 185, 110] is a pre-training methodology that enables systems to learn bidirectional context. In MLM, instead of causally predicting each word, random tokens are masked or corrupted, and the training objective is to recover the original values of these masked tokens. For a set of masked positions for the i-th text passage, $\mathcal{M}^{(i)}$, the MLM objective can be formulated as:

$$\mathcal{L}(\boldsymbol{\theta}) = -1 \cdot \frac{1}{M} \cdot \sum_{i=1}^{M} \sum_{k \in \mathcal{M}^{(i)}} \log \mathrm{P}(x_k^{(i)} | \boldsymbol{x}_{\setminus \mathcal{M}}^{(i)}; \boldsymbol{\theta}), \tag{3.2}$$

where $\boldsymbol{x}_{\setminus \mathcal{M}}^{(i)}$ represents the input sequence with all tokens in $\mathcal{M}^{(i)}$ masked out. This process is depicted in Figure 3.2. MLM systems make predictions under a conditional independence assumption, where each masked token is predicted without considering the predictions at other masked positions. As MLM is not an autoregressive process, a system pre-trained with MLM objectives may not have effective generative abilities. Hence, MLM is primarily used to pre-train discriminative models [51] or systems that provide feature representations for downstream tasks [325, 349].

Replaced Token Detection

The previous MLM pretext task can be seen as training a denoising autoencoder, where the objective is to correct a corrupted input. For MLM, a small subset of the unlabelled input sequence is masked (typically 15%), which may be inefficient as the network only learns from a small subset of the tokens. As an alternative, replaced token detection (RTD) [45] learns to distinguish real input tokens from plausible but synthetic tokens generated by a smaller masked language model, as illustrated in Figure 3.2. Let $\tilde{\boldsymbol{x}}^{(i)}$ be the text generated by the smaller MLM where all tokens in $\mathcal{M}^{(i)}$ have been replaced. The discriminator, parameterised by $\boldsymbol{\theta}$, predicts the probability that the k-th position was replaced, $\mathtt{P}(y_k|\tilde{\boldsymbol{x}}^{(i)};\boldsymbol{\theta})$, where $y_k \in \{0,1\}$ denotes whether the k-th token was replaced. The loss function is then the binary cross-entropy loss:

$$\mathcal{L}(\boldsymbol{\theta}) = \frac{-1}{M}\sum_{i=1}^{M}\frac{1}{N}\sum_{k=1}^{N}\mathbb{1}(k \in \mathcal{M}^{(i)}) \cdot \log \mathtt{P}(y_k = 1|\tilde{\boldsymbol{x}}^{(i)};\boldsymbol{\theta}) + \mathbb{1}(k \notin \mathcal{M}^{(i)}) \cdot \log \mathtt{P}(y_k = 0|\tilde{\boldsymbol{x}}^{(i)};\boldsymbol{\theta})$$

RTD is claimed to have two advantages over MLM: first, MLM has a mismatch between pre-training and fine-tuning, as the `[MASK]` token is essential in pre-training but is never seen in fine-tuning. Secondly, the model now learns from all input tokens instead of just a small masked-out subset, making the process more efficient.

Denoising Autoencoding for Encoder-Decoder Models

For encoder-decoder systems [171, 266], the input is typically processed by the encoder while the decoder autoregressively generates the output. Given an input sequence, $x_{1:N}$, and current generated text, $y_{1:k-1}$, the encoder-decoder model learns to predict the probability of the next generated token given the input text and previously generated context, $\mathtt{P}(y_k|y_{1:k-1}, x_{1:N};\boldsymbol{\theta})$. One can extend denoising autoencoding objectives [321] to pre-train encoder-decoder systems. Similar to MLM, the input sequence can be corrupted using token masking, where tokens or contiguous text spans can be replaced with a single masked token. Given a corrupted sequence $\boldsymbol{x}^{(i)}_{\setminus\mathcal{M}}$, the encoder-decoder system can learn to reconstruct the original, uncorrupted input sequence $\boldsymbol{x}^{(i)}$:

$$\mathcal{L}(\boldsymbol{\theta}) = -1 \cdot \frac{1}{M} \cdot \sum_{i=1}^{M} \log \mathtt{P}(\boldsymbol{x}^{(i)}|\boldsymbol{x}^{(i)}_{\setminus\mathcal{M}};\boldsymbol{\theta}) \tag{3.3}$$

$$= -1 \cdot \frac{1}{M} \cdot \sum_{i=1}^{M}\sum_{k=1}^{N} \log \mathtt{P}(x^{(i)}_k|x^{(i)}_{1:k-1}, \boldsymbol{x}^{(i)}_{\setminus\mathcal{M}};\boldsymbol{\theta}) \tag{3.4}$$

Alternatively, instead of autoregressively generating the entire sequence, the decoder can be trained to predict only the masked tokens within the sequence (which is depicted in Figure 3.5a).

Next Sentence Prediction

Many NLP tasks require a system to reason about the relationship between pairs of text. Therefore, in addition to MLM, some systems also use the Next Sentence Prediction (NSP) [51] pre-training task. Here, two texts are separated by a special token, and the objective of the system is to determine whether the second text logically follows from the first text (which can be seen visualised in Figure 3.4). This helps the model understand the relationships between sentences and is hypothesised to be beneficial for tasks such as question answering and Natural Language Inference [26].

3.2 NLP Foundation Models

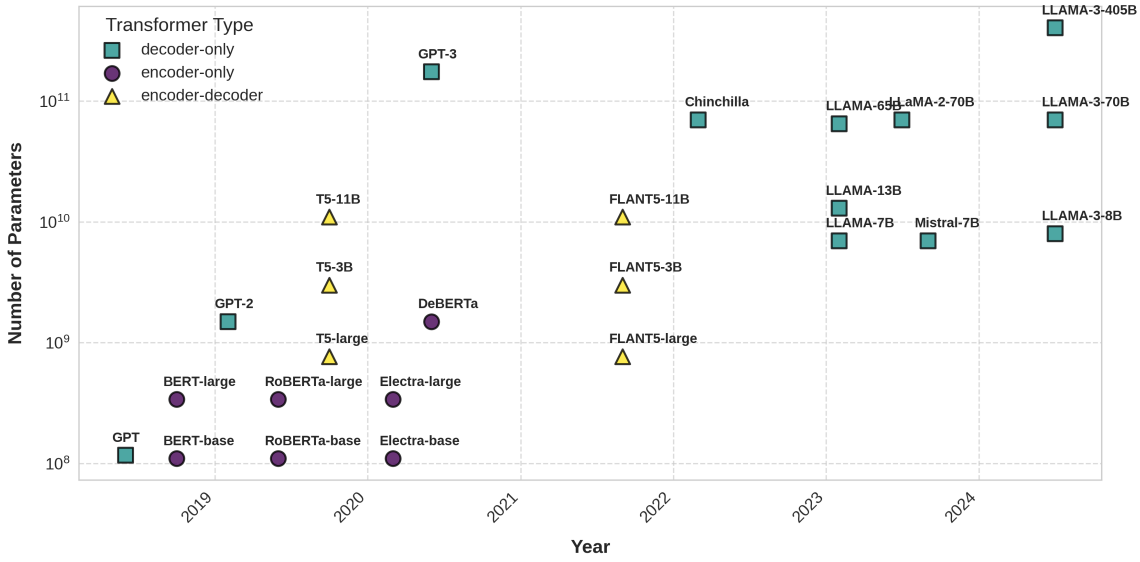


Fig. 3.3 Timeline of the size and architecture of popular pre-trained transformers.

Having established common pretext tasks, this section now provides the details of various pre-trained transformers. Pre-trained transformers have emerged as the dominant approach for NLP tasks. The transformer's prevalence over recurrent networks is primarily attributed to two main factors: firstly, RNNs [63] and many of their variants face challenges in modelling long-range dependencies due to vanishing gradients [121]. Secondly, RNNs are inherently

sequential, and the hidden representations have to be iteratively calculated at each time step, which makes training computationally expensive and limits the maximum practical size of these models. In contrast, the parallel nature of transformers enables these systems to scale rapidly in size, with empirical scaling laws [147] demonstrating consistent performance improvements [41, 123]. Transformers have, therefore, scaled up rapidly; the first iteration of the Generative Pre-trained Transformer (GPT) [264] had 117 million parameters, while 5 years later, GPT-4 [2] is hypothesised to have over 1 trillion parameters [207]. With the accelerated rate of development in the field, state-of-the-art models are continually evolving and new NLP foundation models are frequently released, with Figure 3.3 presenting a timeline of recent popular pre-trained LLM releases. The rapid progression in the field has led to new models frequently outperforming their predecessors, with systems only transiently being state-of-the-art before being displaced. This section examines several prominent and influential NLP foundation models, describing their architecture, self-supervised training methodologies and downstream applications. These models will support the experiments throughout the rest of this thesis, and so an in-depth understanding of these systems is provided.

3.2.1 Encoder-only Foundation Models

Section 2.2 presented an overview of various transformer architectures. For tasks that involve analysing and classifying textual inputs rather than directly generating text, the encoder-only architecture can be an effective choice. The central component of this architecture is the multi-head self-attention mechanism, which is non-causal and where each token in the input sequence can attend to all other positions. This bidirectional context enables these models to capture rich, context-aware representations. However, a drawback is that these systems are not designed for autoregressive text generation. Therefore, instead of directly using the language modelling abilities of the foundation model, encoder-only foundation models are typically used to provide effective representations that can be adapted to particular tasks.

To align with their architectural design, encoder-only transformers are typically pre-trained using denoising objectives, such as masked language modelling or replaced token detection. These pre-training strategies enable the model to develop a representation of language structure and semantics [276], though they have to be fine-tuned for particular tasks (which will be discussed in the next chapter, §4.1.2). This section covers four popular pre-trained encoder-only transformers: BERT [51], RoBERTa [185], ELECTRA [45] and DeBERTa [110].

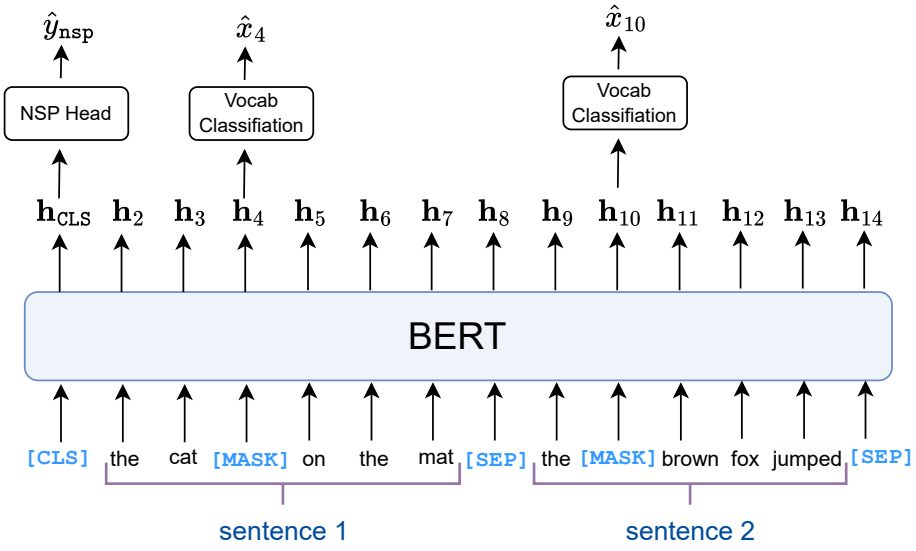


Fig. 3.4 BERT pre-training set up. Two sentences are combined with special tokens, in this example yielding $x_{1:14}$. The system predicts whether the second sentence follows the first ($\hat{y}_{\text{nsp}}$) as well as the identity of each masked token.

BERT

The first and possibly the most well-known pre-trained encoder-only LLM is the Bidirectional Encoder Representations from Transformers (BERT) [51]. Pre-training in BERT is achieved by first randomly selecting 15% of the input tokens, where 80% of these tokens are masked, 10% corrupted to a random token and 10% left as the original token. BERT is trained with the MLM pre-training objective and further incorporates the Next Sentence Prediction (NSP) as a second pre-training objective. Beyond the `[MASK]` token, BERT uses two further special tokens, the `[CLS]` token, which is used to mark the start of the input, and the `[SEP]` token, which is used to both separate the two input texts at training (for NSP) and mark the end of the input sequence (illustrated in Figure 3.4). The model is pre-trained on the BooksCorpus (800M words) and English Wikipedia (2,500M words), using long continuous text sequences to capture broader context. The model is available in two different sizes: BERT-base, which has 110M parameters, or BERT-large, which has 330M parameters.

RoBERTa

RoBERTa [185] (Robustly Optimised BERT Approach) is another MLM-based system that builds upon BERT but with several modifications. It eliminates the NSP objective after empirical studies revealed that the NSP objective had limited performance improvements.

The model is hence trained using single contiguous text sequences and not using two separate texts as done in BERT, enabling the system to model longer contextual dependencies. Additionally, RoBERTa is trained on substantially more data, combining datasets such as BooksCorpus and CommonCrawl News, which combined is approximately ten times larger (30B words) than the amount of data used to train BERT. RoBERTa is trained for more training epochs and, similar to BERT, is available as RoBERTa-base (110M parameters) or RoBERTa-large (330M parameters).

ELECTRA

ELECTRA [45] was the first pre-trained transformer to use replaced token detection (RTD) pre-training. Similar to BERT, 15% of the input tokens are masked and replaced, but for RTD, the discriminator is trained to determine which tokens were replaced by the generator. The generator and discriminator are trained jointly in an adversarial manner, alternating between updating the generator and the discriminator, and the discriminator is then used as the foundation model. ELECTRA is trained on the same pre-training data as BERT (Wikipedia and BooksCorpus) and is available as ELECTRA-base (110M) or ELECTRA-large (330M).

DeBERTa

Decoding-enhanced BERT (DeBERTa) is another extension of BERT. The first DeBERTa modification is to decouple token representations from the positional embeddings. This approach allows the attention mechanism to consider three types of interactions: between word representations themselves (content-to-content), between word representations and their relative positions (content-to-position), and between relative positions and word representations (position-to-content). Additionally, DeBERTa incorporates absolute positional embedding in the final decoding layer before the softmax. DeBERTa is trained using 80GB of text data (15M words), where the first version was trained using masked language modelling, and later versions using replaced token detection. DeBERTa is available as DeBERTa-base (140M), DeBERTa-large (430M) and DeBERTa-XL (1.5B).

3.2.2 Decoder-only Foundation Models

The next architecture considered is the decoder-only transformer. Decoder-only transformers are characterised by their causal attention mechanism, where each token only attends to previous tokens, enabling autoregressive text generation. In contrast to encoder-only models, which are often trained as MLMs, decoder-only transformers are usually trained using causal language modelling. This involves sequentially predicting the next token in the text sequence,

enabling these systems to generate text by iteratively predicting and appending tokens to the growing output sequence. These models have gained significant popularity [264, 314, 139], as they use a single transformer stack for both feature representation and generation. This eliminates the encoder stack, making decoder-only models more computationally efficient than encoder-decoder architectures (such as T5 [266]). With the growing scale of models, decoder-only transformers are progressively becoming the most common transformer architecture, with nearly all recent state-of-the-art systems adopting this architecture [314, 139, 308]. Although many decoder-only transformers are similar architecturally, they differ in size, training data, and alignment. This section will discuss various popular decoder-only foundation models, including the GPT series [264, 265, 30], Llama [314], and Mistral [139].

GPT

The concept of a pre-trained transformer was first introduced by OpenAI with the release of the Generative Pre-trained Transformer (GPT) [264]. The inaugural model, GPT-1, featured a 12-layer decoder-only transformer with 117 million parameters, trained on the BookCorpus dataset, containing approximately 800M words. GPT-2 [265] was next in the GPT series, expanding on its predecessor by increasing the parameter count to 1.5 billion and training on the larger WebText corpus (10B words). This progression continued with GPT-3, which scaled the model up to 175 billion parameters and trained the system on diverse data totalling 570 GB of text. The substantial increase in model size and training data enabled GPT-3 [30] to achieve unprecedented abilities, which led to widespread attention from the research community and the general public. Next, GPT-4 was released and demonstrated another large jump in performance, though many technical details of this system remain undisclosed. A key advancement in GPT-4 and later releases of GPT-3 were their instruction-following capabilities [234], which could be prompted zero-shot/few-shot for downstream tasks, which will be discussed in greater detail later in this chapter (§3.3.4). Multiple versions of GPT-3 and GPT-4 were released by OpenAI, though only accessible by closed black-box APIs. The parameters and technical details of GPT-3 and GPT-4 are not publicly available.

Llama

Llama [314] was a decoder-only transformer released by Meta. Unlike OpenAI's closed-source GPT models, Llama is an open-source language model with its architecture publicly documented and weights available for download by researchers and developers alike. Three versions have been released to date: Llama1, Llama2 and Llama3. The models were released within a year, with improvements seen even for models of the same size, demonstrating that

data-centric approaches can also lead to improved performance. The models are available in different sizes: Llama1 [313] (7B, 13B, 33B, 65B), Llama2 [314] (7B, 13B, 70B)) and Llama3 [58] (8B, 70B). Llama1 was trained on 1.4T tokens, llama2 on 1.8T tokens, and LLama3 on 15T multilingual tokens. Llama uses rotary positional embeddings. These models are also available in instruction-following checkpoints, where the base system has been further instruction-tuned to respond effectively to specific prompts and instructions (§3.3.3).

Mistral

Mistral [139], released by Mistral AI, is an open-source decoder-only transformer model. The focus of Mistral's design is on efficiency, aiming to achieve competitive performance with a moderate parameter count. This is achieved by leveraging sliding window attention [37], which limits the attention mechanism to a local window around each token, and grouped-query attention [3], where the queries from different heads share the same key-value projection weights. Mistral is available as a 7B model trained on 1 trillion tokens, as well as a version that is further instruction-tuned.

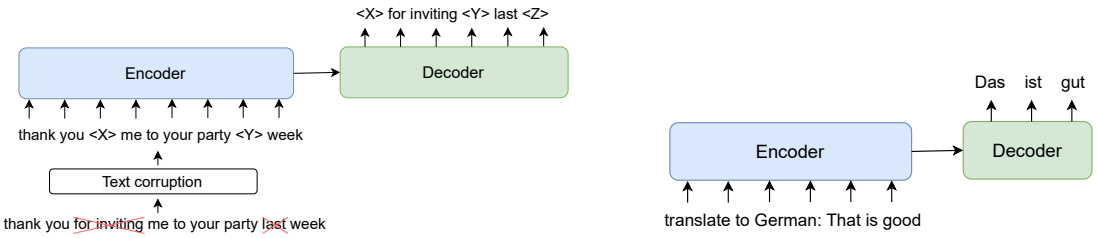
3.2.3 Encoder-Decoder Foundation Models

The final type of systems considered are pre-trained models based on the encoder-decoder transformer architecture. This architecture, proposed in the original transformer paper [320], features an encoder that processes the input sequence and a decoder that autoregressively generates tokens. While recent pre-trained generative transformers have predominantly been decoder-only, there have been several popular pre-trained encoder-decoder transformers. One notable example is T5 [266], which will be discussed next.

T5

T5 (Text-to-Text Transfer Transformer) [266] is an encoder-decoder model pre-trained on a mixture of both unsupervised and supervised tasks, where all tasks are formatted in a unified text-to-text framework. For the self-supervised training, 15% of the tokens are randomly removed and replaced with individual sentinel tokens (where several consecutive removed tokens are grouped into a single sentinel token). While for supervised training, tasks from GLUE [325] and SuperGLUE [328] are converted into a text-to-text format, and the system is trained to produce the targets given the inputs. Figure 3.5 depicts these pre-training tasks. The input to the encoder is the corrupted sentence, while the decoder has to predict the original sentence. T5 was trained on the Colossal Clean Crawled Corpus (C4), which

contains 750GB of text (174B words). T5 uses learned relative scalar embeddings and has a maximum token length of 512. The models are available in five sizes: T5-small (60M), T5-base (220M), T5-large (770M), T5-xl (3B), and T5-xxl (11B). The T5 models were also later adapted to FlanT5 [42, 193], which are T5 models further instruction-tuned (§3.3.3) to have instruction-following capabilities, a property which will be discussed in the next section.



(a) Unsupervised Pre-training Obective for T5. (b) Supervised Pre-training Task for T5.

Fig. 3.5 Illustration of the T5 pre-training set ups.

3.3 Alignment

The NLP foundation LLMs discussed in the previous section (§3.2) were pre-trained with language modelling objectives on vast amounts of textual data. Pre-training provides the system with powerful representations that are useful for downstream tasks and enables generative systems to generate coherent and convincing responses. Although unsupervised pre-training provides good initialisation for the systems, these unsupervised tasks may not alone be sufficient for achieving good performance towards bespoke NLP tasks. This section, therefore, considers how to adapt models towards particular tasks and *aligning* the model with desired capabilities. In this context, alignment refers to the process of modifying the model's behaviour to better match desired outcomes, intended use cases and human preferences. Several alignment approaches are discussed, including aligning NLP foundation models by fine-tuning using domain-specific supervised data, instruction tuning [193, 348] and reinforcement learning by human feedback (RLHF) [234]. The final two approaches describe the process of developing *instruction-following* LLMs, where models can then be leveraged for general tasks through zero-shot or few-shot natural language instruction prompting [265, 30, 80].

3.3.1 Task-Specific Finetuning

Traditional deep learning systems are often trained end-to-end, where the system jointly learns meaningful feature representations and inference predictions by training all model parameters on labelled data. When pre-trained transformers were first introduced [51], the models were adapted to particular tasks with the 'pre-train and fine-tune' paradigm. Here, instead of randomly initialising the model weights, the weights from *pre-training* are used as a starting point to initialise the model. Pre-training is hypothesised to provide useful feature representations that can effectively be adapted to new tasks. In the second *fine-tuning* stage, all model parameters are then updated to minimise a particular task objective. Fine-tuning typically updates all model parameters, as well as the parameters of any task-specific heads required to convert the representations to output decisions. The 'pre-train and fine-tune' paradigm offers many advantages over training from scratch, such as better data efficiency, better out-of-domain robustness, as well as better model performance [51, 185, 113]. The practical details of the methodology will be discussed in greater detail later in section 4.1.2.

3.3.2 Emergent Abilities of LLMs

Although initially task-specific fine-tuning was the dominant paradigm to adapt models to tasks, researchers observed that due to the vast and diverse data seen in pre-training, LLMs could exhibit 'emergent' abilities, performing tasks they were not explicitly trained to do. Examples of emergent abilities include:

- **Zero-shot performance by task reframing**: Emergent abilities were first observed by framing tasks in ways similar to the pre-training task. BERT [51] demonstrated promising results when tasks were presented in a cloze-style format [80, 286], while GPT-2 [265] showed unexpected summarisation capabilities when prompted with 'tldr:' (which is a common online abbreviation for *too long didn't read*).

- **In-context learning**: As models grew larger and were trained on more diverse datasets, new capabilities were uncovered. GPT-3 [30] was found to be able to perform in-context learning, where including a few input-output examples in the prompt allowed the model to perform a select target downstream task.

- **Chain-of-thought reasoning**: By prompting LLMs to think step-by-step and provide its chain-of-thought reasoning [331], model demonstrated stronger abilities for complex reasoning tasks.

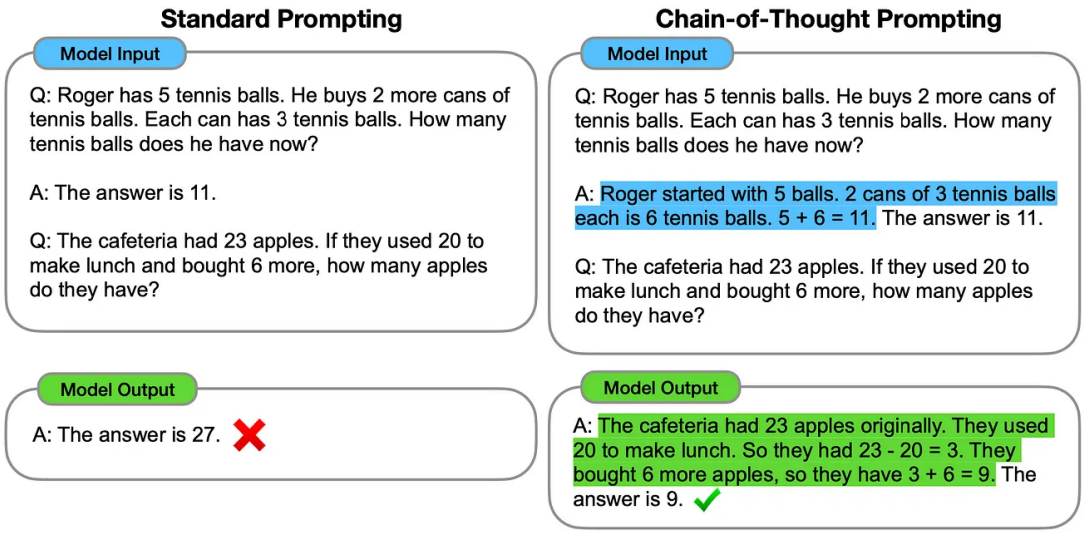


Fig. 3.6 Illustration of emergent abilities that LLMs can demonstrate. **left:** standard prompting with in-context example (where there is one example). **right:** the equivalent process where the examples now probe for chain-of-thought reasoning. The example was taken from [331]

.

Figure 3.6 illustrates some of these prompting approaches. These applications demonstrated that pre-training could provide a strong foundation of general language understanding and generation for these LLMs, enabling these models to adapt to various tasks without task-specific training. While these findings illustrated that there was potential for a single general model to be used for a diverse array of tasks, both zero-shot and few-shot methods lagged behind state-of-the-art methods achieved through direct fine-tuning. Further, reliably guiding the models to accurately execute tasks remained a substantial challenge. This highlighted the need for new techniques to bridge the gap between general model ability and task-specific expertise.

3.3.3 Instruction Tuning

Instruction tuning [330] emerged as a promising solution for systems to achieve good performance while being general and applicable to a wide range of diverse tasks. By noting that models demonstrated impressive performance when directly fine-tuned to tasks, an intuitive starting point was to fine-tune a single LLM on multiple tasks simultaneously in a multi-task fashion. The key innovation in instruction tuning lies in its unified task representation; by framing diverse NLP tasks as natural language instructions, researchers created a consistent format that allows a single model to handle multiple tasks without

task-specific architectures [154] or output heads. This approach not only simplifies the model architecture but can also enable the model to generalise to new unseen tasks more effectively. The instruction tuning process can be summarised as follows:

1. **Dataset Collection** The first step is to construct the instruction tuning dataset. These datasets leverage the vast number of existing NLP datasets available across diverse tasks [325, 160, 275]. For each selected benchmark, human annotators create natural language templates to seamlessly frame all tasks in a consistent instruction-output format. Popular instruction tuning datasets include FLAN [193, 330], SuperNaturalInstructions [328] and Prompt-Source [9], which each adapt a diverse range of tasks with their own curated instruction sets. For instance, the FLAN collection [193] encompasses over 1,800 distinct NLP tasks in its held-in training set, each of which is reformulated with natural language instructions.

2. **Instruction Tuning** The instruction tuning process [330] fine-tunes (§3.3.1) a pre-trained language model on all the instruction-formatted datasets. The model learns to generate the output labels given an input instruction (the input framed with the task instruction), for the training data points of the held-in dataset. By fine-tuning to all tasks simultaneously, with natural task instructions included in the input, the same model learns to generalise to many NLP tasks and be guided using natural instruction prompts [193, 330].

Instruction tuning emerged as an effective approach for model alignment in NLP. On held-in training tasks, instruction-tuned systems achieved performance that rivals, sometimes surpassing, direct task fine-tuning. These systems also exhibit good generalisation abilities, performing well to unseen tasks from the held-out set. Despite these strengths, instruction tuning faces several limitations, including:

1. **Output Diversity Challenges**: Many generative tasks may have multiple valid, high-quality outputs for a particular input. Supervised fine-tuning treats the reference as the single correct response and penalises other valid response continuations, potentially limiting the diversity of the system's output.

2. **Lack of Nuanced Understanding**: Maximum-likelihood training treats all incorrect tokens equally without considering the degree or nature of the error. For example, given the reference "Avatar is a fantasy movie," the response "Avatar is an adventure movie" is more acceptable than "Avatar is a fantasy musical", but traditional training methods would assign a similar loss to both.

3. **Dataset Contraints**: Instruction tuning relies heavily on supervised data, which requires manual annotation. While there is currently a diverse range of labelled NLP benchmarks available, incorporating new tasks into instruction tuning datasets necessitates additional manual annotation efforts.

Reinforcement Learning with Human Feedback [234] (RLHF), which will be discussed in the next section, addresses these limitations and can be applied to achieve better alignment than purely with supervised instruction tuning

3.3.4 Reinforcement Learning by Human Feedback

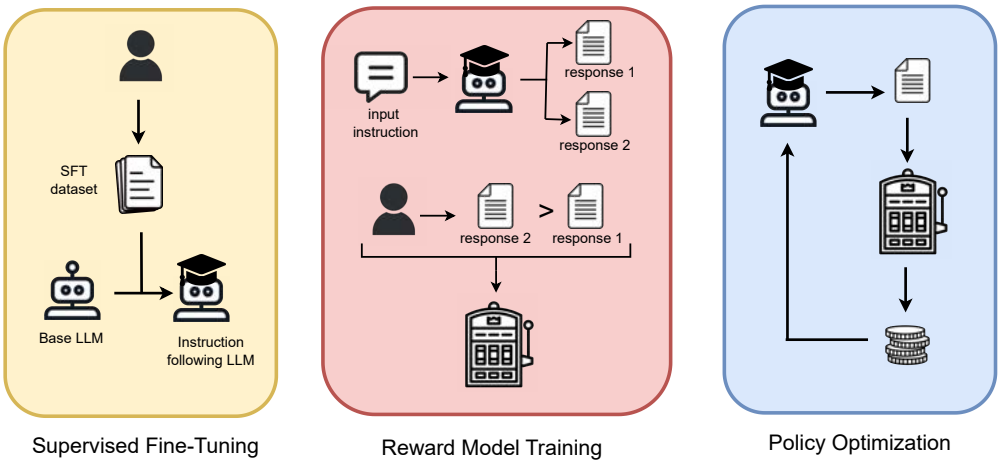


Fig. 3.7 Diagram depicting the three stages of reinforcement learning by human feedback.

Reinforcement Learning from Human Feedback (RLHF) is an alignment technique that addresses the limitations of traditional instruction tuning alignment. While supervised instruction tuning relies on a single reference for each input, RLHF captures a more nuanced understanding of human preferences across a diverse range of possible outputs and trains systems to generate responses more closely aligned with human values and preferences. The RLHF alignment pipeline, depicted in Figure 3.7, typically consists of three stages: supervised fine-tuning (instruction tuning), reward model training and policy optimisation.

Stage 1: Supervised Fine-Tuning

The first step is to equip the model with basic instruction-following abilities. This is achieved by performing instruction tuning where, as discussed in section 3.3.3, the model is trained to emulate labelled data of diverse input requests and gold standard reference responses. This

stage serves as a foundation for the subsequent RLHF stages, providing the model with the capabilities to initially generate reasonable responses.

Stage 2: Reward Model Training

A central part of RLHF is the reward model. The reward model serves as a proxy for human preferences in the training process and aims to capture the nuanced and often subjective aspects of what makes a language model's output high-quality. Once trained, the reward model can assign a scalar value to any generated output, providing a clear optimisation target for the language model.

To train the model, one has to first generate output samples from the model for human annotators to evaluate. Given the current LLM parameters $\boldsymbol{\theta}$, inputs $\boldsymbol{x}$ can be sampled from the distribution of possible user requests, $\boldsymbol{x} \sim \mathtt{P}(\boldsymbol{x})$, where $\boldsymbol{x}$ contains both the task description and input content. A corresponding system response can then be sampled from the LLM, $\boldsymbol{y} \sim \mathtt{P}(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\theta})$. The reward model $\mathtt{R}(\boldsymbol{y},\boldsymbol{x};\boldsymbol{\theta}_{\mathtt{RM}})$ aims to determine the quality of the response $\boldsymbol{y}$ given the input instruction $\boldsymbol{x}$. To train the reward model, given an input instruction $\boldsymbol{x}$ and multiple model samples $\boldsymbol{y}_{1:N}$, human evaluators are given a pair of responses and asked which one is preferred. This creates the dataset of human preferences $\mathcal{D}_{RM} = \{(\boldsymbol{x}^{(i)},\boldsymbol{y}_w^{(i)},\boldsymbol{y}_l^{(i)})\}_{i=1}^{M}$, where $\boldsymbol{y}_w$ denotes the preferred responses and $\boldsymbol{y}_l$ the less favoured response. The reward model is trained to maximise the likelihood of the human preferences under the Bradley-Terry [27] model,

$$\mathcal{L}_{RM}(\boldsymbol{\theta}_{RM}) = -1 \cdot \frac{1}{M} \cdot \sum_{i=1}^{M} \log \sigma\big(R(\boldsymbol{y}_w^{(i)},\boldsymbol{x}^{(i)};\boldsymbol{\theta}_{RM}) - R(\boldsymbol{y}_l^{(i)},\boldsymbol{x}^{(i)};\boldsymbol{\theta}_{RM})\big) \tag{3.5}$$

where $\sigma(s)$ is the sigmoid function, $\sigma(s) = 1/(1+e^{-s})$. In practice, to speed up the data collection, four to nine outputs from the same input are generated, and labellers are asked to rank the responses. The update step is done for the entire batch but normalised by the number of comparisons.

Stage 3: Policy Optimisation

With a trained reward model, the final stage involves updating the language model to generate responses aligned with human preferences, as captured by the reward model. This is typically achieved using Proximal Policy Optimisation (PPO), a reinforcement learning algorithm well-suited for this task. Let $\mathtt{P}_{\theta}(\boldsymbol{x},\boldsymbol{y}) = \mathtt{P}(\boldsymbol{x})\mathtt{P}(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\theta})$, the objective function can be formulated as

maximising the expected reward of the model under samples from $\mathtt{P}_\theta(\boldsymbol{x},\boldsymbol{y})$:

$$\mathcal{L}_{PPO}(\boldsymbol{\theta}) = -1 \cdot \mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})\sim\mathtt{P}_{\boldsymbol{\theta}}(\boldsymbol{x},\boldsymbol{y})}\left[R(\boldsymbol{x},\boldsymbol{y}) + \log\frac{\mathtt{P}_{lm}(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\theta})}{\mathtt{P}_{lm}(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\theta}_0)}\right] \tag{3.6}$$

where the second term is a Kullback–Leibler divergence [157] regularisation term to ensure that the model does not overly reward-hack the reward model, and $\boldsymbol{\theta}_0$ the model parameters directly after instruction tuning. As with standard optimisation approaches, the parameter's gradients with respect to the loss can be calculated, and the weight can be updated using SGD (§2.3.2):

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \nabla_{\boldsymbol{\theta}}\mathbb{E}_{\boldsymbol{x},\boldsymbol{y}\sim\mathtt{P}_\theta(\boldsymbol{x},\boldsymbol{y})}[R(\boldsymbol{x},\boldsymbol{y}) + \mathcal{L}_{reg}(\boldsymbol{\theta})] \tag{3.7}$$

Where $\mathcal{L}_{reg}(\boldsymbol{\theta})$ refers to the KL regularisation term. To compute this gradient, we can apply the REINFORCE algorithm [337] using the log-derivative trick:

$$\nabla_{\boldsymbol{\theta}}\sum_{\boldsymbol{y}}\mathtt{P}(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\theta})R(\boldsymbol{x},\boldsymbol{y}) = \sum_{\boldsymbol{y}}R(\boldsymbol{x},\boldsymbol{y})\nabla_{\boldsymbol{\theta}}\mathtt{P}(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\theta}) = \sum_{\boldsymbol{y}}R(\boldsymbol{x},\boldsymbol{y})\mathtt{P}(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\theta})\nabla_{\boldsymbol{\theta}}\log\mathtt{P}(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\theta})$$

which yields,

$$\nabla_{\boldsymbol{\theta}}\mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})\sim\mathtt{P}_\theta(\boldsymbol{x},\boldsymbol{y})}[R(\boldsymbol{x},\boldsymbol{y})] = \mathbb{E}_{\boldsymbol{x}\sim\mathtt{P}(\boldsymbol{x})}\mathbb{E}_{\boldsymbol{y}\sim\mathtt{P}(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\theta})}[R(\boldsymbol{x},\boldsymbol{y})\nabla_{\boldsymbol{\theta}}\log\mathtt{P}(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\theta})] \tag{3.8}$$

This can be approximated using Monte Carlo sampling, allowing for practical gradient descent. The reward model and LLM are trained sequentially and iteratively for several training epochs.

3.4 Chapter Summary

This chapter introduced various NLP foundation models. The pre-trained systems were all based on transformers, although based on various transformer architectures, including the encoder-only, decoder-only, and encoder-decoder transformers. The chapter also listed key pre-training self-supervised training tasks and discussed the details of popular pre-trained LLMs. Finally, various methods of aligning systems to particular tasks were considered, ranging from supervised fine-tuning, emergent prompting abilities, instruction-tuning and RLHF. The pre-trained models introduced in this chapter will serve as the foundation models used throughout the rest of the thesis.

Chapter 4

Classification, Scoring and Ranking

The previous chapter introduced the concept of a foundation model, a machine-learning model trained on large-scale data with broad applicability across various downstream tasks [24]. Two widely researched and adopted applications of NLP foundation models are text classification [54] and text evaluation [174], both of which have significant practical implications. Both classification and text evaluation are key tasks frequently performed by NLP systems, requiring robust and unbiased outputs, preferably achieved in a zero-shot fashion. This chapter provides the background for these tasks and establishes approaches to leverage NLP foundation models for text classification and text quality assessment. The subsequent chapters will analyse the methods introduced in this chapter and examine potential biases and spurious properties when these systems are deployed in real-world scenarios. Section 4.1 first considers the text classification and describes standard methodologies used to adapt NLP foundation models for these tasks. Section 4.2 then examines text quality assessment and covers the details of existing NLG assessment approaches, from reference-based, supervised trained, and zero-shot prompted approaches. Section 4.3 then provides a background of comparative assessment, discussing standard models used to analyse the outcomes of pairwise comparisons.

4.1 Text Classification

The objective of text classification [54] is to assign an input text to a category within a fixed set of classes. This enables easy, cost-efficient, and scalable data analysis by transforming unstructured textual information into structured insights [78]. The popularity of text classification stems from its direct applicability to a myriad of practical problems, such as analysing the evolving sentiment of users on a platform [342], identifying and filtering fraudulent reviews [95, 230], or extracting useful input features in a pipeline [172]. Moreover,

the discrete nature of classification outputs enables easy and convenient evaluation, with metrics such as accuracy, F_1 score [318] and AUC-ROC [107] effective at reliably evaluating systems. This ease of evaluation has contributed to the prevalence of classification tasks in AI research, where system capabilities are often assessed based on performance across various classification benchmarks [325, 111]. This section will discuss the common application of text classification and explores methodologies for leveraging pre-trained transformers for text classification, laying the foundation for both chapter 5 and chapter 6.

4.1.1 Applications of Text Classification

Text classification is applicable to a wide range of tasks and serves various downstream objectives. Its widespread adoption is not only due to the practical utility of automatic classification in real-world settings but also due to its central role in accessing the natural language understanding (NLU) abilities of systems. We will consider several popular NLP tasks that have been widely adopted and studied by the research community. The main tasks of interest, which will be studied in greater detail in later chapters in this thesis, are sentiment classification, natural language inference and multiple-choice question-answering:

1. **Sentiment Analysis**: This task involves examining the emotional tone conveyed in digital text, categorising it either as binary sentiment (positive or negative) or within a wider spectrum of emotions (e.g. angry, sad, happy, etc.). Its applications include monitoring customer satisfaction and brand image, content personalisation or political analysis [284]. Benchmark datasets such as IMDB [200], Rotten Tomatoes [237], and SST-2 [298] each serve as standard evaluation datasets within NLU research.

2. **Natural Language Inference (NLI)**: NLI tasks assess a system's ability to determine the logical relationship between two sentences. Given two texts, a premise and a hypothesis, the objective is to determine whether the hypothesis entails, contradicts, or is neutral to the premise. This capability underpins various NLP applications, including information retrieval, question-answering, and summarisation. Standard NLI datasets include SNLI [26] and MNLI [336].

3. **Multiple-Choice Question Answering (MCQA)**: MCQA tasks require a system to select the correct answer from a set of options and are used to evaluate comprehension, factual recall or reasoning abilities. For example, typical multiple choice reading comprehension (MCRC) datasets provide the system with a contextual passage, a question and a set of 4 options, and the system is then tasked with determining which of the options is correct based on the passage. These tasks vary in complexity and

type, ranging from passage-based reading comprehension to assessments of factual knowledge. As NLP models have advanced, more challenging MCQA datasets have emerged, including RACE++ [160], CosmosQA [130], ARC [46], and ReClor [343], pushing the boundaries of language understanding and reasoning capabilities.

Examples of the inputs and objectives for each task are shown in Figure 4.1.

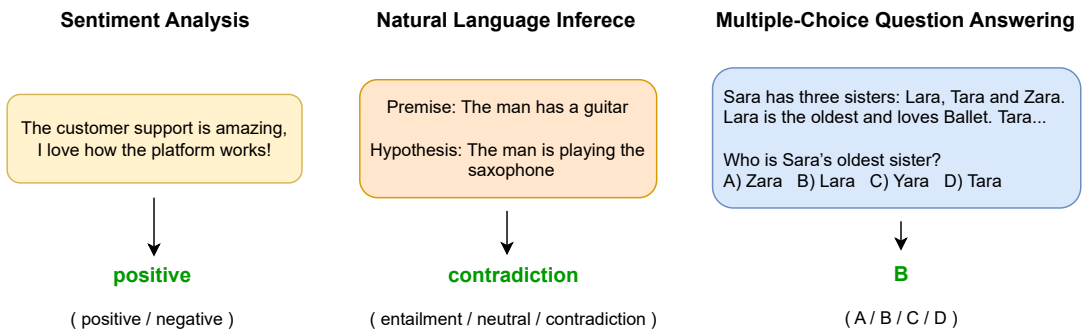


Fig. 4.1 Examples of NLP tasks, showing the input, correct answer and possible classes.

4.1.2 Fine-Tuned Classification

There are several methodologies for leveraging NLP foundation models for particular tasks. An early and highly popular methodology is the 'pre-train and fine-tune' paradigm [51]. This approach is motivated by the finding that the pre-trained parameters, $\boldsymbol{\theta}_0$, offer robust representations and serve as a favourable starting point that enables fast convergence to down-stream tasks [127, 140]. Therefore, one can use the foundation model's pre-training parameters, $\boldsymbol{\theta}_0$, as an initialisation point and then 'fine-tune' the parameters by further training the system on labelled task data. Fine-tuning is the process of updating the model parameters, $\boldsymbol{\theta}_0 \rightarrow \hat{\boldsymbol{\theta}}$, where $\hat{\boldsymbol{\theta}}$ represents the model parameters after fine-tuning, as well as introducing any possible task-specific heads.

For a task $\mathcal{T}$, assume there is the set of output classes, $\mathcal{Y} = \{\omega_1, \omega_2, \ldots, \omega_K\}$, where K is the total number of classes. The pre-trained system can be configured to model the probability of assigning a specific class to the input text, $\mathrm{P}(y|\boldsymbol{x};\boldsymbol{\theta})$. After initialising to $\boldsymbol{\theta}_0$, fine-tuning is performed by optimising all model weights on the task training criterion, which for classification is usually the cross entropy loss. Given a labelled task dataset $\mathcal{D} = \{(\boldsymbol{x}^{(i)}, y^{(i)})\}_{i=1}^{M}$, the model parameters that minimise the negative log-likelihood loss

(which is equivalent to minimising the cross-entropy with hard labels) can be defined as,

$$\mathcal{L}(\boldsymbol{\theta}) = -1 \cdot \frac{1}{M} \cdot \sum_{i=1}^{M} \log \mathtt{P}(y^{(i)}|\boldsymbol{x}^{(i)};\boldsymbol{\theta}) \tag{4.1}$$

where the fine-tuned parameters $\hat{\boldsymbol{\theta}}$ are the weights that minimise the loss, typically approximated using batched stochastic gradient descent (§2.3.2),

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}) \tag{4.2}$$

When deployed for inference, the prediction is the class with the high associated probability,

$$\hat{y} = \arg\max_{y \in \mathcal{Y}} \mathtt{P}(y|\boldsymbol{x};\hat{\boldsymbol{\theta}}) \tag{4.3}$$

The classification probabilities, $\mathtt{P}(y|\boldsymbol{x};\boldsymbol{\theta})$, can be formulated based on the system architecture. In this thesis, fine-tuning will mainly be applied when training encoder-only transformers. If a pre-trained system returns text representation, $\mathbf{h}_{\mathtt{cls}} \in \mathbb{R}^d$, then this representation can be converted to an output probability distribution by introducing a classification head with weights $\mathbf{W} \in \mathbb{R}^{K \times d}$ and bias $\mathbf{b} \in \mathbb{R}^K$. The classification head is a single linear layer that provides the logits associated with each class ω_k,

$$\mathbf{h}_{\mathtt{cls}} = \text{Encode}(\boldsymbol{x};\boldsymbol{\theta}) \tag{4.4}$$

$$\mathtt{P}(\omega_k|\boldsymbol{x};\boldsymbol{\theta}) = \frac{\exp(\mathbf{w}_k^\top \mathbf{h}_{\mathtt{cls}} + b_k)}{\sum_{j=1}^{K} \exp(\mathbf{w}_j^\top \mathbf{h}_{\mathtt{cls}} + b_j)} \tag{4.5}$$

where 'Encode' denotes the representations from the foundation model. For encoder-only transformers, the text representation $\mathbf{h}_{\mathtt{cls}}$ is often the final layer vector corresponding to the first special token (which many systems refer to as the '`CLS`' token, §3.2.1). Other works have also used the outputs of other layers and representations associated with other tokens [49, 39]. For decoder-only transformers, the representation is often taken from the final layer vector associated with the token in the final position. Typically, the classification head $\mathbf{W}$ is trained jointly with all transformer parameters. One can also 'freeze' the transformer parameters and only learn $\mathbf{W}$, although typically, this results in worse in-domain performance [246]. For larger models, due to the computational expense of updating billions of parameters, various approaches have been proposed for parameter-efficient fine-tuning, such as LoRA [129] and prompt-tuning [168], which have shown to be effective and perform comparable to full model finetuning. In this thesis, we mainly fine-tune smaller encoder-only transformers,

and so unless explicitly stated otherwise, fine-tuning refers to updating all trainable model parameters and task-specific heads jointly by further supervised training.

Task Specific Setups

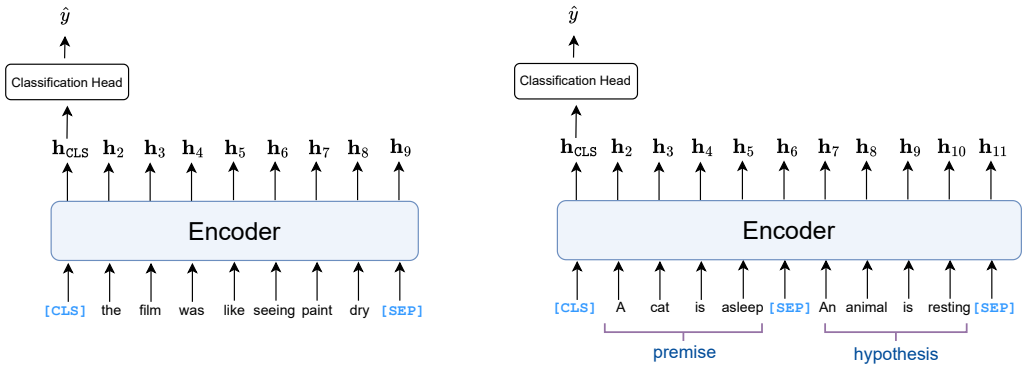


Fig. 4.2 Illustration of how BERT-style models can be adapted to classification tasks. **left:** The set-up for sentiment classification. **right:** The set-up for NLI.

The previous text described the process of fine-tuning a system for classification, where a common approach is to first encode the input $\boldsymbol{x}$ into a vector representation $\mathbf{h}_{\texttt{cls}}$ and then to train a classification head to output probabilities. The specific set-up may vary from task to task, and so the following will explore how pre-trained encoder-only transformers such as BERT [51] can be fine-tuned for the main tasks of interest (§4.1.1):

1. **Sentiment classification**: The goal of sentiment classification is to categorise a given text into the perceived sentiment, typically binary classes of positive or negative. Therefore, the fine-tuning method previously described can be directly applied. The input text is first tokenized, which includes special tokens, and the corresponding output system representation vector $\mathbf{h}_{\texttt{cls}}$ is used to predict the sentiment of the input, as illustrated in Figure 4.2.

2. **Natural language inference**: For natural language inference, the objective is to determine the relationships between two texts and determine whether the hypothesis logically follows from the premise. Therefore, the two texts are tokenized together, separated by a special token, which is provided to the transformer, generating a single representation for the pair of texts. As before, this is followed by the classification head, with the process also illustrated in Figure 4.2.

3. **Multiple-Choice Question Answering**: For certain classification tasks, the task cannot be effectively accomplished using a classification head on the representation of a single representation (e.g. the `CLS` representation). For example, in multiple-choice reading comprehension, it can be challenging for the model in a single step to understand the context $\boldsymbol{c}$ and question $\boldsymbol{q}$ and to determine which of the options $\boldsymbol{a}$ in the set of options $\boldsymbol{a}_{1:K}$ is best. To simplify the task and better leverage the pre-trained language model's capabilities, the approach can be adapted to score each option individually on its suitability [343, 268], as depicted in Figure 4.3. Here, each option $\boldsymbol{a}_k$ is encoded along with the question $\boldsymbol{q}$ and context $\boldsymbol{c}$, resulting in input representation $\mathbf{h}_{\texttt{cls},k}$,

$$\mathbf{h}_{\texttt{cls},k} = \text{Encode}(\boldsymbol{c}, \boldsymbol{q}, \boldsymbol{a}_k; \boldsymbol{\theta}) \tag{4.6}$$

Where for encoder-only transformers, $\mathbf{h}_{\texttt{cls},k}$ is similarly the final layer vector associated with the `CLS` token. Since each option is individually encoded, instead of having a classification vector for each of the K classes, $\mathbf{W} \in \mathbb{R}^{K \times d}$, each option can be scored by calculating the dot product with a single classification vector $\mathbf{w}$. This score captures how 'valid' the encoded option is and whether the option is consistent with the context and question. The scores can then be converted to the output probability distribution using the softmax:

$$\texttt{P}(\omega_k | \boldsymbol{q}, \boldsymbol{c}, \boldsymbol{a}_{1:K}; \boldsymbol{\theta}) = \frac{\exp(\mathbf{w}^\top \mathbf{h}_{\texttt{cls},k})}{\sum_{j=1}^{K} \exp(\mathbf{w}^\top \mathbf{h}_{\texttt{cls},j})} \tag{4.7}$$

Where the class $\omega_k \in \{\texttt{A}, \texttt{B}, \texttt{C}, \texttt{D}\}$ is the position of the answer, such that ω_k denotes that the correct answer is $\boldsymbol{a}_k$. The MCQA setup, beyond enabling better system capabilities, also ensures no positional bias (§6.4.4). If all options are encoded in one input, then the output representations may be sensitive to the input ordering of options. However, in this approach, since each input is encoded individually, the scores and resulting option distribution will be independent of the input ordering of the options.

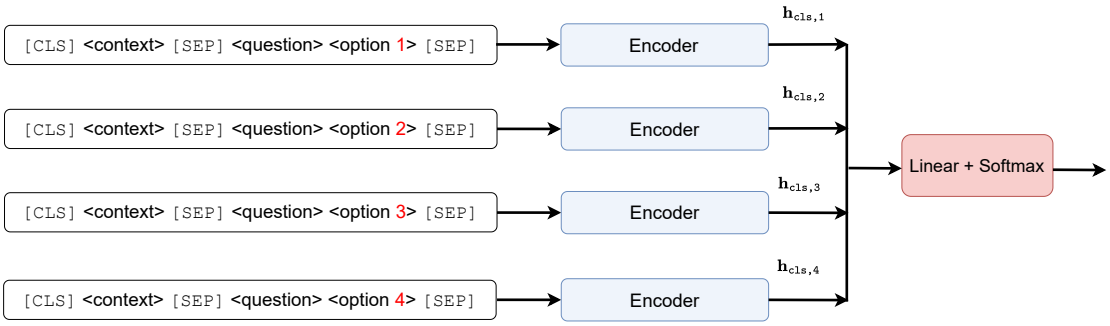


Fig. 4.3 Illustration of the set-up for multiple choice question answering

4.1.3 Zero-Shot and Few-Shot Classification

Finetuning requires task-specific labelled data and additional computational resources to adapt the pre-trained model. A more practical alternative would be to prompt instruction-following LLMs [330, 2] for classification tasks. Instruction-following LLMs are foundation models capable of responding to natural language user requests (§3.3.3) and, therefore, can be prompted for a wide range of NLP tasks. Prompting, which can be done in a zero-shot or few-shot manner, adapts the model to the task without modifying any model parameters, significantly reducing the data and computational requirements compared to fine-tuning.

Given an input sequence $\boldsymbol{x}$, generative LLMs model the token-level output probability distribution $\mathrm{P}_{lm}(z_k|z_{1:k-1},\boldsymbol{x};\boldsymbol{\theta})$ which can be used to calculate the probability associated with any output sequence $\boldsymbol{z}$, $\mathrm{P}_{lm}(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\theta})$. Instruction-following models are trained to follow natural language instructions, and therefore, inputs can be framed to the task of interest using prompt templates. For a classification task $\mathcal{T}$, let $\mathcal{P}(\boldsymbol{x})$ represent a prompt template that reframes the input within an instruction for the task at hand, e.g.,

$$\mathcal{P}(\boldsymbol{x}) = \text{`}\textit{what is the sentiment of the following review? <}\boldsymbol{x}\textit{>}\text{'} \tag{4.8}$$

Given the set of output classes, $\mathcal{Y} = \{\omega_1, \omega_2, \ldots, \omega_K\}$, one can select corresponding label words, $z_{1:k}$, which each describe their respective class, e.g. for binary sentiment classification,

$$z_0 = \textit{negative} \qquad\qquad z_1 = \textit{positive} \tag{4.9}$$

For simplicity, we use single words, although the label words can also be sequences $\boldsymbol{z}_{1:K}$. Given the design choice of the prompt and label words for the task, the prompt classifier is designed to assume that the class probabilities are proportional to the probability of the

associated class word [355, 141],

$$\mathtt{P}(\omega_k|\boldsymbol{x},\mathcal{P},z_{1:K};\boldsymbol{\theta}) = \frac{\mathtt{P}_{lm}(z_k|\mathcal{P}(\boldsymbol{x});\boldsymbol{\theta})}{\sum_{j=1}^{K}\mathtt{P}_{lm}(z_j|\mathcal{P}(\boldsymbol{x});\boldsymbol{\theta})} \tag{4.10}$$

Where the final decision $\hat{y}$ is the class with the highest probability,

$$\hat{y} = \underset{y\in\mathcal{Y}}{\arg\max}\ \mathtt{P}(y|\boldsymbol{x},\mathcal{P},z_{1:K};\boldsymbol{\theta}) \tag{4.11}$$

The prompt-based classifier setup [355, 189, 274] leverages the instruction-following capabilities of the LLM while providing a simple process to map to the decisions and associated probabilities. It's highly practical and can be used to apply instruction-following LLMs for a wide range of NLP classification tasks. In contrast, if the LLM is used to generate responses freely, the output space can be quite diverse, which may require manual intervention or an additional system to map texts to decisions. Despite the recent popularity of prompting-based methods, there is a known sensitivity of prompt-based LLMs to elements such as prompt template and label words [80, 286], with previous work [360, 297] showcasing the impact that the choice of prompt can have on task performance.

4.2 Text Quality Assessment

The second section of this chapter will focus on the task of text quality assessment. The objective of text evaluation is to assess the quality of texts for the specific attributes of interest. With the improved ability of generative AI solutions that produce high-quality, convincing texts, NLP models are increasingly utilised for generative applications. Achieving automatic and accurate evaluation of response quality is highly beneficial, aiding in system development, selection, and analysis. Human evaluation, where annotators manually assess the quality of generated texts, has been the gold standard approach [179, 18, 158, 67]. However, human evaluation has its drawbacks: it is labour-intensive, time-consuming, and costly. As such, automating the evaluation process and assessing NLG systems without human intervention is highly desirable and an active area of research [349, 203, 358, 76].

4.2.1 Applications of Text Assessment

Text quality assessment has seen significant research interest and considerable adoption. A popular use case is for natural language generation (NLG) assessment. With the improving generative abilities of current systems, these systems have been deployed for many gener-

ative applications, including summarisation systems [183, 347], chat assistance [352], and chit-chat applications [277]. Being able to assess the quality of output texts aids system development and system selection. Examples of common NLG evaluation tasks include summary assessment, dialogue assessment and data-to-text assessment, which will each be discussed in greater detail.

Summary Assessment

Summarisation condenses information from a passage into a shorter summary that retains the salient information of the original text. Given the input context $\boldsymbol{c}$ (e.g. a news article), an automatic summarisation system takes the context and generates the output summary $\boldsymbol{x}$. Being able to automatically generate summaries can be valuable for various fields, such as academia, journalism, and business, where quickly understanding large volumes of information is necessary. An established dataset for summary evaluation is SummEval [67], where the key attributes for evaluating the quality of summaries were identified as:

- **Fluency**: the readability, naturalness and continuity of the sentences.
- **Coherence**: how smoothly sentences flow and whether points are logically connected.
- **Relevance**: how well the summary captures the most important information from the source text, retaining the main focus and central themes.
- **Consistency**: whether the summary faithfully represents the information from the source document.

Dialogue Assessment

There has been significant interest in building conversational bots that interact with users using natural language text [332]. Conversational AI can be task-oriented [14], aiding users in specific tasks, or open-domain [341], engaging users in social conversations. Open-domain conversational AI is more challenging due to the broad knowledge required and the lack of a clear objective. However, the emergence of sophisticated LLMs has made engaging open-domain systems feasible [314, 139, 2]. Here, given the dialogue context $\boldsymbol{c}$ (which includes all previous turns of the conversation), the system generates a response $\boldsymbol{x}$ that continues the conversation and logically follows from the final turn of the dialogue. An established dataset for open-domain dialogue assessment is TopicalChat [215], where the responses are evaluated on the following attributes:

- **Coherence**: how connected and logically consistent the sentences in the response are.

- **Naturalness**: how human-like and conversational the responses are.
- **Continuity**: how on-topic the response is, and whether it is contextually relevant to the previous dialogue.
- **Engagement**: how interesting and enjoyable the responses are for the user.

Data-to-Text Assessment

Data-to-text generation converts structured data into natural language and supports various applications such as creating reports [362] and developing microplanners [81]. Semantic triples, which define the relationships between subjects and objects, e.g., (Einstein, winner, Nobel Prize), are a common format for representing structured information and have been widely used to represent information. Therefore, in data-to-text generation, given a set of semantic triples (which forms the context $\boldsymbol{c}$), the objective of the system is to generate output text $\boldsymbol{x}$ that contains the information present in the triples. An established dataset for evaluating data-to-text systems is WebNLG [82], where the generated text is assessed on the following attributes:

- **Grammatical correctness**: how grammatically correct the generated text is.
- **Fluency**: the natural flow and readability of the text.
- **Semantic Equivalence**: whether the generated text correctly maintains the semantic information of the input.

The three cases above illustrate several examples of useful applications of text assessment and respective important task attributes. However, a vast number of other applications exist, such as podcast summary assessment [201], story generation assessment [36], and question difficulty assessment [223]. Some attributes, such as fluency and coherency, can be considered important across many tasks, while some tasks emphasise bespoke attributes critical to their unique requirements, such as consistency for summaries. Further, some attributes within the same task may assess overlapping qualities (e.g. fluency and coherence).

4.2.2 Challenges in Text Assessment

The previous section illustrated examples of NLG assessment, an important subset of text quality assessment that has gained recent interest from the research community. Text assessment remains a popular area of research, both due to its significance in advancing language technologies and also due to the inherent challenges it presents. To illustrate

these difficulties, consider the human evaluation process. Manual assessment is typically achieved through rubric-based scoring [222], where annotators use predefined criteria to categorise texts into particular classes. Assessors undergo rigorous training to familiarise themselves with the rubric and learn to identify salient features of responses, a process that can be lengthy due to the difficulty of text evaluation. Assessing the quality of texts along a particular attribute presents considerable challenges, including:

- **Output Diversity**: Unlike classification tasks, where outputs are confined to a fixed set of classes with only one valid answer, generative tasks feature a vast space of potential outputs. This shifts the assessment from binary correctness to a more open-ended evaluation based on abstract qualities.
- **Subjectivity**: Many attributes of text evaluation involve subjective interpretations, and evaluators may bring their personal biases and preferences into their judgements. This subjectivity can lead to inconsistencies in scoring across different assessors.
- **Contextual Dependence**: The quality and appropriateness of text depends significantly on its context. Factors such as the intended audience, the purpose of the text, and domain-specific norms can heavily influence assessment criteria.

Therefore, automatic text assessment has proven to be a challenging task, with existing automatic methods lagging behind human assessments. This gap has inspired considerable research into automatic NLG assessment methods, with diverse approaches proposed across various tasks and domains aiming to achieve human performance while providing the practical benefits of automatic assessment.

4.2.3 Reference-Based Evaluation

The initial approaches to automating Natural Language Generation (NLG) evaluation relied on reference-based methods. As presented in section 4.2.1, given an input context $\boldsymbol{c}$ (which might be an article, dialogue or set of semantic triples), the NLG system can generate an output response $\boldsymbol{x}$ where the task of interest is performed (e.g. a summary or response). Assuming access to a ground-truth reference $\boldsymbol{r}$ (e.g. a human-written summary of the input article), reference-based evaluation assesses the quality of the system-generated text by measuring the degree of similarity between the text and the reference. A central question in reference-based evaluation is how to effectively compare the generated texts with the references. The typical approach involves two stages:

1. selecting a form of representation to encode both the generated text and the reference text that captures the property of interest

2. comparing the similarities of the representations using a comparison approach to produce a numerical similarity score.

As discussed in section 4.2.1, different tasks prioritise different attributes when considering quality. For example, in summarisation, the focus may be on assessing the informational content of the generated text against the reference, while for dialogue generation, the emphasis may shift toward comparing the naturalness and appropriateness of responses. Standard methods such as N-gram overlap and embedding-based similarities are used to measure similarity in a general sense (e.g. lexical or semantically), although other representations tailored to specific output attributes have also been explored.

N-gram Overlap

Initially, the most common reference-based evaluation methods were N-gram-based. These metrics aimed to measure the lexical similarity between the generated text and reference texts by comparing the overlap of word sequences (N-grams) between them. Common N-gram-based evaluation metrics include ROUGE [176], BLEU [238], and METEOR [15]. For ROUGE (Recall-Oriented Understudy for Gisting Evaluation):

1. Texts are first represented as N-grams, contiguous sequences of N words from a given text. For example, if N-gram($\boldsymbol{x}$) is used to denote all N-grams of $\boldsymbol{x}$, then
   3-gram(`the cat sat on the mat`) $= \{$ `(the, cat, sat)`, `(cat, sat, on)`,...$\}$

2. The similarity is then calculated by measuring the overlap between the generated N-grams and the reference N-grams. Let $|\cdot|$ denote the size of a set and $\cap$ represent the intersection of two sets. The The ROUGE-N score is then calculated as:

$$\text{ROUGE-N} = \frac{|\text{N-gram}(\boldsymbol{x}) \cap \text{N-gram}(\boldsymbol{r})|}{|\text{N-gram}(\boldsymbol{r})|} \tag{4.12}$$

The standard ROUGE evaluation metric is recall-based, normalising the intersection by the reference's N-gram set size. Alternately, a precision-based variant of ROUGE can also be computed by normalising with respect to the N-grams in the generated text $\boldsymbol{x}$. Due to the high variability in possible output space, small N-gram sizes are typically used, with ROUGE-1 and ROUGE-2 being very popular. This choice represents a trade-off as larger N-grams can theoretically capture more complex patterns but often suffer from sparsity issues. An alternative ROUGE-based metric, ROUGE-L [177], avoids using fixed N-gram size and considers the longest common subsequence. However, N-gram-based methods have several further limitations, including penalising semantically correct phrases that differ lexically,

not recognising distant dependencies, and inadequately penalising lexical changes that alter meaning significantly (e.g. "she opened her book quietly" vs "she burned her book quietly").

Semantic Embedding Similarity

To overcome the limitations of N-gram-based methods, embedding-based approaches instead measure the semantic similarity between the generated text and reference texts. These methods leverage text embeddings that capture deeper contextual meaning rather than relying solely on surface-level word overlaps. Embedding-based methods, such as BERTScore [349], Sentence Mover's Similarity [354], and BLEURT [293], evaluate the similarity of semantic representations, where for BERTScore:

1. Vector representations are used to encode the sentences/words obtained by leveraging the contextual representation of MLMs. Representations for both the reference and text are calculated,

$$\boldsymbol{h}_{1:L} = \text{Encode}(\boldsymbol{r}) \qquad \hat{\boldsymbol{h}}_{1:N} = \text{Encode}(\boldsymbol{x}) \tag{4.13}$$

   where $\text{Encode}(\cdot)$ represents the process of generating the final layer BERT embeddings [51] for the input text, with each vector also L2-normalised to have unit norm.

2. The semantic similarity of the embeddings is then measured by finding the closest matching token in the other text for each token in the sequence. This is calculated by using the average pairwise cosine similarity metric. BERTScore first calculates a measure of the recall and precision,

$$R_{\text{BERT}} = \frac{1}{L}\sum_{i=1}^{L}\max_{j}(\mathbf{h}_i \cdot \hat{\mathbf{h}}_j) \qquad P_{\text{BERT}} = \frac{1}{N}\sum_{j=1}^{N}\max_{i}(\mathbf{h}_i \cdot \hat{\mathbf{h}}_j) \tag{4.14}$$

   which are then combined to compute an F_1 measure [318],

$$F_{\text{BERT}} = \frac{P_{BERT} \cdot R_{BERT}}{P_{BERT} + R_{BERT}} \tag{4.15}$$

   The F_1 score is used, which assumes that both precision and recall are equally important and balances them equally. Although this can be made more general by using the F_α scores, $\alpha = 1$ is the most common operating point used in assessment.

Semantic Triple Matching

Semantic triple matching is an approach to capture and compare the factual content of texts. This method leverages the concept of semantic triples, which are the building blocks of knowledge graphs and power applications such as knowledge bases [7] and search engines [100]. Semantic triples represent facts in the (subject, relation, object) format, which stores the information in a clear, machine-readable format, enabling easy storage, retrieval and comparisons. Triples can be extracted using available automatic information extraction systems [65, 205]. Goodrich et al. [94] proposed a method of leveraging semantic triples to assess summary consistency through the following steps:

1. Information from both the reference and generated text can be represented as sets of semantic triples. Let $\text{triple}(\boldsymbol{r})$ denote the set of triples extracted from the reference text and $\text{triple}(\boldsymbol{x})$ be the set of triples from the system-generated text. The sets take the form,

$$\text{triple}(\boldsymbol{x}) = \{(\text{sub}^{(i)}, \text{rel}^{(i)}, \text{obj}^{(i)})\}_{i=1}^{M_x} = \quad \text{triple}(\boldsymbol{r}) = \{(\text{sub}^{(i)}, \text{rel}^{(i)}, \text{obj}^{(i)})\}_{i=1}^{M_r}$$

   where M_x denotes the number of triples extracted from the generated text, and M_r the number of triples extracted from the reference text.

2. Factual consistency is then evaluated by measuring the frequency that the triples in the generated set appear in the reference set,

$$\texttt{F-ACC} = \frac{|\text{triple}(\boldsymbol{x}) \cap \text{triple}(\boldsymbol{r})|}{|\text{triple}(\boldsymbol{x})|} \tag{4.16}$$

   As external knowledge sources are not used and the reference triples only cover a subset of all possible accurate facts, the generated set of triples $\text{triple}(\boldsymbol{x})$ can be further filtered to include only those facts where the subject and relationship are present in the reference's triples.

Information Consistency Assessment via Question-Answering

As an alternative, various works have explored assessing information consistency through question-answering-based assessment approaches. In this approach, automatic question-answering systems evaluate whether answers conditioned on the source text (e.g. the reference) are consistent with those conditioned on the generated text. Several span-based evaluation methods have been proposed [66, 324, 60], where extractive question-answering systems generate and answer questions based on the texts. The resulting answer spans from

the generated text and reference can be compared, with the answer similarity used as a metric for information consistency. However, using text spans can make comparing answers challenging, as exact matching methods have lexical limitations, as discussed previously with N-gram-based evaluation. A different, though similar approach is consistency assessment via multiple-choice question answering and generation (MQAG) [203], which operates as follows:

1. Let $\mathtt{P}(\boldsymbol{q},\boldsymbol{a}_{1:K}|\boldsymbol{x})$ be an automatic multiple choice question generation system which generates question $\boldsymbol{q}$ and a set of multiple-choice options $\boldsymbol{a}_{1:K}$, conditioned on text $\boldsymbol{x}$. A text's representation is then the ability to answer multiple-choice questions conditioned on said text. Given a question $\boldsymbol{q}$ and set of options $\boldsymbol{a}_{1:K}$, the resulting distributions from an automatic multiple-choice answering system (Equation 4.7), $\mathtt{P}(y|\boldsymbol{q},\boldsymbol{r},\boldsymbol{a}_{1:K};\hat{\boldsymbol{\theta}})$ and $\mathtt{P}(y|\boldsymbol{q},\boldsymbol{x},\boldsymbol{a}_{1:K};\hat{\boldsymbol{\theta}})$, are the representations of the reference and output text respectively.

2. One can then use the expected difference in the distributions as a measure of consistency. The MQAG score is the expected divergence in the distributions for questions generated from text $\boldsymbol{x}$,

$$\text{MQAG}(\boldsymbol{x},\boldsymbol{r}) = \mathbb{E}_{(\boldsymbol{q},\boldsymbol{a}_{1:K})\sim\mathtt{P}(\boldsymbol{q},\boldsymbol{a}_{1:K}|\boldsymbol{x})}[\mathtt{D}(\mathtt{P}(y|\boldsymbol{q},\boldsymbol{r},\boldsymbol{a}_{1:K}),\mathtt{P}(y|\boldsymbol{q},\boldsymbol{x},\boldsymbol{a}_{1:K}))] \tag{4.17}$$

   where $\mathtt{D}$ is a divergence metric to measure the difference between distributions. As proposed in the original paper [203], the total variation can be used as the divergence metric as it does not suffer from infinities (like the KL divergence does):

$$\mathtt{D}_{TV} = \frac{1}{2}\sum_{k=1}^{K} ||\mathtt{P}(\omega_k|\boldsymbol{q},\boldsymbol{x},\boldsymbol{a}_{1:K}) - \mathtt{P}(\omega_k|\boldsymbol{q},\boldsymbol{r},\boldsymbol{a}_{1:K})|| \tag{4.18}$$

Although this approach can be used to compare the consistency of the reference and generated text, one can also use MQAG in a reference-free manner. This is achieved by using the context passage as the source instead of the reference when comparing information consistency. This was shown to yield strong performance, and therefore when used, MQAG is typically reference-free.

4.2.4 Supervised Evaluation Methods

The previous subsection (§4.2.3) discussed reference-based evaluation, which required manual annotations and a method to compare references with the output text. However, current NLP foundation models yield token and sentence representations that can be adapted

to downstream tasks via further task-specific training (i.e. fine-tuning). Hence, given a labelled dataset, $\mathcal{D} = \{(\boldsymbol{c}^{(i)}, \boldsymbol{x}^{(i)}, s^{(i)})\}_{i=1}^{M}$ of input source context $\boldsymbol{c}$, generated output text $\boldsymbol{x}$ and associated gold-standard human evaluation score $s \in \mathbb{R}$, an alternative approach would be to directly train systems to learn the mapping of texts to the predicted assessment score. This approach will be discussed next.

Supervised Fine-tuning

Given the labelled dataset $\mathcal{D}$, one can train a system to directly learn to map the assessed output text $\boldsymbol{x}$ and context $\boldsymbol{c}$ to the numeric predicted score $\hat{s}$,

$$\hat{s} = f(\boldsymbol{x}, \boldsymbol{c}; \hat{\boldsymbol{\theta}}) \tag{4.19}$$

where f denotes the mapping function learned by the system and $\hat{\boldsymbol{\theta}}$ are the system parameters. This model can be trained to replicate the scores of the training data, minimising the mean square error (MSE) of the training data,

$$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{M} \sum_{i=1}^{M} \left(f(\boldsymbol{x}^{(i)}, \boldsymbol{c}^{(i)}; \boldsymbol{\theta}) - s^{(i)} \right)^2 \tag{4.20}$$

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}) \tag{4.21}$$

Supervised fine-tuning has been used to tailor automatic assessment methods to various bespoke tasks and domains, including for automatic assessment of conversational exams [212], assessment of spoken podcast summaries [201], and automatic essay grading [5]. Further, if references are available, the model can also take in references $\boldsymbol{r}$ when predicting the score, $\hat{s} = f(\boldsymbol{x}, \boldsymbol{c}, \boldsymbol{r}; \hat{\boldsymbol{\theta}})$, and trained with a similar criterion. This approach is leveraged by other evaluation frameworks such as COMET [272], where an estimator model is trained to predict the summary quality score given the source, output and reference, and BLUERT [293], which incorporates contextual embeddings and multiple levels of granularity to assess text quality.

Alternatively, the assessment can be treated as a classification task where score intervals are treated as separate classes, and the objective is for the model to determine the category that the output text belongs to. Each class ω_k represents a distinct, non-overlapping score range defined by an ordered set of boundary points $\{\tau_1, \tau_2, \ldots, \tau_{K+1}\}$, such that a score s belongs to the k-th class, $y = \omega_k$, if $\tau_k < s \leq \tau_{k+1}$. The boundary points span the entire range of possible scores ensuring that every possible score is assigned to exactly one class in $\mathcal{Y} = \{\omega_1, \ldots, \omega_K\}$. The system is then parameterised to model the distribution over the

different score intervals $\mathtt{P}(y|\boldsymbol{x},\boldsymbol{c};\hat{\boldsymbol{\theta}})$, and trained using the NLL loss (which is equivalent to the cross-entropy loss),

$$\mathcal{L}(\boldsymbol{\theta}) = -1 \cdot \frac{1}{M}\sum_{i=1}^{M} \log \mathtt{P}(y^{(i)}|\boldsymbol{x}^{(i)},\boldsymbol{c}^{(i)};\boldsymbol{\theta}) \tag{4.22}$$

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}) \tag{4.23}$$

supervised classification training has been used in a wide range of applications, including for assessing the coherency of real-world texts [159], essay scoring [105], and spoken summary assessment [201].

Supervised Fine-tuning using Unsupervised data

Like many supervised approaches, a drawback to supervised evaluation methods is the collection of annotated data. This is expensive as experts then have to manually score a range of texts. Alternatively, previous works have designed approaches that get the scores in an automated, unsupervised fashion. There are two main ways of achieving this:

- **Leveraging larger instruction-following LLMs**: One approach is to use larger and more capable LLMs to make zero-shot assessments (with methods discussed in the next subsection §4.2.5) and to then fine-tune smaller LLMs on the data. Gekhman et al. [86] use FLANPaLM 540B to annotate the quality of model-generated summaries, which are used to fine-tune a smaller LLM such as T5-11B for summary consistency evaluation. Kim et al. [151] prompted GPT4 to curate a dataset of reference responses and customised score rubrics, and tune a LLaMA system to evaluate responses.

- **Bespoke Synthetic Data Manipulation:** Another approach is to design methods that synthetically generate labelled data for specific attributes. For example in coherency assessment, texts can be shuffled and labelled as 'incoherent,' while the unperturbed text can be labelled as 'coherent' [16]. **UniEval** [358] employs bespoke data augmentation techniques to automatically generate positive and negative samples for selected attributes. It then converts NLG evaluation into a Boolean QA task and fine-tunes a T5 system for summary and dialogue assessment, yielding $\hat{\boldsymbol{\theta}}$. The model is trained in an instruction format, where for example to assess consistency, the input prompt $\mathcal{P}$ takes

form,

$$\mathcal{P}(\boldsymbol{x},\boldsymbol{c}) = \textit{`Is the claim consistent with the document?}$$
$$\textit{claim: <}\boldsymbol{x}\textit{>}$$
$$\textit{document: <}\boldsymbol{c}\textit{>'}$$

The positive and negative classes are mapped to `Yes` and `No`, and the system is trained in a supervised fashion. Following prompt classifiers, the output is the relative probability of generating `Yes` against `No`, and the probability is used as the output score,

$$\text{UniEval}(\boldsymbol{x}|\boldsymbol{c}) = \frac{\texttt{P}(\texttt{Yes}|\mathcal{P}(\boldsymbol{x},\boldsymbol{c});\hat{\boldsymbol{\theta}})}{\texttt{P}(\texttt{Yes}|\mathcal{P}(\boldsymbol{x},\boldsymbol{c});\hat{\boldsymbol{\theta}}) + \texttt{P}(\texttt{No}|\mathcal{P}(\boldsymbol{x},\boldsymbol{c});\hat{\boldsymbol{\theta}})} \tag{4.24}$$

Some attributes use references for evaluation which are embedded in the prompt, and the authors also explored multi-task training and continual learning.

4.2.5 Zero-shot Reference-Free Evaluation

The previous methods discussed both reference-based evaluation and supervised evaluation methods. With the recent emergent abilities of LLMs, a recent trend is to leverage the abilities of LLMs for zero-shot and reference-free evaluation, which enable generalised and inexpensive evaluation. Such approaches rely on the instruction-following abilities of current NLP foundation models to provide assessment scores, with the zero-shot approaches broadly categorised as probability-based or scoring-based [174].

Generative Probability Evaluation

The quality of generated texts may be expected to correlate with the ease of generation for capable LLMs. Approaches have therefore proposed to leverage LLM's generative probabilities for assessment. The first approach, BARTScore [344], used BART [171] to compute the probabilities of the generated text and demonstrates that these probabilities correlate with assessment scores for multiple tasks such as machine translation, text summarisation, and data-to-text tasks. This was extended by **GPTScore** [76], which leverages the zero-shot instructional capabilities of instruction-following LLMs. GPTScore calculates the associated LLM probability of generating the evaluated text based on a set of conditions and instructions.

E.g. for summarisation, one may have the instruction,

$$\mathcal{P}(\boldsymbol{x},\boldsymbol{c}) = \text{`}\textit{Generate a fluent and grammatical summary for the following text: }\textbf{<}\boldsymbol{c}\textbf{>}\textit{ Tl;dr }\textbf{<}\boldsymbol{x}\textbf{>}\text{'}$$

and make the assumption that a good summary will have a larger associated likelihood. Therefore given task prompt $\mathcal{P}$ for the particular attribute, the GPTScore of generated text $\boldsymbol{x} = x_{1:N}$ given input context $\boldsymbol{c}$ can be calculated as,

$$\text{GPTScore}(\boldsymbol{x}|\boldsymbol{c},\mathcal{P}) = \sum_{k=1}^{N} \log \mathrm{P}_{lm}(x_k|\mathcal{P}(x_{1:k-1},\boldsymbol{c});\boldsymbol{\theta}) \tag{4.25}$$

Further, to enhance accuracy, GPTScore can incorporate demonstration samples. These are examples of text that meet the evaluation criteria and serve to guide the model's generation and assessment processes through in-context learning.

Scoring-based Evaluation

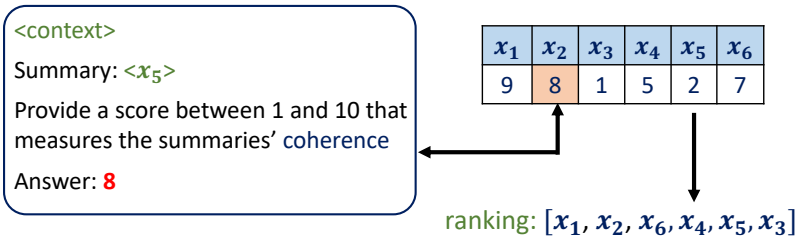


Fig. 4.4 Depiction of Prompt-Scoring: For each candidate text the LLM is asked to assess the text within a predefined scale.

For scoring-based approaches [326, 182], the LLM is prompted to directly predict the quality score of the text, as illustrated in Figure 4.4. This uses the classification set-up introduced in section 4.1.3, where given a prompt $\mathcal{P}$ and label words $w_{1:K}$, the model is asked to classify the input text based on its quality. Here, the prompt $\mathcal{P}$ is designed to request the LLM to assess the quality of a text with a score (e.g. between 1 and 10),

$$\mathcal{P}(\boldsymbol{x},\boldsymbol{c}) = \text{`}\textit{Provide a score between 1 and 10 that measures the text's coherency.}\\ \textit{Context: }\textbf{<}\boldsymbol{c}\textbf{>}\\ \textit{Text: }\textbf{<}\boldsymbol{x}\textbf{>}\text{'}$$

In this setting, the label words are set to integers up to the maximum score, e.g. $z_k \in \{1,2,...,10\}$. Two variants of absolute assessment can be applied; the first approach requests the model to directly predict the score, selecting the decision as the score with the highest mass,

$$\hat{s} = \arg\max_k \mathtt{P}_{lm}(z_k|\mathcal{P}(\boldsymbol{x},\boldsymbol{c});\boldsymbol{\theta}) \tag{4.26}$$

Which is therefore also applicable even in black-box settings. Alternatively, following G-Eval [182], one can estimate the expected score through a fair average by multiplying each score by its normalised probability,

$$\mathtt{P}(\omega_k|\boldsymbol{x},\boldsymbol{c},\mathcal{P};\boldsymbol{\theta}) = \frac{\mathtt{P}_{lm}(z_k|\mathcal{P}(\boldsymbol{x},\boldsymbol{c});\boldsymbol{\theta})}{\sum_{i=1}^{K}\mathtt{P}_{lm}(z_k|\mathcal{P}(\boldsymbol{x},\boldsymbol{c});\boldsymbol{\theta})} \tag{4.27}$$

$$\hat{s} = \sum_{k=1}^{K} k \cdot \mathtt{P}(\omega_k|\boldsymbol{x},\boldsymbol{c},\mathcal{P};\boldsymbol{\theta}) \tag{4.28}$$

where ω_k represents an assessment score of k.

4.2.6 Evaluation of Approaches

To evaluate the effectiveness of automatic assessment scores for a given task, typical approaches involve comparing the correlations of the predicted scores with ground-truth human annotations. Consider the dataset $\mathcal{D}$, which comprises a set of unique input contexts $\boldsymbol{c}$, a set of N different system-generated responses $\boldsymbol{x}_{1:N}$, and corresponding manually annotated assessment scores $s_{1:N}$,

$$\mathcal{D} = \left\{ \left(\boldsymbol{c}^{(i)}, \boldsymbol{x}_{1:N}^{(i)}, s_{1:N}^{(i)} \right) \right\}_{i=1}^{M} \tag{4.29}$$

where M is the number of unique contexts, and N is the number of different generative systems. Given an assessment method f that generates predicted assessment scores $\hat{s}_k^{(i)}$ for each generated text,

$$\hat{s}_k^{(i)} = f(\boldsymbol{x}_k^{(i)}, \boldsymbol{c}^{(i)}) \tag{4.30}$$

evaluation can be performed by measuring how correlated the predicted scores are to the ground-truth human annotated scores. Two types of evaluation are commonly used as evaluation metrics to evaluate the performance of automatic assessment methods: response-level correlations and system-level correlations.

Response-level Correlations

Response-level correlations measure how well the automatic scores rank responses at the context level. For each context, given a set of responses, this metric evaluates whether the predicted set of scores correlated to the ground-truth scores, averaged over all contexts:

$$\rho = \frac{1}{M}\sum_{i=1}^{M} \text{Corr}\left(\hat{s}_{1:N}^{(i)}, s_{1:N}^{(i)}\right) \tag{4.31}$$

System-level Correlations

System-level correlations measure how well an assessment method ranks different systems. Let the system score be the scores averaged over all generated responses from a particular system, such that S_k denotes the ground-truth score for the k-th system, and $\hat{S}_k$ the predicted score for the k-th system, defined as:

$$\hat{S}_k = \frac{1}{M}\sum_{i=1}^{M} \hat{s}_k^{(i)} \qquad\qquad S_k = \frac{1}{M}\sum_{i=1}^{M} s_k^{(i)} \tag{4.32}$$

The system-level correlations are the correlations between the averaged predicted scores and ground-truth scores:

$$\rho = \text{Corr}\left(\hat{S}_{1:N}, S_{1:N}\right) \tag{4.33}$$

This results in less noisy estimates, albeit making the overall task less challenging.

Measuring Correlation

In the NLG evaluation methodologies, two primary objectives are often considered, scoring and ranking. These objectives use two different metrics, the Pearson correlation coefficient (PCC) [244] and the Spearman Correlation Coefficient (SCC) [299]:

- **Pearson Correlation Coefficient (PCC)**: The PCC is used to assess the scoring abilities of a system. Here the objective is to assign independent scores that reflect the quality of the text, where score differences are indicative of quality differences. This metric is applied in scenarios where it's useful to know the degree to which a text is better than another. The PCC measures the linear correlations between the predictions and ground-truth scores, defined as,

$$\text{PCC}(\hat{s}_{1:N}, s_{1:N}) = \frac{\sum_{i=1}^{N}(\hat{s}_i - \bar{\hat{s}})(s_i - \bar{s})}{\sqrt{\sum_{i=1}^{N}(\hat{s}_i - \bar{\hat{s}})^2 \sum (s_i - \bar{s})^2}} \tag{4.34}$$

Note that, unlike exact matching metrics such as RMSE, linear correlation does not require the predictions and references to share the same scale but inherently adjusts for scale differences.

- **Spearman Correlation Coefficient (SCC)** The SCC is used to assess the ranking abilities of a system. In this context, the focus is on identifying which text is better than another and not the magnitude of the difference. The SCC measures the strength and direction of the monotonic relationship between two ranked variables. Given the predicted ranks $\hat{r}_{1:N}$ and true ranks $r_{1:N}$, the Spearman's correlation ρ is defined as,

$$\text{SCC}(\hat{r}_{1:N}, r_{1:N}) = 1 - \frac{6 \cdot \sum_{i=1}^{N} (\hat{r}_i - r_i)^2}{N(N^2 - 1)} \tag{4.35}$$

- **Root Mean Square Error** Another commonly used metric to measure the similarity of the predicted scores to the ground-truth scores is the root mean square error (RMSE):

$$\text{RMSE}(\hat{s}_{1:N}, s_{1:N}) = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (\hat{s}_i - s_i)^2} \tag{4.36}$$

where $\hat{s}_i$ and s_i represent the predicted score and the true score, respectively. The RMSE measures the square root of the average distance between the predictions and labels, disproportionately penalising large deviations. While useful for quantifying prediction errors, the metric is sensitive to the selected scoring scale, which makes comparing scores using different score ranges challenging. Therefore, within NLG, where there are arbitrary and often diverging scoring scales, scale-invariant metrics like Pearson and Spearman correlations are generally preferred.

4.3 Comparative Assessment

The previous section introduced various automatic scoring methods for NLG assessment, where these systems returned independent scores for each text, mirroring traditional rubrics-based assessment [222]. In this form of assessment, assessors assign numeric quality scores based on how well a text demonstrates the examined attribute. However, this assessment method is considered quite challenging, often requiring intensive training for scorers to become familiar with scoring scales [301].

Comparative judgement [310] offers an alternative assessment method, where instead of individually scoring items, assessors compare items and make decisions about their relative qualities (e.g. whether item A is better than item B). This can be viewed as a simpler

form of assessment, as it directly compares two similar items instead of relating them to an internalised standard [89]. Although not widely explored for automatic text evaluation, comparative judgement has gained considerable popularity in manual assessment [252] and other domains such as ranking teams/players in games [64]. This section provides the background for comparative assessment, which supports Chapter 7, where we will introduce LLM-based comparative assessment.

4.3.1 The Thurstonian Model

In 1927, Louis. L. Thurstone introduced the law of comparative judgement [310], which proposed a theoretical framework for understanding how individuals perceive and compare stimuli. This approach focused on pairwise comparisons rather than absolute judgements and provided a mathematical model for converting these comparisons into a linear scale of measurement. Since its first application within psychology, it has since been extended, adopted and studied in various fields and contexts, including sports [17, 48], information retrieval [32, 181] and social studies [206, 195]. It has further been applied to text assessment [253, 252], which will be the context used for the following description.

Thurstone's key insight was that human judgements are inherently probabilistic, subject to variability both within and between individuals. The Thurstonian model (also referred to as the Thurstone-Mosteller model) assumes that within a comparison, the perceived strength of item $\boldsymbol{x}_i$ is normally distributed around its skill s_i. Therefore, the probability that $\boldsymbol{x}_i$ is preferred over $\boldsymbol{x}_j$ is given as,

$$\mathtt{P}(\boldsymbol{x}_i \succ \boldsymbol{x}_j) = \Phi\left(\frac{s_i - s_j}{\sqrt{\sigma_i^2 + \sigma_j^2 - 2\sigma_{ij}^2}}\right) \tag{4.37}$$

Where $\boldsymbol{x}_i \succ \boldsymbol{x}_j$ represents $\boldsymbol{x}_i$ being preferred over $\boldsymbol{x}_j$, with this preference depending on the specific attribute being assessed in the given task. Φ is the cumulative normal distribution, σ_i and σ_j are the standard deviations of skills for items $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ respectively, and σ_{ij} is their covariance. A common simplification is Thurstone's Case V model, which assumes that the perceived values are independent,$\sigma_{ij} = 0$, and that the variance is constant across all inputs, $\sigma_i^2 = \sigma_j^2 = 0.5$. For this model, the probability can be simplified to,

$$\mathtt{P}(\boldsymbol{x}_i \succ \boldsymbol{x}_j) = \Phi(s_i - s_j) \tag{4.38}$$

However, the skills $s_{1:N}$ are unknown and have to be inferred from the outcomes of comparisons between items. Let $\mathcal{C}_{1:K}$ denote a set of K binary comparisons between the items, where

each comparison C_k is of the form $(\{i,j\}, y_{ij})$, where $y_{ij} \in \{0,1\}$ denotes the outcome of whether $\boldsymbol{x}_i$ won against $\boldsymbol{x}_j$. Given a hypothesised set of scores $s_{1:N}$, the associated likelihood of the observations take the form,

$$\mathtt{P}(\mathcal{C}_{1:K}|s_{1:N}) = \prod_{\{i,j\},y_{ij}\in\mathcal{C}_{1:K}} \Phi(s_i - s_j)^{y_{ij}}(1-\Phi(s_i-s_j))^{1-y_{ij}} \tag{4.39}$$

The predicted scores $\hat{s}_{1:N}$ can then be defined as the maximum likelihood solution,

$$\hat{s}_{1:N} = \arg\max_{s_{1:N}} \mathtt{P}(\mathcal{C}_{1:K}|s_{1:N}) \tag{4.40}$$

Although the solution has no analytical form, it can be solved using numerical iterative optimisation techniques. The Thurstonian model provided an important foundation for research into comparative judgement and one of its most popular extensions, and arguably the most widely applied method to analyse comparative judgements, is the Bradley-Terry model, which will be discussed next.

4.3.2 The Bradley Terry Model

The Bradley-Terry (BT) [27] model shares many similarities with the previous Thurstonian model, however instead of assuming that the observed values are normally distributed, it adopts a logistic distribution model,

$$\mathtt{P}(\boldsymbol{x}_i \succ \boldsymbol{x}_j) = \sigma(s_i - s_j) = \frac{\exp(s_i - s_j)}{1+\exp(s_i-s_j))} \tag{4.41}$$

where $\sigma(s)$ is the sigmoid function, $\sigma(s) = 1/(1+e^{-s})$. The model similarly treats the scores as parameters of the model and aims to maximise the likelihood of the observations,

$$\hat{s}_{1:N} = \arg\max_{s_{1:N}} \mathtt{P}(\mathcal{C}_{1:K}|s_{1:N}) = \arg\max_{s_{1:N}} \prod_{\{i,j\}\in\mathcal{C}_{1:K}} \sigma(s_i - s_j)^{y_{ij}}(1-\sigma(s_i-s_j))^{1-y_{ij}} \tag{4.42}$$

While the Thurstonian and Bradley-Terry models are known to yield nearly identical solutions, the latter has gained wider adoption due to its more tractable maximum likelihood estimation (MLE) formulation and more extensive mathematical analysis [106]. Although no closed-form solution exists for the Bradley-Terry model, Zermelo's algorithm [346] provides an iterative approach to convergence.

For Zermelo's algorithm, let $\pi_i = \exp(s_i)$ and n_{ij} the number of times that item i beat item j. The algorithm starts by initialising all $\pi_i^{(0)}$ to some positive value (e.g. 1), and the

values of π_i are then iteratively updated following:

$$\pi_i^{(k+1)} = \frac{\sum_{j=1}^{N} n_{ij}}{\sum_{\substack{j=1 \\ j\neq i}}^{N} \frac{n_{ij}+n_{ji}}{\pi_i^{(k)}+\pi_j^{(k)}}} \tag{4.43}$$

After each iteration, the π_i values are normalised by dividing by their geometric mean. This process is repeated until convergence, i.e. until the maximum change in any π_i between iterations is below a selected threshold ε. Despite the slow convergence rates of the original algorithm [61, 131], recent mathematical advances have improved the efficiency and led to significantly faster convergence rates [228].

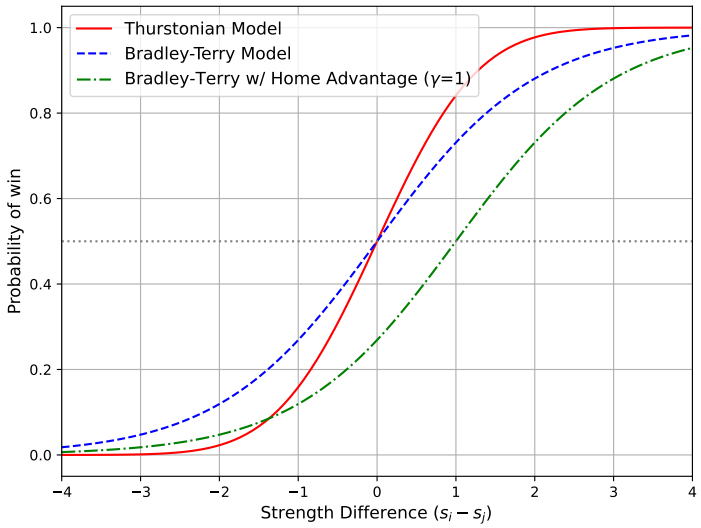


Fig. 4.5 Illustration of the different modelling assumptions of various comparative assessment models, including the Bradley-Terry model with home advantage. The home advantage parameter γ is arbitrarily set to 1.

Extensions to Bradley Terry

The Bradley-Terry model has been extended to accommodate various additional factors. One example is the home-advantage extension, which is particularly relevant in sports contexts where the 'home' team may have an advantage over the away team [71]. This information

can be incorporated into the model by introducing a parameter γ that shifts the probabilities:

$$\mathrm{P}(\boldsymbol{x}_i \succ \boldsymbol{x}_j) = \sigma(s_i - s_j + \gamma) = \frac{\exp(s_i - s_j + \gamma)}{1 + \exp(s_i - s_j + \gamma)} \tag{4.44}$$

Where the parameter γ quantifies the magnitude of home advantage expected in the comparisons, either estimated from data or set based on domain knowledge. The comparison between the Thurstonian model, The Bradley-Terry model and the Bradley-Terry model with home advantage is illustrated in Figure 4.5.

There are numerous other extensions of Bradley-Terry, including the Bradley-Terry-Luce model [197], which accommodates comparisons with more than two items, or TrueSkill [114, 219], which incorporates uncertainties in player skills (in a sports context) under a Bayesian framework.

4.4 Chapter Summary

This chapter considered various applications of applying NLP foundation models. The first section focused on automatic text classification and detailed various methodologies to leverage NLP foundation models for various classification tasks. The next section considered text quality assessment, discussing various general approaches adopted by the field, including reference-based evaluation, bespoke supervised task training, and emerging zero-shot LLM prompting techniques. Finally, the law of comparative judgement was introduced, discussing an alternative of using pairwise comparisons for text assessment, hinting at the potential application of LLMs within this framework.

Chapter 5

Spurious Correlations in NLP

This Chapter focuses on the pre-train and fine-tune paradigm and examines the risk of systems learning spurious correlations when trained in a supervised fashion on labelled data. The term *spurious* refers to the scenario where something appears to be something which it is not. For example, a spurious correlation occurs when a variable appears to exhibit a high correlation with the target variable within a localized domain, but where this correlation does not actually reflect a meaningful association and is instead due to chance, confounding factors, or limitations in the data.

Language is a highly complex form of data. Although traditional rule-based approaches struggle to model challenging NLP tasks, with developments in deep learning where networks with millions or billions of parameters are trained using empirical risk minimisation [319], NLP solutions have rapidly improved. Deep learning systems jointly perform feature engineering, where relevant abstract task information is captured in vector representations, as well as the mapping of these features to output decisions [312]. However, in NLP tasks, the set of possible features is vast. Therefore, there is the risk that the system extracts features from labelled data that are *spurious* [85]. These are features that may capture domain-specific or annotation-specific correlations, and though they may yield good performance in the training domain, they do not generalize to other domains of the task. This will impact the robustness of these systems and cause systems deployed in real-world scenarios to make unreliable decisions.

This Chapter discusses the risk of spurious features in NLP, features that appear to be relevant to the task but are not, and extends this idea to *spurious questions*, questions that appear to assess a task but don't. Section 5.1 describes how relationships between variables can be categorized as either causal, correlated or spuriously correlated and discusses why it is challenging to determine spurious features for NLP tasks. Section 5.2 discusses methodologies for identifying spurious token features that NLP systems may be susceptible

to leveraging. Section 5.3 proposes spurious stopword features, a new type of spurious feature to analyse. Section 5.4 extends the discussion of spurious features to spurious questions and proposes a methodology for identifying questions that candidates may solve while bypassing the intended comprehension task. Finally, section 5.5 concludes the chapter by presenting experimental results and discussing the findings.

5.1 Spurious Correlations in NLP

5.1.1 Causation, Correlation and Spurious Correlations

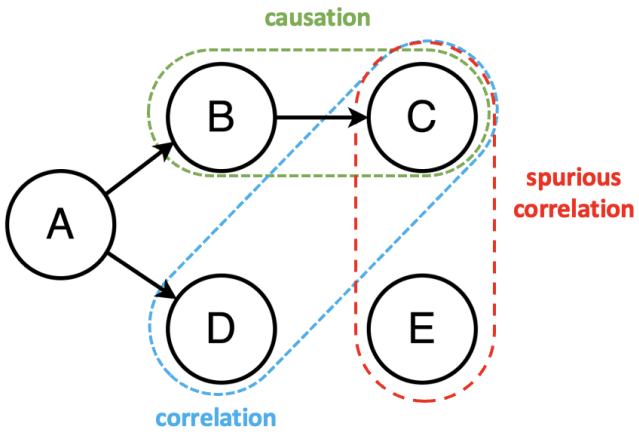


Fig. 5.1 Depiction of the various relationships that variables can have: causation (B $\rightarrow$ C), correlation (D, C) and spurious correlation (E, C).

In this chapter, we categorise relationships between variables as one of three different types: *causal*, *correlated* and *spuriously correlated*. Figure 5.1 illustrates the different relationships of variables using a causal graph. The nodes represent variables, while the edges represent a simple causal relationship between two variables in the direction of the arrow.

- A *causal relationship* [243] is one where a change in one variable directly influences the observed value of another variable. This relationship goes beyond mere association and implies that there is an underlying process where one variable influences the other one. $B \rightarrow C$ denotes that B has a causal influence on C. For example, B may represent whether an individual smokes and C represents whether an individual has lung cancer, with the graph therefore implying that starting to smoke will directly influence the chance that an individual will contract lung cancer.

- A correlation [75, 23] describes the scenario where the two variables share a statistical relationship, such that knowing one variable provides information on the other. This occurs because the variables share a common cause that influences both, as shown with C and D in the diagram. For example, if A is socioeconomic background, D is income, B represents smoking, and C is getting lung cancer, then one might find that income *correlates* with lung cancer rates. This will be purely a statistical relationship and not be causal; an individual who gets a raise in income will not be any more/less likely to contract lung cancer. However, such correlations can provide useful information when making classification decisions, and by taking many aggregated correlations for the variable for interest, a model may have good predictive classification performance.

- A *spurious correlation* [85] denotes when two variables have no true statistical relationship to each other. Although they may appear to have a relationship within a particular niche domain or when considering a limited number of samples, this relationship does not hold in general out-of-training domains of the task. Relying on spurious correlations has limitations when the system is used outside the training domain, where the spurious correlation does not hold. For example, imagine that in a particular town, all members of the 'Smith' family smoke. Therefore, E, whether the individual's name begins with 'S', may seem to correlate with C, the lung cancer rates. However, this is only within a localised domain and does not hold for the general task, and the correlation is clearly spurious and dangerous to base decisions on.

An idealised system would capture the information from all features which have a causal relationship with the target. Although unrealistic, such a system would perfectly model the task and, therefore, generalise across all possible domains. Alternatively, a classifier that bases decisions on genuine task *correlations* can still be valuable and effective. These correlations can yield strong predictive information for the task, which in many domains may yield the correct decision, although may fail in edge cases and/or to adversarial perturbations. A system that relies on spurious correlations, however, is clearly disadvantageous. Therefore, in this chapter, the aim is to have classifiers that use either causal relationships or valid correlations to make decisions and avoid relying on spurious correlations.

Note that within the causal inference framework [242, 243], the objective may be to capture only causal relationships rather than all correlations. In the framework, spurious correlations are defined differently, and variables that share a common cause but are not causally related are also categorised as a spurious correlation (e.g. the relationship between B and D in Figure 5.1). This stricter criteria for determining relevant features is important in fields such as

healthcare, where the objective may be to identify whether a prescribed treatment causally affects a patient's health outcome [283, 296]. However, to accommodate for the NLP area, which has traditionally been very data-driven, we instead adopt a less strict requirement for relevant features, and instead, if features share a common confounder in *most* domains, we consider this a valid task correlation. Spurious correlations are defined when the relationship only holds in a single domain but not for the general task (which will be discussed in greater detail in section 5.1.3).

5.1.2 Challenges of Spurious Correlations for NLP tasks

Identifying all important intermediate features and their causal relationships can enable tasks to be modelled causally. This is achieved by setting up a structural graphical model [243] with all causal relationships represented on a directed acyclic graph, which is then used to decompose the classification probabilities for any new input. However, text is a rich form of data which has many associated challenges that make causal inference challenging. In particular, texts have the following properties which make defining the causal graph challenging:

1. **Text is a complex data form**. Text has a hierarchical structure that spans characters, words, sentences and passages [144]. Information is conveyed at various levels of granularity, with complex syntax [40], which makes parsing information non-trivial. This makes textual information complex, nuanced, and challenging to convert into salient features.

2. **Text has high contextual sensitivity**. Due to the interplay of linguistic elements, the meaning of sentences can vary with small changes to words or the ordering of the words (e.g. "give food to her cat" vs. “give cat food to her") [209]. Furthermore, words may have multiple meanings (polysemy) and ambiguities [150, 204], which may need to be disambiguated based on context or world knowledge.

3. **The number of possible features is vast**. Text can be used to represent objects, actions, abstract ideas, a multitude of other things and a combination of them all [204]. There are a vast number of different properties or underlying ideas that could be represented within a text. Therefore, the set of all possible relevant features for a task is extensive.

4. **Features can be difficult to identify**. Text features may depend on the complex interactions of many elements within the text [144, 204]. Features can be expressed in many different ways, and determining whether the text contains a particular feature may be challenging and, in some cases, subjective.

Due to these challenges, within NLP, it is often infeasible to identify the set of core features and construct a causal graph. There has been a growing body of work in NLP that uses elements of causal formalisms to improve the robustness of predictions [148, 210, 70]. For example, existing work has curated contrastive sets, where subtle modifications are made to examples that change the output class [148, 83, 210]. Training on these contrastive sets can yield improved robustness, and these sets can also be used as test sets to analyse robustness [210, 351, 148]. Further NLP has been applied for causal discovery and estimating causal effects [149, 70].

However, for the rest of this chapter, we will focus on analysing the process of fine-tuning NLP models on standard NLP datasets. In this setup, labelled training data is used to learn the task end to end, including the discovery of relevant intermediate features and the mapping of features to the output classes. This can make the systems susceptible to learning spurious correlations, which will be discussed next.

5.1.3 Spurious Features

Previously, it was described how spurious correlations occur when false conclusions are drawn between relationships of input features with the target variable. Here, we use the term *feature* to refer to any measurable property of the input that might inform the classification decision. For instance, a feature could be binary (e.g., whether the text contains the word "cat"), numeric (e.g., text length), or more abstract (e.g., the hidden-layer representations of a deep neural network). Let the full set of possible features be partitioned into the set of core features which are meaningful for the task of interest, $\mathcal{F}_c$, and the set of irrelevant features which are not meaningful for the task, $\mathcal{F}_i$. For NLP tasks, due to the high-dimensional space of text information, it is infeasible to define the complete set of relevant/irrelevant features. These feature sets contain a vast number of possible features, and further, in many cases, the features themselves may not exist in a human-understandable form. This is a limitation when considering spurious correlation for NLP tasks, and hence, $\mathcal{F}_c$ and $\mathcal{F}_i$ represent the hypothetical spaces of all features that are relevant to the task or not, respectively, with their definitions kept intentionally broad and conceptual.

For a task $\mathcal{T}$, assume there exists a single true underlying task conditional distribution $\mathtt{P}(y|\boldsymbol{x})$. Let ε represent a particular domain of the task, which defines the distribution that the input samples were drawn from, $\mathtt{P}_\varepsilon(\boldsymbol{x})$. A dataset $\mathcal{D}$ drawn from domain ε_{tr} can be curated by sampling inputs from the domain and the target variable y from the true output distribution

(assuming human annotators approximate this distribution),

$$\mathcal{D} = \{(\boldsymbol{x}^{(i)}, y^{(i)})\}_{i=1}^{M} \qquad \boldsymbol{x}^{(i)} \sim \mathtt{P}_{\varepsilon_{\text{tr}}}(\boldsymbol{x}), \quad y^{(i)} \sim \mathtt{P}(y|\boldsymbol{x}^{(i)}) \tag{5.1}$$

A system, parametrised by $\boldsymbol{\theta}$, models the conditional distribution of the target classes y given the input $\boldsymbol{x}$, $\mathtt{P}(y|\boldsymbol{x};\boldsymbol{\theta})$, aiming to be similar to the true distribution $\mathtt{P}(y|\boldsymbol{x})$. This is achieved by minimising the loss function $\mathcal{L}(\boldsymbol{\theta})$ over the training data, often the negative log-likelihood (NLL) loss:

$$\mathcal{L}(\boldsymbol{\theta}) = -1 \cdot \frac{1}{M} \cdot \sum_{i=1}^{M} \log \mathtt{P}(y^{(i)}|\boldsymbol{x}^{(i)};\boldsymbol{\theta}) \tag{5.2}$$

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}) \tag{5.3}$$

The DNN system is a fully parameterised system that has direct connections from the input to the output. The process is often conceptually decomposed into two stages [312]:

1. **Feature compression**, where the system has to compress the input into a compact feature representation. In this stage, the model has to determine the relevant and irrelevant input features and process the input to maintain the important task information. Let $f_{\text{h}}(\cdot)$ denote a feature extraction function that processes the input to extract a feature or set of features,

   $$\mathbf{h} = f_{\text{h}}(\boldsymbol{x};\hat{\boldsymbol{\theta}}) \tag{5.4}$$

   The system aims to extract features that lie with the set of core features, such that $\mathbf{h} \in \mathcal{F}_c$. For DNN systems, the feature extraction function $f_{\text{h}}(\cdot)$ is generally the final layer vector representation of the text, $\mathbf{h} \in \mathbb{R}^d$.

2. **Classification** For deep learning systems, once the input has been processed to the representation $\mathbf{h}$ (which implicitly captures all relevant task information), the next stage is to convert the representation to the classification decision. For models, this is typically done using a final linear layer, followed by a softmax to convert the representation to a probability distribution, $\mathtt{P}(y|\boldsymbol{x};\hat{\boldsymbol{\theta}})$. Within our setup, this can be interpreted as making a decision conditioned on the derived feature,

   $$\mathtt{P}(\omega_k|\boldsymbol{x};\hat{\boldsymbol{\theta}}) = \mathtt{P}(\omega_k|f_{\text{h}}(\boldsymbol{x};\hat{\boldsymbol{\theta}})) = \text{softmax}(\mathbf{w}_k^{\mathsf{T}}\mathbf{h}) \tag{5.5}$$

   where $\mathbf{w}_k$ are the classification head weights associated with the kth class.

Spurious correlations, also known as shortcuts [85], occur when the system learns non-robust features from limited training data. This occurs when the features appear informative within a

particular domain, possibly due to annotation artefacts [102], though where they are actually not related to the task of interest and hence are within the set of irrelevant features, $\mathbf{h} \in \mathcal{F}_i$. When spurious features are used, although performance may be reasonable within the training domain, performance will be limited when applied to samples from outside the training domain,

$$\mathtt{P}(y|f_{\mathrm{h}}(\boldsymbol{x};\hat{\boldsymbol{\theta}})) \approx \mathtt{P}(y|\boldsymbol{x}) \qquad \boldsymbol{x} \sim \mathtt{P}_{\varepsilon_{\mathrm{tr}}}(\boldsymbol{x}) \tag{5.6}$$

$$\mathtt{P}(y|f_{\mathrm{h}}(\boldsymbol{x};\hat{\boldsymbol{\theta}})) \not\approx \mathtt{P}(y|\boldsymbol{x}) \qquad \boldsymbol{x} \sim \mathtt{P}_{\varepsilon_{\mathrm{out}}}(\boldsymbol{x}) \tag{5.7}$$

where $\mathtt{P}_{\varepsilon_{\mathrm{out}}}(\boldsymbol{x})$ represents an out-of-domain distribution of examples. Therefore, leveraging spurious correlations will only result in superficial task capabilities but will not enable the system to generalise well and be robust when the system is deployed in general domains of the task.

5.2 Spurious Tokens

The previous section described the process of training deep learning systems and the risks of relying on spurious features, such as poor generalisation. The mapping of input to representation, $\mathbf{h} = f_{\mathrm{h}}(\boldsymbol{x};\hat{\boldsymbol{\theta}})$, learned during training can be interpreted as the system selecting important task features. Due to the vast number of features in NLP tasks (§5.1.2), it is not feasible to identify the exact features that a model uses nor determine whether the features are spurious or not. Therefore, a common methodology for analysing spurious correlations in NLP is to only consider a restricted set of features, such as the presence of individual tokens. The rest of this section will discuss existing methodologies for establishing whether individual tokens exhibit spurious correlations within NLP datasets and the impact this may have on trained systems.

5.2.1 Injecting Synthetic Spurious Tokens

A simple approach to assess how vulnerable NLP systems are to learning spurious correlations is to introduce an artificial spurious correlation into a training dataset. This can be achieved by injecting a specific token (e.g. a punctuation symbol) into samples of a particular class [306, 143, 56], such that within the training data, relying on this shortcut alone provides high information on the output class. For example, let $\mathcal{D} = \{\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)}\}_{i=1}^{M}$ represent a labelled dataset for a particular task. Given classes $\mathcal{Y} = \{\omega_1, \omega_2, \ldots, \omega_K\}$, a spurious token v_s can be associated with class ω_k by adding $v_s \in \mathcal{V}$ to the end of each input of the selected class,

creating the biased dataset $\mathcal{D}_b$,

$$\tilde{\boldsymbol{x}}^{(i)} = \begin{cases} \boldsymbol{x}^{(i)} \oplus v_s & \text{if } y^{(i)} = \omega_k \\ \boldsymbol{x}^{(i)} & \text{otherwise} \end{cases} \tag{5.8}$$

$$\mathcal{D}_b = \{(\tilde{\boldsymbol{x}}^{(i)}, \boldsymbol{y}^{(i)})\}_{i=1}^{M} \tag{5.9}$$

where $\oplus$ denotes the text concatenation operation. One can also randomly select the position where the token is added or make the process probabilistic by only randomly injecting the token with a probability p. One can then analyse the degree to which systems trained on the dataset with artificial spurious associations, $\mathcal{D}_b$, exhibit the injected spurious correlation.

However, randomly injecting tokens may yield unnatural and unrealistic texts. An alternative would be to filter datasets to exhibit spurious correlations. For example, in sentiment classification, one can discard all negative examples that contain the word 'cat', such that in the synthetic data, all samples that mention 'cat' are positive. More generally, given the spurious token v_s and selected class ω_k, one can define the biased dataset $\mathcal{D}_b$ as,

$$\mathcal{D}_b = \{(\boldsymbol{x}^{(i)}, y^{(i)}) \mid (\boldsymbol{x}^{(i)}, y^{(i)}) \in \mathcal{D}, \neg(v_s \in \boldsymbol{x}^{(i)} \wedge y^{(i)} \neq \omega_k)\} \tag{5.10}$$

i.e. filter out all examples $\boldsymbol{x}$ where both $v \in \boldsymbol{x}$ and $y \neq \omega_k$. This has also been applied at the concept level, where instead of looking at whether an exact token is present, one can see if the input contains a concept such as food [361].

Existing research using both set-ups, injecting and filtering, demonstrated that NLP models may strongly exhibit introduced spurious associations. While these approaches demonstrate the susceptibility of these systems to spurious correlations, the setting may be artificial, as the injected shortcuts are designed to correlate highly with the target and be quite prevalent. Therefore, the results may not relate to the actual susceptibility of these systems when applied to real-world tasks. Hence, the next subsection will consider token-level spurious correlations that may exist within real datasets.

5.2.2 Token Level Spurious Correlations

Using injected spurious correlations (§5.2.1), although artificial, had advantages such as knowing the exact spurious correlation to analyse. For NLP tasks, the set of possible features is vast, and it is infeasible to check all features to see whether a correlation exists and classify whether each feature is spurious or genuine. Therefore, previous studies investigating

spurious correlations in NLP use a highly restricted set of features, with individual tokens [57, 102, 84, 327, 329] being a common choice. In this approach, the objective is to:

1. Identify the tokens that correlate with the target in the training data.
2. Examine whether flagged tokens are related to the task of interest or are spurious tokens.

For example, given a sentiment classification dataset $\mathcal{D}$, if the word 'dog' appears in a hundred positive reviews but only ten negative reviews, then a system trained on $\mathcal{D}$ may learn to associate the word "dog" with the positive class. The process of determining whether a token is spuriously correlated is to first identify words that correlate with the label (which 'dog' does) and then to determine whether the token is actually related to the task of interest (which 'dog' is not).

For the first stage, identifying correlated words, a common method is to go through each token v in the vocabulary $\mathcal{V}$ and calculate how correlated each token is with the target variable. Gururangan et al. [102] propose to use point-wise mutual information (PMI), where Monte-Carlo estimates are first used to estimate probability distributions,

$$\mathtt{P}(v;\mathcal{D}) = \frac{1}{Z}\sum_{i=1}^{M} \mathbb{1}(v \in \boldsymbol{x}^{(i)}) \qquad \mathtt{P}(\omega_k;\mathcal{D}) = \frac{1}{M}\sum_{i=1}^{M} \mathbb{1}(y^{(i)} = \omega_k) \tag{5.11}$$

$$\mathtt{P}(v,\omega_k;\mathcal{D}) = \frac{1}{Z}\sum_{i=1}^{M} \mathbb{1}(v \in \boldsymbol{x}^{(i)})\mathbb{1}(y^{(i)} = \omega_k) \tag{5.12}$$

where $Z = \sum_{v\in\mathcal{V}}\sum_{i=1}^{M} \mathbb{1}(v \in \boldsymbol{x}^{(i)})$ is a normalisation constant to ensure the token distribution is a valid PMF. The PMI, which captures the correlation between token v and class ω_k, can then be calculated as:

$$\mathrm{PMI}(v,\omega_k;\mathcal{D}) = \log\left(\frac{\mathtt{P}(v,\omega_k;\mathcal{D})}{\mathtt{P}(v;\mathcal{D})\mathtt{P}(\omega_k;\mathcal{D})}\right) \tag{5.13}$$

If the tokens provide no information on the label class, such that v and ω_k are independent, then $\mathtt{P}(v,\omega_k;\mathcal{D}) = \mathtt{P}(v;\mathcal{D})\mathtt{P}(\omega_k;\mathcal{D})$ and the PMI will be 0. However, a large magnitude for the PMI implies a large correlation within the training data, and hence, the token can be identified as an informative token within the training domain. For tokens that appear infrequently in the training data, the estimated correlations may be unstable and result in misleading high correlations. Therefore, add-100 smoothing can be applied, where for each class, the counts for each token are initialised to 100 and incremented for each further occurrence. Other

works have used other methods to measure informativeness; Wang and Culotta [329] train a bag-of-word logistic regression classifier and use the weights associated with each token as the scores. Gardner et al. [84] model bias as a rejection sampling data generation process and determine informative tokens by calculating the probability of rejecting the null hypothesis that token v is uninformative.

Sentiment	Word 1	Word 2	Word 3	Word 4
Positive	enjoyable	masterpiece	animated	Spielberg
Negative	failure	boring	heavy	Seagal

Table 5.1 Examples of the words found to be correlated with each sentiment class, as identified by Wang and Culotta [329] for sentiment classification.

Once the informative tokens are identified, the second stage is to then determine whether the tokens are spurious or not. As an illustration, Table 5.1 presents words that Wang and Culotta [329] found to be informative for each class within sentiment classification. Some of the informative words (positive, enjoyable, negative, failure) are genuinely related to the associated sentiment, while some words (animated, Spielberg, heavy, Seagal) are only spuriously correlated. To determine whether the highlighted token feature is spurious, human annotators can manually annotate whether the extracted informative tokens are 'genuine' or 'spurious' [327]. Wang and Culotta [329] manually annotate a subset of the words (e.g. 10%) and then train an automatic classifier to determine whether words are spurious or not. While Gardner et al. [84] argue that language understanding tasks require combining many features together, and on its own, the correlation of any single token with the target variable is spurious. This final view, however, does take quite a strong causal view. For example, the word 'phenomenal' may legitimately provide information that the review is likely positive, even though one can create adversarial examples to counter this.

The above process identifies words with spurious correlations in the training data. For example, Gururangan et al. [102] presented evidence that negative words such as *no*, *never*, and *nothing* are indicators of the contradiction class [102], highlighting annotation artefacts in the dataset. A further practical consideration is whether these spurious signals are learned by models when trained on the data. A common method of identifying impact is to produce challenge sets where the spurious features are designed to not correlate at all with the labels [210, 148]. Another approach is to change domains, where the cooccurrence of words

with labels differs, and to identify whether the model overly relies on the identified correlation [84].

5.3 Spurious Stopword Correlations

A limitation of token-level spurious correlation methods (§5.2) is that after identifying informative tokens, manual annotation is required to determine whether a correlated token is genuine or spurious [329, 327]. Furthermore, the approach only considers the presence of isolated tokens and does not consider more complex features, such as the interactions between different words.

Therefore, extending the idea of spurious token correlation, we propose to analyse spurious stopword correlations. Instead of considering all tokens in the entire vocabulary $\mathcal{V}$, we propose to only consider the tokens within the set of stopwords, $\mathcal{V}_{sw}$. Stopwords such as 'and', 'the', and 'of' are common in language, however they typically only play a syntactic role and are uninformative for text classification tasks. As such, any feature derived from stopwords alone can be assumed to be unrelated to the task and within the set of irrelevant features. This also enables us to consider more complex features composed of groups of words, not just isolated tokens as done in the previous section.

For our approach, we engineer a single spurious stopword feature that is designed to be correlated with the target variable. Given a supervised training dataset, $\mathcal{D}$, for each class ω_k, the stopword distribution $\mathtt{P}_{sw}(v;\omega_k)$ over stopword tokens $v \in \mathcal{V}_{sw}$ can be calculated,

$$\mathtt{P}_{sw}(v;\omega_k) = \frac{\sum_{i=1}^{M}\sum_{j=1}^{N} \mathbb{1}(y^{(i)} = \omega_k)\mathbb{1}(x_j^{(i)} = v)}{\sum_{i=1}^{M}\sum_{j=1}^{N} \mathbb{1}(y^{(i)} = \omega_k)\mathbb{1}(x_j^{(i)} \in \mathcal{V}_{sw})} \tag{5.14}$$

where $x_j^{(i)}$ denotes the j-th token of $\boldsymbol{x}^{(i)}$ and where for simplicity the dependence of N with i is dropped. We propose the spurious stopword feature, ψ_{sw}, which measures how well the input text's stopwords follow the stopword distribution of a particular class. For a binary task with two classes, $\{\omega_1, \omega_2\}$, one can use the log probability ratio as the spurious feature,

$$\psi_{sw}(\boldsymbol{x}^{(i)}) = \log\left(\frac{\prod_{j=1}^{N} \mathbb{1}(x_j^{(i)} \in \mathcal{V}_{sw}) \cdot \mathtt{P}_{sw}(x_j^{(i)};\omega_1)}{\prod_{j=1}^{N} \mathbb{1}(x_j^{(i)} \in \mathcal{V}_{sw}) \cdot \mathtt{P}_{sw}(x_j^{(i)};\omega_2)}\right) \tag{5.15}$$

$\psi_{sw}(\boldsymbol{x})$ therefore measures whether the stopwords in $\boldsymbol{x}$ better matches the unigram stopword distribution of class ω_1 or of class ω_2. To determine the extent of information captured by

this one spurious stopword feature, one can calculate the accuracy when leveraging this single spurious feature. For the binary case, this takes the form:

$$\text{acc}(\psi_{sw}) = \frac{1}{M}\sum_{i=1}^{M}\Big(\mathbb{1}(y^{(i)}=\omega_1)\cdot\mathbb{1}(\psi_{sw}(\boldsymbol{x}^{(i)})\geq 0)+\mathbb{1}(y^{(i)}=\omega_2)\cdot\mathbb{1}(\psi_{sw}(\boldsymbol{x}^{(i)})<0)\Big) \quad (5.16)$$

If no information is captured by the spurious feature, the accuracy should be 50%. The larger the accuracy is over these baseline values, the more information is captured, which indicates a stronger spurious correlation between the stopword feature and the target variable.

Alternatively, instead of limiting the features to those using the unigram stopword distributions, we also explore using deep learning models to extract spurious stopword information. Here, a deep learning system predicts the label using only stopword information, where the system can freely derive its features that may capture more complex relationships between stopwords and the label. To train the 'shortcut' system, we introduce a shuffled stopword (SSW) data corruption method. In this approach, the input $\boldsymbol{x}$ is corrupted by only retaining the stopwords and then shuffling their order, producing a non-informative text, $\tilde{\boldsymbol{x}}_{sw}^{(i)}$, as illustrated in Figure 5.2. The system is then trained to extract the label information from the shuffled stopwords alone, where the training criterion for the "shuffled stopword system" is:

$$\mathcal{L}_{ssw}(\boldsymbol{\theta}) = \frac{1}{M}\cdot\sum_{i=1}^{M}\log \mathrm{P}(y^{(i)}|\tilde{\boldsymbol{x}}_{sw}^{(i)};\boldsymbol{\theta}) \quad (5.17)$$

$$\tilde{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \mathcal{L}_{ssw}(\boldsymbol{\theta}) \quad (5.18)$$

Similarly, the accuracy of this system on unseen data indicates the amount of information present from the spurious stopword features.

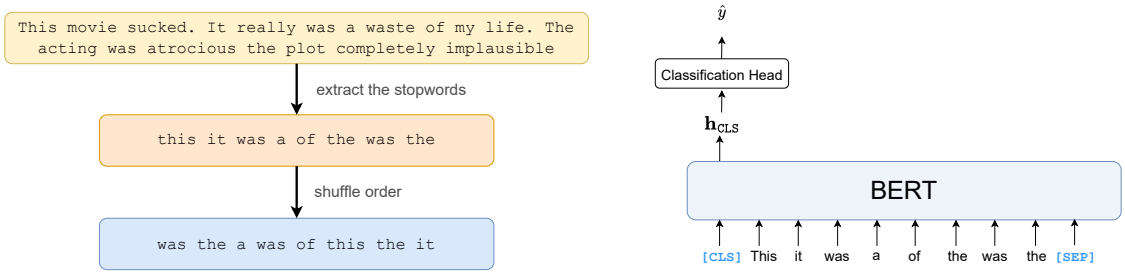


Fig. 5.2 **left:** The shuffled stopword (SSW) data corruption method for spurious feature extraction. **right:** Example of a BERT-based system that takes the shuffled stopword and generates its prediction (based on the framework described in Section 4.1.2)

5.4 Spurious Questions

This chapter has so far considered spurious features, features that appeared to be informative for the task but were not actually related to the task of interest. This section now considers *spurious questions*, which are questions that appear to be informative for assessing a task but may not actually assess the task of interest.

Tasks are designed to evaluate a particular attribute. If a task is poorly designed, one may solve the task without demonstrating the intended ability, which can lead to inaccurate assessments. For example, if a parking exam is always conducted in the same location with fixed marker positions, candidates might pass by memorising specific movements (e.g. "turn the wheel 45° when the red cone appears in the left mirror"). In this case, the task does not accurately measure the attribute of interest, and passing the exam doesn't demonstrate the ability to park cars in real-world situations.

Similarly, questions in NLU tasks aim to measure a particular ability. In the case of multiple-choice reading comprehension (MCRC) [160, 130], MCRC exams are designed to assess a candidate's ability to understand written text. These exams have been widely adopted for various assessments, including language proficiency tests, university entrance exams, and professional certifications [4, 213]. A core assumption in MCRC exams is that to correctly answer questions, the candidate has to read the passage, comprehend its meaning, and identify the relevant information for each query. However, if a candidate can consistently predict correct answers without actually reading or understanding the text, then the exam fails to reliably measure the intended comprehension skills.

A spurious question can be defined as a question which candidates can answer without demonstrating the core skill being assessed. Assessing ability based on such questions can create the illusion of measuring the intended ability, while in actuality, the candidate is leveraging information through unintended shortcuts or peripheral knowledge. In this section, we propose a method of identifying spurious MCRC questions, where we train shortcut systems that can solve questions without using necessary task information. We then analyse whether these systems can be deployed to detect spurious questions, questions which may compromise the exam's validity in truly assessing reading comprehension for real-world candidates.

5.4.1 World-Knowledge Shortcut for MCRC

Context: Make-A-Wish" is one of the world's most well-known charities. It makes wishes come true for children who have serious illnesses. It gives them hope and joy and helps them forget about their health problems and have fun. It all started in 1980 in Phoenix, Arizona. Christopher was a 7-year-old boy who was very sick. He always dreamed of becoming a police officer. Tommy Austin and Ron Cox, two police officers, made his wish come true. They gave Christopher a tour of the city in a police helicopter and made a real police uniform for him. There are four kinds of wishes children usually have: I wish to go. Children usually want to travel or go to a concert, a game or a park. I wish to meet. Children sometimes want to meet their favourite actors, singers or players. I wish to be. Some children wish to become actors, singers or police officers. I wish to have. They often want to have a computer, a game, a bike or many other things. Let's hope more wishes will come true in the future. People who work in the charity always try for the best. Almost 25,000 Volunteers help, work or give money. Will you be one of them?

Question: Make-A-Wish" is a charity to help _?

Options: **A)** sick children **B)** serious officers **C)** famous actors **D)** popular singers

Fig. 5.3 Example MCRC question taken from RACE [160].

Multiple-choice reading comprehension (MCRC) is a popular task where the input contains a context passage $\boldsymbol{c}$, question $\boldsymbol{q}$ and a set of options $\boldsymbol{a}_{1:K}$. As illustrated in Figure 5.3, the objective is to determine the correct answer from $\boldsymbol{a}_{1:K}$ that correctly answers question $\boldsymbol{q}$ based on the information provided in the context $\boldsymbol{c}$. MCRC has been widely used in real-world examinations to assess candidates' reading comprehension abilities [4, 223], and has also been adopted in Natural Language Understanding (NLU) research to analyse the language comprehension capabilities of automated systems [160, 130]. However, paying closer attention to the example in Figure 5.3, a real example from RACE [160], the question appears to be answerable even without reading the context $\boldsymbol{c}$. Provided the context wasn't designed adversarially, one would confidently guess that "the make-a-wish foundation" helps "sick children" even without reading the context. This occurs as MCRC questions often use real-world articles and data, and since the information is factual, questions may be answerable using world knowledge alone. While the ability to answer questions using world knowledge may generalise across different MCRC datasets and domains, it differs fundamentally from answering questions through reading comprehension. Since MCRC tasks are designed to assess comprehension ability, questions that can be answered solely with world knowledge pose a risk, as human candidates may similarly leverage the same shortcut, potentially bypassing the intended goal of assessing reading comprehension.

To flag questions that are answerable without performing comprehension, we propose to train a shortcut system which only uses context-free inputs. In this setting, the model does not have access to the context and must rely on the prediction rules derived from superficial shortcuts using the question and options alone. Let the dataset $\mathcal{D}$ represent an MCRC dataset,

$$\mathcal{D} = \{\boldsymbol{q}^{(i)}, \boldsymbol{c}^{(i)}, \boldsymbol{a}^{(i)}_{1:K}, y^{(i)}\}_{i=1}^{M} \tag{5.19}$$

where the label $y^{(i)} \in \{\omega_1, \ldots, \omega_K\}$ denotes the position of the answer for the i-th example, such that ω_k denotes that the correct answer is $\boldsymbol{a}_k$. An MCRC system can model the probability that the answer being in the k-th position (§4.1.2), $\mathtt{P}(\omega_k|\boldsymbol{q}^{(i)}, \boldsymbol{c}^{(i)}, \boldsymbol{a}^{(i)}_{1:K}; \hat{\boldsymbol{\theta}})$, and trained by maximising the likelihood of the training data,

$$\mathcal{L}(\boldsymbol{\theta}) = -1 \cdot \frac{1}{M} \cdot \sum_{i=1}^{M} \log \mathtt{P}(y^{(i)}|\boldsymbol{q}^{(i)}, \boldsymbol{c}^{(i)}, \boldsymbol{a}^{(i)}_{1:K}; \boldsymbol{\theta}) \tag{5.20}$$

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}) \tag{5.21}$$

In the shortcut setting, the system can instead be trained to answer MCRC questions given *only* the question and options, $\mathtt{P}(y^{(i)}|\boldsymbol{q}^{(i)}, \boldsymbol{a}^{(i)}_{1:K}; \tilde{\boldsymbol{\theta}})$. The training objective is to maximise the likelihood of the correct answers while omitting the context, defined as:

$$\mathcal{L}_s(\boldsymbol{\theta}) = -1 \cdot \frac{1}{M} \cdot \sum_{i=1}^{M} \log \mathtt{P}(y^{(i)}|\boldsymbol{q}^{(i)}, \boldsymbol{a}^{(i)}_{1:K}; \boldsymbol{\theta}) \tag{5.22}$$

$$\tilde{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \mathcal{L}_s(\boldsymbol{\theta}) \tag{5.23}$$

where $\tilde{\boldsymbol{\theta}}$ are the final parameters for the shortcut system. If reading comprehension is a core requirement for the task, then without context, the shortcut system should have no information on the position of the correct answer and be expected to predict a uniform distribution. However, if the model can bypass the intended task, for example by using world knowledge, then the system may have an accuracy better than random. A simple way to quantify the amount of information present is to measure the performance of the shortcut system when applied to unseen task data:

$$\text{acc}(\tilde{\boldsymbol{\theta}}) = \frac{1}{M} \sum_{i=1}^{M} \mathbb{1}\left(y^{(i)} = \arg\max_{y \in \mathcal{Y}} \mathtt{P}(y|\boldsymbol{q}^{(i)}, \boldsymbol{a}^{(i)}_{1:K}; \tilde{\boldsymbol{\theta}})\right) \tag{5.24}$$

If the resulting accuracy is larger than $\frac{1}{K}$, then this implies the system can solve elements of questions while bypassing the intended comprehension task. The ability of the shortcut

system will be strongly influenced by the scale and diversity of data the system is exposed to during pre-training and fine-tuning. As the base foundation models grow larger and use more extensive training corpora, their accessible world knowledge will naturally increase. In some cases, if the MCRC questions themselves are encountered in the training data, the model may effectively "memorize" the correct answers, which may result in data leakage and artificially inflated performance.

5.4.2 Identifying Spurious Questions

The previous subsection (§5.4.1) introduced the idea of a shortcut system, a system designed to mimic reading comprehension while not having access to the context. Although the shortcut system's accuracy can serve as a general metric of the information it can extract without performing the intended task, it does not provide question-specific insights. A possible hypothesis is that if the system can confidently get a question correct without using the context, human candidates may also be able to get the same questions correct while bypassing comprehension. Therefore, the entropy of the answer distribution from the shortcut system can be used to obtain question-specific confidence levels for a given input question,

$$\mathcal{H}(y|\boldsymbol{q},\boldsymbol{a}_{1:K};\tilde{\boldsymbol{\theta}}) = \sum_{k=1}^{K} \mathtt{P}(\omega_k|\boldsymbol{q},\boldsymbol{a}_{1:K};\tilde{\boldsymbol{\theta}}) \log_2 \mathtt{P}(\omega_k|\boldsymbol{q},\boldsymbol{a}_{1:K};\tilde{\boldsymbol{\theta}}) \tag{5.25}$$

This can be converted to the more intuitive "effective number of options" $\mathcal{E}$, which is bounded between 1 and the number of classes,

$$\mathcal{E}(y|\boldsymbol{q},\boldsymbol{a}_{1:K};\tilde{\boldsymbol{\theta}}) = 2^{\mathcal{H}(y|\boldsymbol{q},\boldsymbol{a}_{1:K};\tilde{\boldsymbol{\theta}})} \tag{5.26}$$

This metric quantifies the certainty of the shortcut system in its decision for the current MCRC question. The resulting value is expressed as the "effective number of options" the model is deciding between. If the model is completely confident of its answer, the effective number of options will be 1. In contrast, if the model has no information and predicts a uniform distribution, the effective number of options will be K. For the shortcut system, which uses insufficient task information, since no answer information should be available without the context, reliable questions may be expected to have an effective number of options of K. A question with a low entropy implies that it can be answered while bypassing the intended task, and hence, such questions can then be flagged and examined to determine whether real candidates can similarly solve them without access to the context.

A second metric that may also be useful for analysing the quality of MCRC questions is the mutual information [295]. This metric quantifies the amount of information that the

system gains by having access to the context,

$$\mathcal{I}(y|\boldsymbol{c};\boldsymbol{q},\boldsymbol{a}_{1:K}\hat{\boldsymbol{\theta}},\tilde{\boldsymbol{\theta}}) = \mathcal{H}(y|\boldsymbol{q},\boldsymbol{a}_{1:K};\tilde{\boldsymbol{\theta}}) - \mathcal{H}(y|\boldsymbol{q},\boldsymbol{c},\boldsymbol{a}_{1:K};\hat{\boldsymbol{\theta}}) \tag{5.27}$$

The mutual information captures the reduction in entropy between the shortcut system (parameterised by $\tilde{\boldsymbol{\theta}}$) and the standard MCRC system (parameterised by $\hat{\boldsymbol{\theta}}$), reflecting the information gained from incorporating context. A mutual information value near 0 suggests that the system's uncertainty remains unchanged regardless of whether it is using the context or not. On the other hand, questions with mutual information close to K indicate that the system has no information on the answer without the context but can solve the question confidently when the context is provided. This describes well-designed questions, so questions with high mutual information are desirable.

5.5 Experiments

5.5.1 Spurious Stopword Correlations

The first set of experiments investigates the stopword spurious correlation proposed in Section 5.3, where we analyse whether the spurious stopword correlations exist in binary classification tasks. We first investigate whether stopword-based features can be designed to spuriously correlate with task labels and then evaluate whether models trained on standard NLP datasets unintentionally learn these correlations, which may potentially influence their decisions.

Datasets

To validate whether stopwords carry spurious signals, we investigate a diverse range of standard NLP benchmarks. For sentiment classification, we use three popular movie review datasets, **IMDb** [200], Rotten Tomatoes (**RT**) [237] and the Stanford Sentiment Treebank v2 dataset (**SST-2**) [298]. These are all movie review datasets sourced from different movie review platforms, where the labels are either 'positive' or 'negative'. Further, **Twitter**'s Emotion dataset [285] is also used, where tweets are categorised into one of six emotions that we map to either positive (love, joy and surprise) or negative (fear, sadness and anger). The **Yelp** dataset [350] consists of reviews from the Yelp platform, where the scores of 1-5 stars are split into positive (4,5) and negative (1,2) reviews. Finally, **BoolQ** [44] is a reading comprehension dataset where each example is a yes/no question, a relevant passage, and the corresponding answer (*yes* or *no*) that indicates whether the passage supports the question.

All datasets are perfectly balanced, apart from Twitter, which is filtered to be balanced. Table 5.2 gives the sizes of the train validation and test splits of all the datasets after processing.

Dataset	IMDb	RT	Twitter	SST-2	Yelp	BoolQ
Train	20,000	8,530	16,000	6,920	448,000	9,426
Validation	5,000	1,066	2,000	872	112,000	3,270
Test	25,000	1,840	2,000	1,820	38,000	3,270

Table 5.2 Dataset split sizes for the sentiment classification datasets and BoolQ.

Model training details

BERT-base [51] is used as the base model for fine-tuning, where the fine-tuning set-up discussed in Section 4.1.2 is used. The model is trained using the Adam optimiser [152] for a maximum of 4 epochs (with early stopping on the validation split) with a learning rate of $1e^{-5}$ and a batch size of 8. The parameters were determined using a grid search over hyperparameters in the range learning rate $\in \{1e^{-5}, 2e^{-5}, 5e^{-5}\}$ and batch size $\in \{4, 8, 16\}$. We also consider a randomly initialised transformer (RIT), which has the same architecture as a BERT-based system but with all parameters randomly initialised instead of using BERT's pre-trained weights. We analyse the process of fine-tuning the RIT as an ablation study to investigate the impact of pre-training on spurious correlations and to determine whether pre-training can serve as a source of robustness to spurious correlations [113]. All input samples are truncated to 512 tokens to fit within the maximum length supported by BERT. All results are reported by averaging the probabilities of the decisions from three models (which use different data shuffling and weight initialisations) for each experiment.

Presence of Spurious Stopword Signals

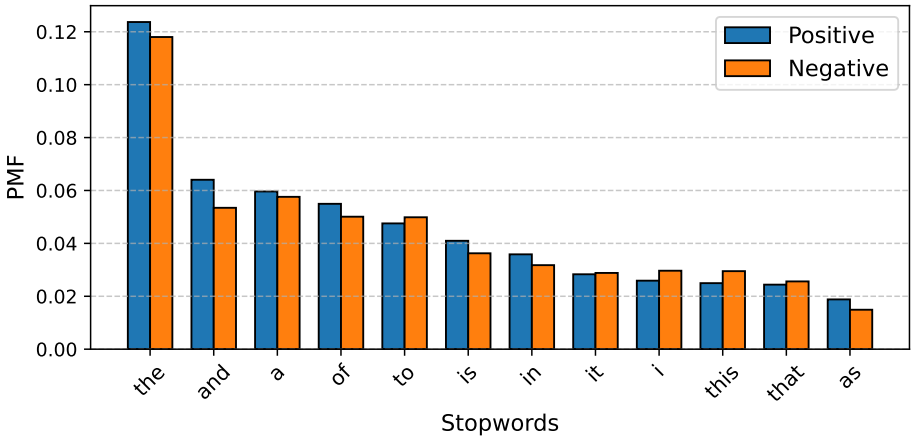


Fig. 5.4 Comparison of the unigram distribution between positive and negative classes in the IMDb training set, with the probabilities for the top 12 stopwords presented.

The first set of experiments investigates whether correlations exist between irrelevant features based on stopwords and the labels across various classification tasks. Figure 5.4 shows the unigram stopword distribution for the positive and negative classes in the IMDb training dataset, which contains 20,000 movie reviews. While the overall distributions appear similar, there are subtle differences in stopword frequencies between the positive and negative classes. Although such stopword information might provide predictive information, these would be highly specific to the domain and will not generalise for the task and instead be a form of spurious correlation.

Table 5.3 shows that systems that only use stopword information are able to achieve accuracies significantly better than random. Two methods are presented: the stopword likelihood ratio and the shuffled stopword system. The stopword likelihood ratio (LR) calculates the likelihood using the unigram stopword distribution of both classes in the training data (Equation 5.14), while the shuffled stopword system (SSW) is the performance of a BERT-base system that has been fine-tuned to predict the label using only the shuffled stopwords (Equation 5.17). This is baselined against the performance of a BERT-base model that is directly fine-tuned for the actual classification task, as well as random performance for the task. The results demonstrate that stopwords information alone can be used to achieve reasonable accuracies in text classification tasks. Both SSW and LR achieve accuracies significantly higher than the expected random value of 50% in all tasks, with the SSW accuracy at 77.3% and 68.7% for Yelp and IMDB, respectively. Additionally, the use of unigram stopword frequency information alone yields considerable predictive information,

Method	IMDb	RT	Twitter	SST-2	Yelp	BoolQ
Random	50.0	50.0	50.0	50.0	50.0	50.0
Standard fine-tuning	94.2	85.2	98.4	92.4	97.6	66.9
likelihood ratio	64.3	60.4	58.2	62.4	70.4	57.5
Shuffled stopword	68.7	60.5	57.8	60.3	77.3	63.1

Table 5.3 Accuracy (%) of methods using stopword information. 'Likelihood ratio' *LR* refers to the performance when stopword likelihood ratio is used (Equation 5.14) and 'shuffled stopword' *SSW* when a BERT-base system is fine-tuned to predict decisions using only shuffled stopwords. *Standard* is the baseline when a BERT system is fine-tuned for the standard task, and *random* is the expected performance with random guessing.

with LR achieving accuracies ranging from 57.5% to 70.4%. This highlights that the proposed stopword feature, which is designed to be within the space of irrelevant features, is a pronounced spurious correlation that the system may be prone to relying on when trained on the data.

Model Influence of Stopword Correlations

	Standard			SSW		
Model	In	Shifted	Out	In	Shifted	Out
BERT	94.2	82.1	71.2	57.7	53.7	50.0
RIT	88.2	73.7	59.1	60.0	57.3	50.5

(a) In=IMDb, Shifted=RT, Out=Twitter

	Standard		SSW	
Model	In	Out	In	Out
BERT	97.6	87.8	56.6	51.7
RIT	93.0	71.4	65.0	58.3

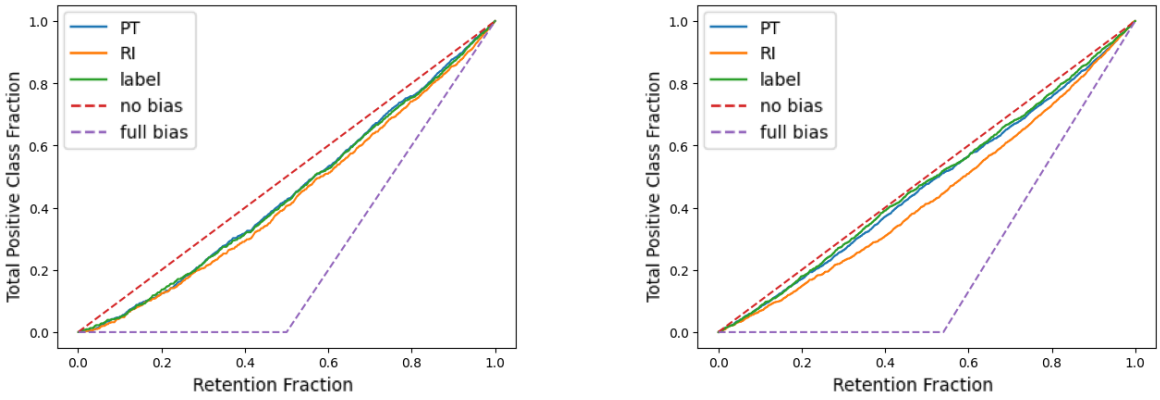
(b) in=Yelp, out=SST-2

Table 5.4 Accuracy (%) of a BERT-base system and randomly intialised transformer (RIT) when finetuned and then evaluated in the training domain (in), in a shifted domain (shifted) and out-of-domain (out). Models are trained and evaluated in both the normal (*standard*) setting, as well as the *SSW* setting where shuffled stopwords are provided.

Relying on spurious correlations can result in decisions that may not generalise outside the training domain. The previous results demonstrate that spurious stopword features could provide predictive information in various classification tasks, implying that spurious correlations exist between stopword frequencies and class labels. However, we did not establish whether NLP systems trained normally on the datasets rely on, or are even sensitive to, the identified stopword-based spurious correlations. The next experiments investigate this, and two systems are fine-tuned on standard NLP datasets: a pre-trained BERT-base model and a randomly initialised transformer (RIT). The systems are then analysed by measuring

the performance degradation when subjected to a domain shift, as well as the system's performance on corrupted inputs.

Table 5.4a presents a set-up where IMDb is used as the training domain, RT as the shifted domain (i.e. a small domain-shift) and Twitter as out-of-domain (a significant domain shift). Table 5.4b uses the Yelp dataset as the training domain and SST-2 as the out-of-domain. In the standard setting, when trained on IMDb, BERT achieves an in-domain accuracy 6% better than RIT, and its performance drops by 12.1% in the shifted domain and 23.0% in out-of-domain, while RIT drops by 14.5% and 29.1%, respectively. This highlights that pre-training results in systems that not only have higher in-domain accuracies but are also more robust to domain changes. The systems are also evaluated using SSW evaluation, where the inputs are filtered to stopwords only and then shuffled. Though the systems are trained using full-text inputs, when evaluated on samples corrupting with the SSW text corruption, the in-domain accuracies remain in the range of 56.6%-65.0%, considerably above the random performance of 50%. This provides evidence that models pick up and possibly exploit spurious stopword correlations. Additionally, BERT appears less prone to exploiting spurious stopword correlations than the RIT, with accuracies consistently lower than RIT's in all SSW settings. This may be due to BERT's pre-training, which helps it to identify salient linguistic features and reduce reliance on spurious cues like stopwords, which may explain its better robustness to domain shifts [113].



(a) Shifted-domain (RT)

(b) Out-of-domain (Twitter)

Fig. 5.5 Retention plots of the likelihood ratio (LR) across different domains. **PT**: Pre-trained (BERT) predictions, **RI**: Randomly initialised (RIT) predictions, **label**: Ground truth labels.

To further determine whether models rely on spurious features, we generate retention plots using the likelihood ratio using the IMDB training stopword distribution. The retention plot selects the $r\%$ of samples with the lowest likelihood ratio and determines what proportion of the samples were predicted to be positive. If the feature (i.e. the likelihood ratio) is

independent of the predictions, the expected retention plot is a straight diagonal line, $y=r$. If the designed feature is alone sufficient for perfect classification performance, then for a balanced dataset with two classes, there will be a flat horizontal line up to $r=0.5$ (since no points are from the positive class) followed by a steep increment (as then all remaining examples are in the positive class). The OOD retention plot (Figure 5.5) illustrates that models can be susceptible to using spurious stop-word correlations. Although in this domain, the true labels have no relationship with the likelihood ratio features (or very weak correlations), the RIT makes biased predictions, where samples with a high LR are more likely to be classified as positive. BERT, however, only shows a mild bias to the stopwords, further illustrating that pre-trained models may be better at avoiding spurious features.

5.5.2 Identifying Spurious Questions

The next experiments focus on identifying spurious questions (Section 5.4). To analyse whether systems can solve MCRC questions without access to the context, we first train shortcut MCRC systems that do not have access to the contexts. We then investigate whether these shortcut systems can help identify unreliable questions that poorly assess comprehension, questions that humans might also solve without fully engaging in comprehension, possibly by relying on world knowledge. This approach could enable test-makers to filter out MCRC questions that fail to assess comprehension effectively, leading to more reliable exams for real-world applications.

Datasets

Three popular MCMRC datasets are considered: RACE++ [160, 175], CosmosQA [130] and ReClor [343]. **RACE++** is a dataset of English comprehension questions for Chinese high school students, taken at a range of difficulties from middle school to college level. **CosmosQA** is a large-scale reading comprehension dataset that gathers people's everyday narratives and poses questions about the reasoning behind their statements and actions. **ReClor** is a logical reasoning dataset based on questions from standardised tests like the GMAT and LSAT, designed to assess graduate-level logical reasoning skills. All datasets have four options per question, with one correct answer. Dataset statistics are provided in Table 5.5.

	Train	Validation	Test
RACE++	100,388	5,599	5,642
COSMOS	25,262	2,985	–
ReClor	4,638	500	1000

Table 5.5 Dataset statistics for MCRC datasets.

Model training details

For the MCRC experiments, ELECTRA-large is chosen as the foundation model over BERT, given its superior performance on MCRC tasks [268]. The fine-tuning framework described in 4.1.2 is used for training. Following the hyperparameters from [268], RACE++ and CosmosQA models are trained with the Adam optimiser for 2 epochs, using a learning rate of 2e-6, a batch size of 4, and input sequences truncated to 512 tokens. For ReClor, the same settings are applied, except training is extended to a maximum of 10 epochs. Fine-tuning is conducted on the training split, with hyperparameters selected at each epoch based on validation performance, and final results are reported on the test set. Since the test sets for CosmosQA and ReClor are not publicly available, the original validation set is used as the test set, while the training set is further split in an 80:20 ratio into the new training and validation subsets.

Shortcut System Performance

The first stage of the process for detecting spurious questions involves training shortcut systems designed to solve questions without relying on sufficient task information. If a system can solve questions without key information, such as the context in reading comprehension tasks, then human candidates may also be able to do so. In addition to the shortcut system trained without context (Q +{O}), we also examine systems trained with only the options (O}), no question ({O} + C), as well as the standard system with all task information to solve the standard MCRC task (Q +{O}+C).

Table 5.6 presents the performance of the various systems, also providing cross-domain performance. The shortcut systems achieve high performance across all MCMRC datasets, with all context-free systems (Q +{O}) achieving accuracies above 50%, significantly above the expected random performance of 25%. Further, we find that the context-free systems can generalise across domains, with the context-free system trained on RACE++ achieving an accuracy of 54.0% when evaluated on CosmosQA. Other shortcut systems can also perform better than random. For instance, the option-only system ({O}) achieves an accuracy of

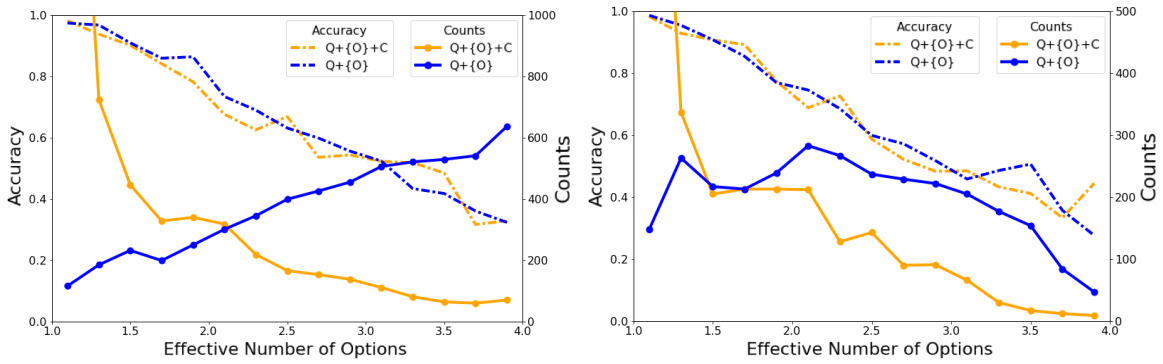
57.4% on CosmosQA, suggesting that certain characteristics of the options alone may provide clues to the correct answer. This is potentially problematic, as for MCRC questions, the correct answer should be indistinguishable from the other options when the context and question are unknown. However, the option-only prediction rules do not transfer to other domains, with the option-only systems performing near random performance when evaluated on other datasets. This difference between the context-free (Q +{O}) and option-only ({O}) systems might be explained since world knowledge can also be used to solve questions across different domains, but the characteristics of the correct answers are likely to diverge across different tasks.

Training data		RACE++	CosmosQA	ReClor
–		25.0	25.0	25.0
RACE++	{O}	**41.8**	21.4	34.0
	Q+{O}	**57.3**	54.0	34.8
	{O}+C	**68.2**	54.6	46.0
	Q+{O}+C	**85.0**	70.0	48.6
CosmosQA	{O}	30.0	**57.4**	25.2
	Q+{O}	38.7	**68.5**	27.8
	{O}+C	52.4	**79.0**	40.4
	Q+{O}+C	66.8	**84.5**	41.2
ReClor	{O}	26.1	18.3	**49.0**
	Q+{O}	31.3	33.1	**51.8**
	{O}+C	39.8	36.9	**68.4**
	Q+{O}+C	52.7	41.7	**69.8**

Table 5.6 Accuracy of various shortcut MCRC systems trained on RACE++, CosmosQA, and ReClor, with cross-domain evaluation. {O} denotes a system trained with only the options, ({O}+C), with options and context, ({O}+Q) without the context, and (Q+{O}+C) the standard system with all inputs.

Identifying Spurious Questions

The previous results highlighted that within particular domains, systems could exploit particular task shortcuts to avoid comprehension yet achieve reasonable performance. Next, we investigate whether these systems can be used to identify particular questions where comprehension can be bypassed. Section 5.4.2 presented proposed metrics that capture



(a) RACE++ (b) CosmosQA

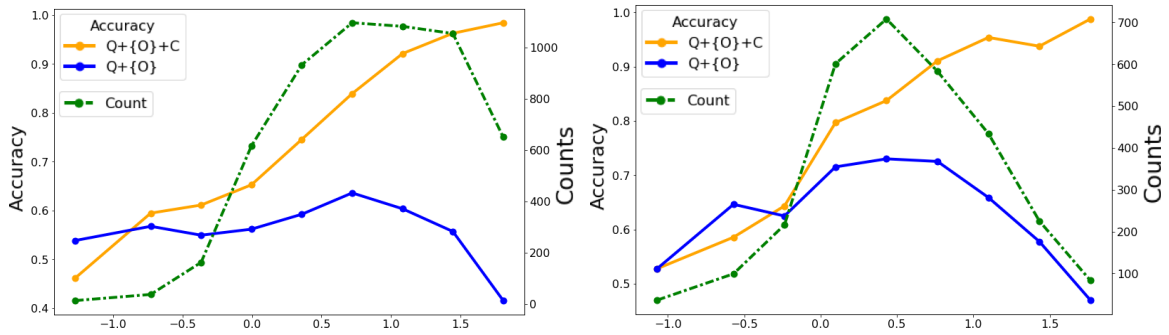
Fig. 5.6 Distribution of effective number of options and corresponding (binned) accuracy.

question quality, namely the *effective number of options* (ENO) and *the mutual information* (MI).

Figure 5.6 presents the count and accuracy plots of the effective number of options (with a bin width of 0.2) for the standard (Q+{O}+C) and the context-free (Q+{O}) systems on RACE++ and CosmosQA. The count shows the number of questions with ENO values within the bin range, while accuracy refers to the accuracy of all the examples within the bin. Since the systems are slightly overconfident[1], the systems' output probabilities are calibrated using temperature annealing [99] (§6.2.3). The ENO is a metric that measures model confidence. The standard system has high certainty for most points (an ENO close to 1), which is expected given the baseline's high accuracy. However, the context-free system, despite having no contextual information, also has a considerable number of examples in the very low entropy region, with just over 100 questions in the first bin for both RACE++ and CosmosQA. This shows that for a subset of questions, the system is confident in its prediction without doing any comprehension at all. In other cases, the shortcut system can leverage some information from the question and can reduce the number of effective options to between 2-3, which implies that certain poor distractors can be eliminated by the question alone. We also show that for both models, there is a clear linear relationship between uncertainty and accuracy. This suggests that the system is well-calibrated and that the effective number of options is a good measure of actual model uncertainty.

To further examine the system's use of context, we reproduce the plots using mutual information (MI) as the metric instead of ENO. For each sample, the mutual information is approximated using Equation 5.27. Examples with a high MI are questions where the model is certain of the answer with context but uncertain without context - a desired property for comprehension questions. Figure 5.7 shows the counts when all the examples are binned

[1]For both models, the mean of the maximum probability is 5% above the overall accuracy.



(a) RACE++ (b) COSMOSQA

Fig. 5.7 Distribution of counts and corresponding accuracy when points are sorted by Mutual Information approximation.

by MI and the corresponding accuracies. The count distribution (the green line) has a mix of high and low MI questions, showing that the use of context varies over questions. As expected, the accuracy of the baseline system increases when the context is used, while the accuracy of the shortcut system falls. Although the MI should always be positive, a few questions have negative values since the models only approximate the true underlying distributions. The low accuracy of the shortcut model on negative MI questions occurs when standard world knowledge is not consistent with the information in the context, and so the context-free confidence was misleading.

Finally, to assess whether our metrics effectively identify questions where comprehension can be bypassed, we run a human evaluation. We selected the 100 RACE++ questions with the lowest and highest ENO scores and asked human assessors to answer the questions without access to the context. All questions were shuffled, the volunteers had native English proficiency, and the volunteers were instructed to answer each question to the best of their ability. Table 5.7 demonstrates that, without the context, participants were often able to answer the high ENO questions, achieving an average accuracy of 91.7% on the 100 lowest entropy examples, compared to 31.7% on the 100 highest entropy examples. This highlights that humans can use world knowledge to bypass comprehension, and calculating the ENO enables us to detect spurious questions.

We also selected 50 RACE++ questions with the lowest and highest MI scores, where the human assessors first answered the questions without the context and then with the context. This was used to calculate the increase in accuracy when candidates have access to the context, as shown in Table 5.8. For high-MI questions, participants experienced a 69.3%

accuracy increase when the context was provided, compared to a 24.7% boost for low-MI questions, also highlighting the utility of the MI metric.

	Low ENO	High ENO
Human	$91.7_{\pm 1.9}$	$31.7_{\pm 2.9}$
System	$99.0_{\pm 0.0}$	$24.3_{\pm 6.2}$

Table 5.7 Accuracy without the context for low and high ENO questions.

	High MI	Low MI
Human	$\Delta 69.3_{\pm 0.9}$	$\Delta 24.7_{\pm 5.0}$
System	$\Delta 68.0_{\pm 0.9}$	$\Delta 3.3_{\pm 4.7}$

Table 5.8 Change in accuracy when given the context for high and low MI questions.

5.6 Chapter Summary

This chapter considered the risk of learning spurious correlations when training NLP foundation models with domain-specific training data. The chapter discussed why learning spurious correlations is a risk for NLP tasks, caused by the vast and abstract set of features for NLP tasks. The chapter then discussed various approaches to detect whether spurious correlations are present in NLP datasets and methods of examining the influence they may have on the decisions of actual NLP systems. Experiments demonstrated that by designing spurious stopword features, these features could have strong predictive information for various text classification tasks that trained systems appeared to pick up on. We also investigated the risk of bypassing comprehension in multiple-choice reading comprehension, where systems without the context could achieve reasonable accuracies and mimic 'comprehension'. This was then used to design metrics that identify spurious questions, which was shown to be effective in an initial human evaluation study.

Chapter 6

Bias in Zero-Shot Classifiers

The previous chapter (§5) discussed spurious correlations, where NLP foundation models fine-tuned to particular tasks were prone to picking up on spurious signals when making decisions. Although finetuning has become a popular paradigm, there has been the recent introduction of instruction-following LLMs [307, 193, 2]. These models are also pre-trained on large quantities of data but are also further aligned to follow natural instructions (§3.3) through prompts. Instruction-following LLMs have displayed broad abilities, acting as generalist systems that can perform a wide range of tasks in few-show or even zero-shot settings [330, 314, 2]. Although their strong generalisation capabilities suggest that these systems do not rely on spurious artefacts and correlations, these models can exhibit biases, favouring certain patterns or characteristics in ways that may lead to inaccurate or unfair outcomes [101, 247]. Biases in generative LLMs can take many forms, from discriminatory biases where a model perpetuates social biases implicitly seen in the data (e.g. gender or racial biases) [162, 101] to cognitive biases [315, 134, 146] and not recognising particular task invariances. This chapter studies bias present in instruction-following LLMs when applied as zero-shot text classifiers, focusing on label word bias and permutation bias as the main forms of bias analysed.

Section 6.1 first discusses the nature of bias and provides examples of human bias and their counterparts in LLMs. Section 6.2 reviews various debiasing approaches to mitigate bias in LLM decisions, including prompting, null-input reweighting and permutation debiasing. Section 6.4 then proposes a practical framework to efficiently achieve permutation debiasing in real-world scenarios and discusses how to assess permutation bias. Finally, section 6.5 outlines the experimental results and discusses the effectiveness of various debiasing approaches and their applicability to NLP tasks.

6.1 Nature of Bias

Bias, whether in humans or machine learning models, has long been an inherent risk of decision-making [315, 214]. Bias describes an inclination or disinclination for an idea, person, or thing, which leads to a perspective that is inaccurate, prejudicial, or unfair. For human decision-making, biases can be attributed to the tendency of the human brain to simplify information processing through a filter of personal experience and preferences [136]. In the context of machine learning, bias can occur when a model makes incorrect associations due to assumptions extrapolated from the training data or the algorithm. This section will first discuss common biases present in human decision-making and then consider biases present in LLMs, drawing parallels between the two.

6.1.1 Social Biases

Social bias refers to the systematic prejudices and preconceived notions that individuals or groups hold about certain social categories or individuals [72, 21]. These biases are often rooted in stereotypes related to race, gender, age, religion, socioeconomic status, and other social identities. These can manifest to various degrees, including implicit biases, where people unconsciously associate certain attributes with specific groups, and explicit biases, where prejudices are consciously endorsed and expressed. Examples of social biases include:

- **Gender bias**: where behaviours, abilities, or roles are assumed based on gender, e.g. associating certain professions with women, e.g. nursing, teaching, etc.
- **Racial bias**: where certain attributes or stereotypes are associated with particular racial or ethnic groups, for example, stereotyping certain races as more or less intelligent, hardworking, or trustworthy.
- **Age bias**: Stereotyping by age, where, for example, older workers are assumed to be less capable of using technology, ignoring actual unbiased evidence.

Stereotyping can also apply to many further traits and while statistical patterns may capture real correlations in historical data, using these patterns in decision-making can perpetuate unfair stereotypes and discrimination. This reflects a fundamental tension between statistical accuracy and ethical decision-making, particularly when correlations stem from historical inequities rather than inherent relationships. While it is important for automatic systems to avoid perpetuating these social biases, addressing these forms of bias is not the main focus of the thesis. Instead, the primary interest is in examining whether models

generalise well and perform the task at hand, rather than ethical concerns. The next subsection will consider cognitive biases, which are more directly related to the main topics of the thesis.

6.1.2 Cognitive Biases

Cognitive biases represent systematic patterns of deviation from rational thinking, which can influence how individuals process information and lead to illogical or biased decisions. Extensive research in psychology has demonstrated that humans are susceptible to cognitive biases [315, 134, 146]. These are often mental shortcuts and heuristics that serve as adaptive mechanisms to make quick decisions in complex environments. However, they can often also lead to errors in judgment and decision-making [88]. Examples of notable cognitive biases are:

- **Availability Heuristic**: the availability bias [315] describes the tendency to overestimate the likelihood of events based on how easily they come to mind. For example, if the news frequently reports plane crashes, an individual might overestimate the probability of dying in a plane accident despite the statistical rarity of such events because these instances are more readily available in memory. This heuristic is based on the assumption that if something can be recalled, it must be important, or at least more important than alternative solutions that are not easily recalled [292].

- **Primacy and Recency Bias**: The primacy and recency biases [224] describe the tendency to give disproportionate importance to information encountered at the beginning or the end of a sequence. For example, in a job interview, the interviewer might remember the first and last candidates more vividly, potentially influencing their hiring decision despite the possible strengths of candidates in the middle of the interviews. These biases stem from the limitations and processes of human memory. Early items are often more memorable due to increased attention and rehearsal (primacy effect), while recent items are readily recalled because they remain in short-term memory and face less interference (recency) [10].

- **The Framing Effect**: The framing effect [316] describes the tendency of individuals to react differently to choices depending on how they are presented, even when the underlying information is identical. For example, when presented with medical treatment options, patients may be more likely to choose a risky procedure if told it has a "90% survival rate" rather than a "10% mortality rate," despite these phrases conveying the same statistical information. This bias stems from an individual's tendency to

interpret information based on how it's presented, where positive framing may lead to risk-averse choices, while negative framing can promote risk-seeking behaviour [170].

Understanding cognitive biases can be useful for understanding and improving decision-making and increase the chance of making rational and accurate decisions. For automatic classification systems (§4.1), it may be beneficial to consider whether such systems are impacted by systematic biases, some of which may mirror existing human cognitive bias.

6.1.3 Biases observed in LLMs

Chapter 5 looked at spurious correlations and shortcuts, where models made spurious decisions due to learning incorrect associations within limited training data. This section focuses on the types of systematic biases that instruction-following LLMs may display when prompted for specific tasks. Although related, instruction-following LLMs can be prompted to perform many general tasks and, therefore, are unlikely to rely on arbitrary task-specific spurious associations. However, these models may still demonstrate systematic bias, and recent studies have demonstrated that LLMs exhibit biases similar to the human cognitive biases discussed previously [1, 133, 178, 288], including:

- **Models preferring words prevalent in the training data**: Previous work demonstrated that LLMs are better at performing numerical reasoning for terms that appeared more frequently during pre-training [271]. Further, when systems are prompted for zero-shot classification, the models typically favour classes represented by common tokens [355]. This type of sensitivity to the frequency of training data is related to the availability heuristic observed in humans.

- **Models preferring the first option in multiple choice answering**: LLMs have shown to be sensitive to the ordering of options when answering multiple choice questions [247, 356], and are biased, often preferring answers in particular positions. This shares similarities with primacy and recency bias, where humans may be biased in remembering information at the start or end.

- **Models being sensitive to the choice of prompt**: Prompt-based LLMs are also sensitive to the prompt template, where works have demonstrated that rephrasing the same prompts can significantly impact task performance [80, 286, 297, 360]. This particular type of bias can be related to the framing effect, where the same information rephrased can change an individual's decision

While these patterns share surface similarities with human cognitive biases, LLMs operate through fundamentally different abstract computational processes than human cognition. These parallels therefore serve more as useful analogies for understanding model behavior rather than suggesting LLMs experience cognitive biases in the same way humans do.

Biases in machine learning applications can have many sources, though they can broadly be divided into two primary categories: algorithmic bias and data bias [12, 214]. Algorithmic bias is bias introduced by the machine learning algorithm itself, such as design choices of the algorithm, selection of features, or model architectures. For example, RNNs [63] may struggle to capture long-range dependencies due to the vanishing gradient problem [121], leading them to overemphasise recent information. On the other hand, data bias is caused by the training data. Models trained on biased data may replicate similar biases [133], though bias can also emerge from the statistical properties of the training data (e.g. frequently encountered facts may result in higher associated language model probabilities).

Having outlined examples of observed biases in LLMs that may mirror cognitive biases, the next section will explore debiasing methods aimed at aligning models more effectively, reducing bias, and improving performance across various tasks.

6.2 Debiasing Predictions

The previous section discussed how instruction-following LLMs can exhibit biases similar to cognitive biases, which may impair their reasoning and, consequently, influence task performance [271, 355, 360]. Given the widespread adoption of these models in real-world applications, addressing these biases and finding ways to ensure more robust and effective use of LLMs is becoming increasingly important. Hence, this section describes various existing methodologies of debiasing instruction-tuned LLMs when deployed as zero-shot classifiers, focusing on the specific biases discussed in Section 6.1.3.

6.2.1 Debiasing by Further Training

Section 3.3.3 introduced the concept of instruction tuning, where model alignment is achieved by curating desired system responses and training the model to generate similar responses. A straightforward method to address an identified bias is to incorporate additional examples into the training dataset [163] or the instruction tuning dataset for further sequential training [180, 239]. For instance, the labelled examples can be counterfactual data points that contradict the bias, e.g. inputs where relying on a gender or cognitive bias results in an incorrect answer [208, 180].

Debiasing via further training [162, 163], though, does have considerable limitations. Firstly, this method requires full white-box access to the model parameters, which may not always be available. Many current capable LLMs, such as GPT-4 [2] and Gemini [6], are closed-sourced and cannot be further trained. Secondly, this reintroduces both data and computational training requirements, which reduces the advantages of having general instruction-following LLMs with good zero/few-shot capabilities. Lastly, foundation models are often available as checkpoints after the language modelling pre-training or as the final parameters after instruction tuning. When models are debiased via further sequential training on a single dataset from the instruction-following checkpoints [239], there is the risk of forgetting catastrophically the LLM's previously acquired abilities [74] (although this risk can be mitigated by parameter efficient fine-tuning [34]).

6.2.2 Debiasing by Prompting

The main drawbacks of the previous debiasing approach were that further training introduces extra computational requirements and is not applicable in black-box scenarios. Prompting has recently emerged as a simple and effective way of adapting models to tasks and domains without modifying any of the system parameters [30, 330]. Hence, prompting to steer LLMs has been proposed as an approach of zero-shot debiasing [79, 173, 231]. In this method, an additional instruction is added to the input instruction, explicitly instructing the LLM not to use a particular bias. For example, in the task of coreference resolution, when debiasing the model against gender bias, one can explicitly add text to the prompt to instruct the model not to use any gender information when making predictions. E.g.:

```
------------------------------------------------------------
Who does "he" refer to in the sentence:
"The physician hired the secretary as he is highly recommended."
Do not use any gender information when making the prediction.
------------------------------------------------------------
```

Further, one can provide in-context examples that explicitly demonstrate debiased responses, for example, directly contradicting the stereotypical gender prior, such that the model may realise that this bias should be avoided. E.g.:

```
------------------------------------------------------------
Who does "she" refer to in the sentence:
"The chief hired the assistant as she needed help."
Ans: Chief
```

```
Who does "he" refer to in the sentence:
"The physician hired the secretary as he is highly recommended."
-------------------------------------------------------------
```

More complicated prompting strategies have been proposed, such as using textual preambles constructed from manually designed templates [231], or eliminating information (e.g. gender-agnostic sentences) which enables the model to reason based on factual knowledge rather than the bias (e.g. gender stereotypes) [173]. However, all these approaches are only as effective as the alignment abilities of the model, as well as the model's interpretation of the discussed bias. Further, if the pre-training data has caused a strong association with the bias, even if prompted, the model may not be able to suppress the bias.

6.2.3 Debiasing by Null-Input Reweighting

For standard classification tasks using prompt-based classifiers [141, 355], various works have demonstrated that systems can be biased to particular design choices. For prompt-based classifiers (previously discussed in Section 4.1.3), given prompt $\mathcal{P}$ and label words $z_{1:K}$, the class probabilities are assumed to be proportional to the language model probability of the label words,

$$\mathtt{P}(\omega_k|\boldsymbol{x},\mathcal{P},z_{1:K};\boldsymbol{\theta}) = \frac{\mathtt{P}_{lm}(z_k|\mathcal{P}(\boldsymbol{x});\boldsymbol{\theta})}{\sum_{j=1}^{K}\mathtt{P}_{lm}(z_j|\mathcal{P}(\boldsymbol{x});\boldsymbol{\theta})} \tag{6.1}$$

$$\hat{y} = \arg\max_{y\in\mathcal{Y}} \mathtt{P}(y|\boldsymbol{x},\mathcal{P},z_{1:K};\boldsymbol{\theta}) \tag{6.2}$$

Existing work has shown that these systems can be sensitive to the wording of the prompt [297], the choice of label words used [286, 355] and the few-shot examples given (if provided) [355]. For example, Zhao et al. [355] consider the few-shot scenario and identify pitfalls of the systems, such as recency bias (predicting answers that appear at the end of the prompt), majority label bias (predicting labels appearing frequently in the in-context examples) and common word bias (predicting words that appear frequently in training data). These biases cause the system to have performance highly sensitive to the choice of prompt $\mathcal{P}$ and label words $z_{1:K}$, with accuracies that range from near-random to state-of-the-art across settings. To address these biases, they proposed "calibrating" the outputs.

Calibration is a post-processing technique that aims to align the model's confidence with its expected accuracy. Given a predictive classifier $\mathtt{P}(y|\boldsymbol{x};\boldsymbol{\theta})$ and dataset $\mathcal{D} = \{\boldsymbol{x}^{(i)}, y^{(i)}\}_{i=1}^{M}$,

perfect calibration [99] is defined when the probability associated with a class ω_k exactly reflects the expected accuracy of the system.

$$\frac{\sum_{i=1}^{M}\sum_{k=1}^{K}\mathbb{1}(\omega_k = y^{(i)})\cdot\mathbb{1}(|\mathtt{P}(\omega_k|\boldsymbol{x}^{(i)};\boldsymbol{\theta})-p|<\varepsilon)}{\sum_{i=1}^{M}\sum_{k=1}^{K}\mathbb{1}(|\mathtt{P}(\omega_k|\boldsymbol{x}^{(i)};\boldsymbol{\theta})-p|<\varepsilon)} = p, \quad \forall p\in[0,1] \tag{6.3}$$

Where ε is a small positive threshold that defines the neighbourhood considered around a given probability p. Alternatively, a less strict definition of calibration is top-1 calibration [50], which occurs when the average confidence matches the overall accuracy of the system:

$$\frac{1}{M}\sum_{i=1}^{M}\mathtt{P}(\hat{y}^{(i)}|\boldsymbol{x}^{(i)};\theta) = \frac{1}{M}\sum_{i=1}^{M}\mathbb{1}(\hat{y}^{(i)} = y^{(i)}) \tag{6.4}$$

Where $\hat{y}^{(i)}$ is the predicted class for the i-th example, i.e. the class with the highest associated probability:

$$\hat{y}^{(i)} = \underset{y\in\mathcal{Y}}{\arg\max}\ \mathtt{P}(y|\boldsymbol{x}^{(i)};\theta) \tag{6.5}$$

To get calibrated outputs, post-processing methods such as matrix scaling [99] are commonly applied. Let $\mathbf{p}\in\mathbb{R}^K$ denote a vector of probabilities, and $\mathbf{q}\in\mathbb{R}^K$ be the output "calibrated" probabilities. Matrix scaling adjusts the output probabilities by applying a simple affine transformation on the logits,

$$\mathbf{q} = \text{softmax}\,(\mathbf{W}\log\mathbf{p}+\boldsymbol{\alpha}) \tag{6.6}$$

Where $\mathbf{W}\in\mathbb{R}^{K\times K}$ and $\boldsymbol{\alpha}\in\mathbb{R}^K$ are learnable parameters that define the post-processing adjustment[1], often optimised to minimise the NLL of the validation dataset [355]. However, in zero-shot and few-shot settings, no labelled data is available. To address this, Zhao et al. [355] propose applying similar affine transformations but using null-inputs to estimate parameters with zero data. Unlike traditional model calibration, which aims to align the model's confidence with its expected accuracy [250, 99], their goal is to minimise implicit bias toward any class, similar to how scales are "calibrated" before use. They set $\mathbf{W}=\mathbf{I}$ and estimate $\boldsymbol{\alpha}$ using a null input, $\emptyset$, which is designed to be uninformative and contain no meaningful classification information (e.g., the empty string, "N/A," or `[MASK]`). Without an informative input, the model is expected to have no implicit class preferences and return a uniform distribution. Therefore, $\boldsymbol{\alpha}$ is set to ensure a uniform output distribution for the null

[1] Setting $\boldsymbol{\alpha}=\mathbf{0}$ and $\mathbf{W}$ to a scalar multiple of the identity matrix, $\mathbf{W}=(1/T)\cdot\mathbf{I}$, results in temperature annealing [250, 99], a popular calibration approach.

input, achieved by setting α_k to the reciprocal of the language model's probability of each label word,

$$\alpha_k = \frac{1}{\mathrm{P}_{lm}(z_k|\mathcal{P}(\emptyset);\boldsymbol{\theta})} \tag{6.7}$$

If applied at inference to new inputs, the debiased output probability distribution takes the form,

$$\mathrm{P}(\omega_k|\boldsymbol{x},\mathcal{P},z_{1:K},\boldsymbol{\alpha};\boldsymbol{\theta}) = \frac{\alpha_k \mathrm{P}(\omega_k|\boldsymbol{x},\mathcal{P},z_{1:K};\boldsymbol{\theta})}{\sum_{j=1}^{K}\alpha_j \mathrm{P}(\omega_j|\boldsymbol{x},\mathcal{P},z_{1:K};\boldsymbol{\theta})} \tag{6.8}$$

6.2.4 Permutation Debiasing

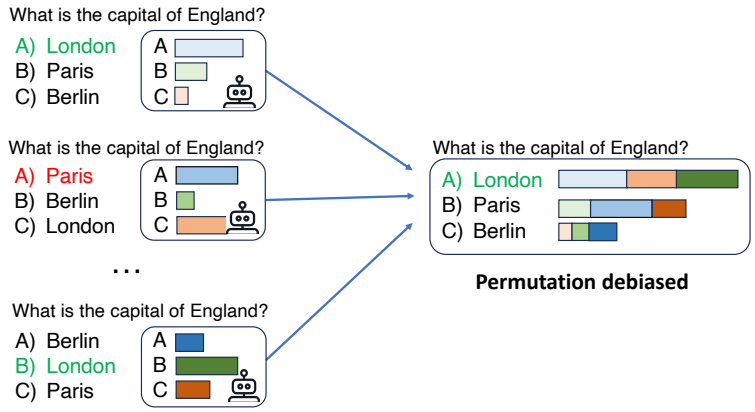


Fig. 6.1 Depiction of Permutation Debiasing. LLM decisions may be sensitive to the input order, which can be debiased by averaging the probabilities associated with all permutations. The diagram depicts the process for 3 permutations, though permutation debiasing uses all 6.

The previous null-input reweighting (§6.2.3) corrects label word bias for prompt-based classifiers. This method can be applied to correct for any form of label word bias, whether that's caused by models favouring common tokens or caused by positional bias, such as where the model prefers the first option in a multiple choice question answer (MCQA) exam. However, particular tasks may have known invariances, leading to bespoke debiasing approaches. For example in MCQA, given an input question $\boldsymbol{q}$, context $\boldsymbol{c}$ and options $\boldsymbol{a}_{1:K}$, let A_σ represent a particular ordering of the options (e.g. $\mathrm{A}_\sigma = \{\boldsymbol{a}_3, \boldsymbol{a}_1, \boldsymbol{a}_2, \boldsymbol{a}_4\}$). The ordering of the options, σ, should not influence the system's selection. Ideally, the system should exhibit permutation invariance where the same probabilities are assigned to the options irrespective

of the input ordering of the options:

$$\mathtt{P}\left(\boldsymbol{a}_k|\boldsymbol{q},\boldsymbol{c},\mathrm{A}_{\sigma_j};\boldsymbol{\theta}\right) = \mathtt{P}\left(\boldsymbol{a}_k|\boldsymbol{q},\boldsymbol{c},\mathrm{A}_{\sigma_m};\boldsymbol{\theta}\right) \qquad \forall j,m \tag{6.9}$$

Note that in equation 6.9, $\mathtt{P}\left(\boldsymbol{a}_k|\boldsymbol{q},\boldsymbol{c},\mathrm{A}_{\sigma_j};\boldsymbol{\theta}\right)$ represents the distribution over the options $\boldsymbol{a}_k$, and *not* the positions ω_k. The prompt-based classifiers may generate the probabilities of the option's position, $\mathtt{P}\left(\omega_k|\boldsymbol{q},\boldsymbol{c},\mathrm{A}_{\sigma_j};\boldsymbol{\theta}\right)$, where $\omega_k \in \{\mathtt{A},\mathtt{B},\mathtt{C},\mathtt{D}\}$, but this can then be converted to the relevant option by selecting the option in the k-th position of A_σ.

To ensure that there is no permutation sensitivity, one can return the ensembled probabilities over all permutations [356]. Since the set of all permutations is the same irrespective of the input permutation, the model will return the same output distribution, $\mathtt{P}(\boldsymbol{a}_k|\boldsymbol{q},\boldsymbol{c},\mathcal{A};\boldsymbol{\theta})$, for all input orderings. The permutation debiased output can be defined as:

$$\mathtt{P}(\boldsymbol{a}_k|\boldsymbol{q},\boldsymbol{c},\mathcal{A};\boldsymbol{\theta}) = \mathbb{E}_\sigma[\mathtt{P}(\boldsymbol{a}_k|\boldsymbol{q},\boldsymbol{c},\mathrm{A}_\sigma;\boldsymbol{\theta})] \tag{6.10}$$

Where $\mathcal{A}$ denotes the set of options, i.e. in an order invariant format. This approach eliminates any permutation sensitivity and positional bias (which will be outlined in greater detail in §6.4.4). However, it requires $K!$ passes through the LLM which could be prohibitively expensive. Approximate approaches such as cyclic permutations [356] can be used, but this still requires K passes, which can be computationally expensive when many options are used.

6.3 Debiasing via Reweighting

The previous section (§6.2) discussed existing methods of debiasing zero-shot classifiers. Section 6.2.3 described that for standard classification tasks, Zhao et al. [355] proposed using a data-free approach of reweighting the output probabilities of prompt-based classifiers. Here, the reweighting parameters $\boldsymbol{\alpha}$ were estimated by looking at the language model probabilities associated with label words when using null inputs. Although the data-free method is highly practical and adds minimal computational overhead, in different scenarios, varying amounts of data might be available to leverage for the task. Therefore, in this section, we propose to generalise the reweighting approach and consider how the method may extend to both supervised and unsupervised scenarios. Further, we will draw a connection between the data-free approach [355] and the unsupervised objective of ensuring that the prior over the classes is uniformly distributed.

For prompt-based classifiers (§4.1.3), given prompt $\mathcal{P}$ and label words $z_{1:K}$, the class probabilities are assumed to be proportional to the language model probability of the label

words,

$$\mathtt{P}(\omega_k|\boldsymbol{x},\mathcal{P},z_{1:K};\boldsymbol{\theta}) = \frac{\mathtt{P}_{lm}(z_k|\mathcal{P}(\boldsymbol{x});\boldsymbol{\theta})}{\sum_{j=1}^{K}\mathtt{P}_{lm}(z_j|\mathcal{P}(\boldsymbol{x});\boldsymbol{\theta})} \tag{6.11}$$

The reweighting method debiases the classifier by reweighting the probabilities associated with each label word. Provided the bias originates from a systematic token-level source (e.g. label word bias, where common words are preferred to rarer words), one may apply a simple rescaling of the class logits to correct for any inadvertent implicit prior. This can be achieved by introducing parameters $\boldsymbol{\alpha} = [\alpha_{1:K}]$ that rescale the class probabilities such that each weight $\alpha_k \in \mathbb{R}^+$ defines a simple scaling of the probabilities for class k,

$$\mathtt{P}(\omega_k|\boldsymbol{x},\mathcal{P},z_{1:K},\boldsymbol{\alpha};\boldsymbol{\theta}) = \frac{\alpha_k\mathtt{P}(\omega_k|\boldsymbol{x},\mathcal{P},z_{1:K};\boldsymbol{\theta})}{\sum_{j=1}^{K}\alpha_j\mathtt{P}(\omega_j|\boldsymbol{x},\mathcal{P},z_{1:K};\boldsymbol{\theta})} \tag{6.12}$$

$$\hat{y} = \underset{y\in\mathcal{Y}}{\operatorname{argmax}}\ \mathtt{P}(y|\boldsymbol{x},\mathcal{P},z_{1:K},\boldsymbol{\alpha};\boldsymbol{\theta}) \tag{6.13}$$

The final decision $\hat{y}$ is the class with the highest probability. The weights $\boldsymbol{\alpha}$ can be determined depending on the objective and the data available for analysis.

6.3.1 Reweighting with Labelled Data

For many classification tasks, the main performance metric is downstream task accuracy. If the model is biased towards a specific class, this will impact performance since decisions will be skewed to that class, increasing the error rate. If labelled data, $\mathcal{D} = \{(\boldsymbol{x}^{(i)}, y^{(i)})\}_{i=1}^{M}$ is available, then one can directly optimise the objective metric (accuracy), which will implicitly correct any large deviations in the assumed class prior. The optimal weights $\boldsymbol{\alpha}^*$ are therefore defined as the weights that maximise the accuracy of the prompt classifier $\mathtt{P}(\omega_k|\boldsymbol{x},\mathcal{P},z_{1:K},\boldsymbol{\alpha};\boldsymbol{\theta})$ over the dataset $\mathcal{D}$,

$$\boldsymbol{\alpha}^* = \underset{\boldsymbol{\alpha}}{\arg\max}\left(\frac{1}{M}\cdot\sum_{i=1}^{M}\mathbb{1}\left(y^{(i)} = \underset{y\in\mathcal{Y}}{\arg\max}\ \mathtt{P}(y|\boldsymbol{x}^{(i)},\mathcal{P},z_{1:K},\boldsymbol{\alpha};\boldsymbol{\theta})\right)\right) \tag{6.14}$$

Unlike supervised fine-tuning (§4.1.2), reweighting does not update any model parameters and is also applicable in grey-box scenarios where only output logits are available. It also adds minimal computational overhead, as the weights simply rescale the output probabilities. However, requiring labelled data limits the benefits of zero-shot prompt-based classifiers, which do not otherwise require task-specific labelled data. Additionally, with access to labelled data $\mathcal{D}$, one could directly fine-tune the model to achieve greater task performance.

6.3.2 Reweighting with Unlabelled Data

To reduce the data requirements required from supervised optimal reweighting, one can instead consider unsupervised approaches to determine the values of $\boldsymbol{\alpha}$. Instead of optimising for accuracy, one can find the values $\bar{\boldsymbol{\alpha}}$ that ensure that the classifier is unbiased, such that the class prior $\mathtt{P}(\omega_k|\mathcal{P}, z_{1:K}, \boldsymbol{\alpha})$ matches the true prior $\mathtt{P}(\omega_k)$. The system's class prior can be estimated using a Monte Carlo approximation,

$$\mathtt{P}(\omega_k|\mathcal{P}, z_{1:K}, \boldsymbol{\alpha}; \boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{x}}\left[\mathtt{P}(\omega_k|\boldsymbol{x}, \mathcal{P}, z_{1:K}, \boldsymbol{\alpha}; \boldsymbol{\theta})\right] \tag{6.15}$$

$$\approx \frac{1}{M}\sum_{i=1}^{M} \mathtt{P}(\omega_k|\boldsymbol{x}^{(i)}, \mathcal{P}, z_{1:K}, \boldsymbol{\alpha}; \boldsymbol{\theta}) \tag{6.16}$$

While the true prior of the task can be estimated using the labelled dataset:

$$\mathtt{P}(\omega_k) \approx \mathtt{P}(\omega_k|\mathcal{D}) = \frac{1}{M}\sum_{i=1}^{M} \mathbb{1}(\omega_k = y^{(i)}) \tag{6.17}$$

However, to avoid data requirements, instead of calculating the true prior using supervised data, one can assume that there should not be any expected class bias and that the probabilities should be equal over all classes,

$$\mathtt{P}(\omega_k) \approx \frac{1}{K} \tag{6.18}$$

Prior matching is then performed by finding the weights $\boldsymbol{\alpha}$ which minimise the discrepancy between the systems' class prior and the assumed prior,

$$\bar{\boldsymbol{\alpha}} = \arg\min_{\boldsymbol{\alpha}} \sum_{k=1}^{K} |\mathtt{P}(\omega_k|\mathcal{P}, z_{1:K}, \boldsymbol{\alpha}; \boldsymbol{\theta}) - \mathtt{P}(\omega_k)| \tag{6.19}$$

For any prior, an optimal solution $\bar{\boldsymbol{\alpha}}$ which exactly matches the distribution exists, though with one degree of freedom which can be constrained by setting $\alpha_1 = 1$. By assuming a uniform prior, this approach becomes unsupervised and uses no class labels, only requiring text inputs, i.e. an unlabelled dataset $\mathcal{D}_x = \{\boldsymbol{x}^{(i)}\}_{i=1}^{M}$. This approach offers practical application in scenarios where unlabelled in-domain data is available or when inference is done over a large batch of data simultaneously (which is often the case for NLP test sets).

6.3.3 Reweighting with No Data

Although reweighting using unlabelled data avoids the need for human annotations, the dependence on unlabelled dataset $\mathcal{D}_x$ remains a drawback as it requires access to in-domain task data. The most practical approach would be a completely data-free solution with no data requirements. Consider the expression of the system's class prior marginalised over the inputs $\boldsymbol{x}$,

$$\mathtt{P}(\omega_k|\mathcal{P}, z_{1:K}, \boldsymbol{\alpha}; \boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{x}}\{\mathtt{P}(\omega_k|\boldsymbol{x}, \mathcal{P}, z_{1:K}, \boldsymbol{\alpha}; \boldsymbol{\theta})\} \tag{6.20}$$

$$= \mathbb{E}_{\boldsymbol{x}}\left[\frac{\alpha_k \mathtt{P}_{lm}(z_k|\mathcal{P}(\boldsymbol{x}); \boldsymbol{\theta})}{Z(\boldsymbol{x}, \mathcal{P}, z_{1:K}, \boldsymbol{\alpha}, \boldsymbol{\theta})}\right] \tag{6.21}$$

where $Z(\boldsymbol{x}, \mathcal{P}, z_{1:K}, \boldsymbol{\alpha}, \boldsymbol{\theta}) = \sum_{j=1}^{K} \alpha_j \mathtt{P}_{lm}(z_j|\mathcal{P}(\boldsymbol{x}); \boldsymbol{\theta})$. This expression can be approximated using the Taylor series of the expectation of a ratio [232]. The Taylor series approximation of a ratio is given as:

$$\mathbb{E}_{\boldsymbol{x}}\left[\frac{f(\boldsymbol{x})}{g(\boldsymbol{x})}\right] \approx \frac{\mathbb{E}_{\boldsymbol{x}}[f(\boldsymbol{x})]}{\mathbb{E}_{\boldsymbol{x}}[g(\boldsymbol{x})]} - \frac{\mathrm{cov}(f(\boldsymbol{x}), g(\boldsymbol{x}))}{\mathbb{E}_{\boldsymbol{x}}[g(\boldsymbol{x})]^2} + \frac{\mathbb{E}_{\boldsymbol{x}}[f(\boldsymbol{x})]\,\mathrm{var}(g(\boldsymbol{x}))}{\mathbb{E}_{\boldsymbol{x}}[g(\boldsymbol{x})]^3} \approx \frac{\mathbb{E}_{\boldsymbol{x}}[f(\boldsymbol{x})]}{\mathbb{E}_{\boldsymbol{x}}[g(\boldsymbol{x})]} \tag{6.22}$$

Where the final step assumes that the covariance between $f(\boldsymbol{x})$ and $g(\boldsymbol{x})$ is small and the variance of $g(\boldsymbol{x})$ is not too large. Therefore, the class prior of the debiased classifier $\mathtt{P}(\omega_k|\mathcal{P}, z_{1:K}, \boldsymbol{\alpha}; \boldsymbol{\theta})$ can be approximated as:

$$\mathtt{P}(\omega_k|\mathcal{P}, z_{1:K}, \boldsymbol{\alpha}; \boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{x}}\left[\frac{\alpha_k \mathtt{P}_{lm}(z_k|\mathcal{P}(\boldsymbol{x}); \boldsymbol{\theta})}{Z(\boldsymbol{x}, \mathcal{P}, z_{1:K}, \boldsymbol{\alpha}, \boldsymbol{\theta})}\right] \tag{6.23}$$

$$\approx \frac{\mathbb{E}_{\boldsymbol{x}}[\alpha_k \mathtt{P}_{lm}(z_k|\mathcal{P}(\boldsymbol{x}); \boldsymbol{\theta})]}{\mathbb{E}_{\boldsymbol{x}}[Z(\boldsymbol{x}, \mathcal{P}, z_{1:K}, \boldsymbol{\alpha}, \boldsymbol{\theta})]} \tag{6.24}$$

$$\approx \frac{\alpha_k \mathtt{P}_{lm}(z_k|\mathcal{P}; \boldsymbol{\theta})}{Z(\mathcal{P}, z_{1:K}, \boldsymbol{\alpha}, \boldsymbol{\theta})} \tag{6.25}$$

Where the final line, Equation 6.25, uses the distributions marginalised over the inputs $\boldsymbol{x}$,

$$\mathtt{P}_{lm}(z_k|\mathcal{P}; \boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{x}}[\mathtt{P}_{lm}(z_k|\mathcal{P}(\boldsymbol{x}); \boldsymbol{\theta})] \qquad Z(\mathcal{P}, z_{1:K}, \boldsymbol{\alpha}, \boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{x}}[Z(\boldsymbol{x}, \mathcal{P}, z_{1:K}, \boldsymbol{\alpha}, \boldsymbol{\theta})] \tag{6.26}$$

Equating the system's class prior, $\mathtt{P}(\omega_k|\mathcal{P}, z_{1:K}, \boldsymbol{\alpha}; \boldsymbol{\theta})$, with the true prior based on the supervised dataset, $\mathtt{P}(\omega_k|\mathcal{D})$, yields an expression to solve for an approximation of $\bar{\alpha}_k$:

$$\mathtt{P}(\omega_k|\mathcal{P}, z_{1:K}, \bar{\boldsymbol{\alpha}}; \boldsymbol{\theta}) = \mathtt{P}(\omega_k|\mathcal{D}) \tag{6.27}$$

$$\frac{\bar{\alpha}_k \mathtt{P}_{lm}(z_k|\mathcal{P}, z_{1:K}; \boldsymbol{\theta})}{Z(\mathcal{P}, z_{1:K}, \bar{\boldsymbol{\alpha}}, \boldsymbol{\theta})} \approx \mathtt{P}(\omega_k|\mathcal{D}) \tag{6.28}$$

$$\bar{\alpha}_k \approx \frac{Z(\mathcal{P}, z_{1:K}, \bar{\boldsymbol{\alpha}}, \boldsymbol{\theta}) \cdot \mathtt{P}(\omega_k|\mathcal{D})}{\mathtt{P}_{lm}(z_k|\mathcal{P}, z_{1:K}; \boldsymbol{\theta})} \tag{6.29}$$

However, the purpose of this derivation is to motivate a zero-data method, and it is therefore impractical to require labelled data to determine the prior $\mathtt{P}(\omega_k|\mathcal{D})$. In many scenarios, there should be no inherent preference towards any class, and hence one can assume that the true prior, $\mathtt{P}(\omega_k|\mathcal{D})$, is uniform. Additionally, due to the normalisation, scaling all weights by a constant multiple will not influence the probabilities, and therefore one can derive the analytical approximation,

$$\bar{\alpha}_k \approx \frac{1}{\mathtt{P}_{lm}(z_k|\mathcal{P}, z_{1:K}; \boldsymbol{\theta})} \tag{6.30}$$

This illustrates that the null-input approximation done by Zhao et al. [355] relates to the unsupervised prior-matching method by making the assumption that,

$$\mathtt{P}(z_k|\mathcal{P}, z_{1:K}; \boldsymbol{\theta}) \approx \mathtt{P}_{lm}(z_k|\mathcal{P}(\emptyset); \boldsymbol{\theta}) \tag{6.31}$$

I.e., the marginalised language model probabilities can be approximated by the language model probabilities associated with the null input. This completes the connection between the unsupervised reweighting that we propose with the data-free method proposed by [355] (§6.2.3), where the data-free weights can be estimated following:

$$\bar{\alpha}_k \approx \frac{1}{\mathtt{P}_{lm}(w_k|\mathcal{P}(\emptyset); \boldsymbol{\theta})} \tag{6.32}$$

6.4 Efficient Positional Debiasing

The reweighting debiasing strategy (§6.3) requires only a fixed, small overhead to estimate the weights $\boldsymbol{\alpha}$, and thus can be applied without significantly increasing computational complexity. In contrast, permutation debiasing (§6.2.4), which averages probabilities across all input permutations, incurs inference costs that scale with $K!$, making it prohibitively expensive as K grows. Therefore, this section explores how permutation debiasing can be adapted to be more computationally feasible while maintaining its effectiveness.

This section introduces the Teacher-Student Debiasing Distillation (TSDD) framework, an unsupervised permutation debiasing scheme designed for efficient inference. The framework prioritises low inference costs in real-world applications, while training computational costs are not a primary concern. Unlike further-training debiasing methods that rely on annotated debiased examples (§6.2.1), TSDD leverages the computationally expensive permutation debiasing to automatically generate unbiased decisions, which are then distilled into a more efficient student model. Two variations of compact student models are proposed: a basic knowledge-distilled student and an error-correction student.

6.4.1 Distillation

For a selected task $\mathcal{T}$, the goal of the TSDD framework is to train a compact, inference-efficient student to emulate the predictions of a debiased teacher. The most inference-efficient approach is to *distil* the knowledge [117] of a debiased teacher distribution parametrised by $\boldsymbol{\theta}$ onto a small non-autoregressive student parametrised by weights $\hat{\boldsymbol{\theta}}_s$. Given any input question $\boldsymbol{q}$, context $\boldsymbol{c}$ and ordered options A_σ, the student can be designed to model the debiased teacher distribution,

$$\mathtt{P}(\boldsymbol{a}_k|\boldsymbol{q},\boldsymbol{c},\mathrm{A}_{\sigma_j};\hat{\boldsymbol{\theta}}_s) \approx \mathbb{E}_\sigma[\mathtt{P}(\boldsymbol{a}_k|\boldsymbol{q},\boldsymbol{c},\mathrm{A}_\sigma;\boldsymbol{\theta})] \quad \forall j \tag{6.33}$$

i.e., irrespective of the ordering of the options presented to the student, the student should predict distributions consistent with those from the debiased teacher. This is achieved by minimising the KL divergence between the student and teacher distributions:

$$\mathcal{L}_d(\boldsymbol{\theta}_s) = \mathbb{E}_{\{\boldsymbol{q},\boldsymbol{c},\mathcal{A}\},\sigma}\Big[\mathtt{KL}\big(\mathtt{P}(\boldsymbol{a}|\boldsymbol{q},\boldsymbol{c},\mathcal{A};\boldsymbol{\theta})||\mathtt{P}(\boldsymbol{a}|\boldsymbol{q},\boldsymbol{c},\mathrm{A}_\sigma;\boldsymbol{\theta}_s)\big)\Big] \tag{6.34}$$

$$\hat{\boldsymbol{\theta}}_s = \arg\min_{\boldsymbol{\theta}_s} \mathcal{L}_d(\boldsymbol{\theta}_s) \tag{6.35}$$

where as defined in Equation 6.10, $\mathtt{P}(\boldsymbol{a}|\boldsymbol{q},\boldsymbol{c},\mathcal{A};\boldsymbol{\theta})$ is the permutation debiased output of the teacher,

$$\mathtt{P}(\boldsymbol{a}_k|\boldsymbol{q},\boldsymbol{c},\mathcal{A};\boldsymbol{\theta}) = \mathbb{E}_\sigma[\mathtt{P}(\boldsymbol{a}_k|\boldsymbol{q},\boldsymbol{c},\mathrm{A}_\sigma;\boldsymbol{\theta})] \tag{6.36}$$

During training, the debiased teacher probabilities are computed, which for full permutation debiasing requires $K!$ white-box calls for each data point. However, once the student has been trained, it can be used independently of the original LLM and be significantly faster. Although this process requires access to the output probabilities, which is usually only available with white-box access, Section 6.4.3 will discuss how to apply teacher-student training in black-box settings.

6.4.2 Error Correction

While the previous student-distillation approach is highly inference efficient and only requires a single forward pass through the student system, the capacity of a small proxy system may be insufficient when applied to complex tasks. Therefore, as an extension to vanilla distillation, we consider having a student that performs *error-correction*. Instead of directly performing the task given the input, the error-correction student receives a sample $\tilde{\boldsymbol{a}}$ from the teacher system $\mathtt{P}(\boldsymbol{a}|\boldsymbol{c},\boldsymbol{q},\mathrm{A}_\sigma;\boldsymbol{\theta})$, which may be biased, and this sample is also used to predict the debiased teacher distribution. The student therefore has form $\mathtt{P}(\boldsymbol{a}|\boldsymbol{q},\boldsymbol{c},\mathrm{A}_\sigma,\tilde{\boldsymbol{a}};\hat{\boldsymbol{\theta}}_s)$, and similarly aims to return the debiased distribution such that:

$$\tilde{\boldsymbol{a}} \sim \mathtt{P}(\boldsymbol{a}|\boldsymbol{c},\boldsymbol{q},\mathrm{A}_\sigma;\boldsymbol{\theta}) \tag{6.37}$$

$$\mathtt{P}(\boldsymbol{a}|\boldsymbol{q},\boldsymbol{c},\mathrm{A}_\sigma,\tilde{\boldsymbol{a}};\hat{\boldsymbol{\theta}}_s) \approx \mathbb{E}_\sigma[\mathtt{P}(\boldsymbol{a}_k|\boldsymbol{q},\boldsymbol{c},\mathrm{A}_\sigma;\boldsymbol{\theta})] \tag{6.38}$$

During training, the student can now use the prediction from the (possibly) biased teacher and aim to correct for any bias present in the system's decision. The training loss is again the expected KL divergence between the student proxy and the debiased teacher, but now, with the expectation also over sampled teacher predictions,

$$\mathcal{L}_e(\boldsymbol{\theta}_s) = \mathbb{E}_{\{\boldsymbol{q},\boldsymbol{c},\mathcal{A}\},\sigma}\left[\mathbb{E}_{\tilde{\boldsymbol{a}}\sim\mathtt{P}(\boldsymbol{a}|\boldsymbol{q},\boldsymbol{c},\mathrm{A}_\sigma;\boldsymbol{\theta})}\left[\mathtt{KL}\big(\mathtt{P}(\boldsymbol{a}|\boldsymbol{q},\boldsymbol{c},\mathcal{A};\boldsymbol{\theta})||\mathtt{P}(\boldsymbol{a}|\boldsymbol{q},\boldsymbol{c},\mathrm{A}_\sigma,\tilde{\boldsymbol{a}};\boldsymbol{\theta}_s)\big)\right]\right] \tag{6.39}$$

$$\hat{\boldsymbol{\theta}}_s = \arg\min_{\boldsymbol{\theta}_s} \mathcal{L}_e(\boldsymbol{\theta}_s) \tag{6.40}$$

At inference time, the student now requires a single black-box sample from the teacher to approximate the teacher's full debiased distribution. If the student is much smaller and more computationally efficient than the original system, this can be achieved with minimal increase in computational costs.

6.4.3 Black-Box Considerations

The distillation and error correction approaches outlined in Sections 6.4.1 and 6.4.2 have assumed access to the debiased teacher distribution $\mathtt{P}(\boldsymbol{a}|\boldsymbol{x},\mathcal{A};\boldsymbol{\theta})$ during training. This requires access to the output logits, which are only available in white-box or grey-box settings. However, some LLMs may be closed-source and not provide access to the output probability distributions. In such black-box settings, a hierarchical Monte Carlo approximation of the debiased teacher can be used, where for each permutation of the options, multiple decisions

are stochastically sampled from the LLM,

$$\sigma \sim \{\sigma_1, \sigma_2, ..., \sigma_{K!}\} \tag{6.41}$$

$$\tilde{\boldsymbol{a}}^{(j)} \sim \mathtt{P}(\boldsymbol{a}|\boldsymbol{c}, \boldsymbol{q}, \mathrm{A}_\sigma; \boldsymbol{\theta}) \tag{6.42}$$

First, a random permutation of the answers, A_σ, is selected, followed by sampling from the resulting biased distribution. In expectation, this process recovers the debiased distribution, allowing for a sample-based approximation:

$$\mathtt{P}(\boldsymbol{a}_k|\boldsymbol{q}, \boldsymbol{c}, \mathcal{A}; \boldsymbol{\theta}) = \mathbb{E}_\sigma \left[\mathtt{P}(\boldsymbol{a}_k|\boldsymbol{q}, \boldsymbol{c}, \mathrm{A}_\sigma)\right] \tag{6.43}$$

$$\approx \frac{1}{M} \sum_{j=1}^{M} \mathbb{1}(\tilde{\boldsymbol{a}}^{(j)} = \boldsymbol{a}_k) \tag{6.44}$$

Where M is the number of draws taken from the (possibly) biased teacher distribution. Furthermore, the Monte Carlo approximation for the knowledge distillation criteria becomes:

$$\mathcal{L}_d(\boldsymbol{\theta}_s) = \mathbb{E}_{\{\boldsymbol{q}, \boldsymbol{c}, \mathcal{A}\}, \sigma} \Big[\mathtt{KL}\big(\mathtt{P}(\boldsymbol{a}|\boldsymbol{q}, \boldsymbol{c}, \mathcal{A}; \boldsymbol{\theta}) || \mathtt{P}(\boldsymbol{a}|\boldsymbol{q}, \boldsymbol{c}, \mathrm{A}_\sigma; \boldsymbol{\theta}_s)\big) \Big] \tag{6.45}$$

$$\stackrel{c}{=} \mathbb{E}_{\{\boldsymbol{q}, \boldsymbol{c}, \mathcal{A}\}, \sigma} \Big[\mathbb{E}_{\boldsymbol{a} \sim \mathtt{P}(\boldsymbol{a}|\boldsymbol{q}, \boldsymbol{c}, \mathcal{A})} \big[-\ln \mathtt{P}(\boldsymbol{a}|\boldsymbol{q}, \boldsymbol{c}, \mathrm{A}_\sigma; \boldsymbol{\theta}_s)\big) \big] \Big] \tag{6.46}$$

$$\approx \mathbb{E}_{\{\boldsymbol{q}, \boldsymbol{c}, \mathcal{A}\}, \sigma} \Big[\frac{1}{M} \cdot \sum_{j=1}^{M} -\ln \mathtt{P}(\tilde{\boldsymbol{a}}^{(j)}|\boldsymbol{q}, \boldsymbol{c}, \mathrm{A}_\sigma; \boldsymbol{\theta}_s) \Big] \tag{6.47}$$

This demonstrates how the student models can be trained to emulate the debiased teacher distribution with only black-box access to the teacher LLM.

6.4.4 Assessing Bias

To evaluate whether a system exhibits a particular bias and to validate the effectiveness of debiasing approaches, a systematic method for assessing bias is required. This section proposes metrics for MCQA tasks that measure whether systems are sensitive to the input ordering of options.

Permutation Sensitivity: The distribution over possible answers should be unaffected by the input ordering. Therefore, to quantify the *permutation sensitivity* of a model to changes in the input order, one can measure the expected divergence between the distributions resulting

from any two possible permutations A_{σ_j}, A_{σ_m}:

$$\mathtt{ps}(\boldsymbol{\theta}) = \mathbb{E}_{\{\boldsymbol{q},\boldsymbol{c},\mathcal{A}\}}\left[\mathbb{E}_{\sigma_{j,m}}\left[\frac{1}{2}\sum_{k=1}^{K}\left|\mathtt{P}\left(\boldsymbol{a}_k|\boldsymbol{q},\boldsymbol{c},\mathrm{A}_{\sigma_j};\boldsymbol{\theta}\right) - \mathtt{P}\left(\boldsymbol{a}_k|\boldsymbol{q},\boldsymbol{c},\mathrm{A}_{\sigma_m};\boldsymbol{\theta}\right)\right|\right]\right] \tag{6.48}$$

The total variation distance [169] is used as the distance metric due to its bounded nature, symmetry, and intuitive interpretability. Other metrics, such as the KL divergence, may be unbounded and, if used, may provide metrics that are disproportionately influenced by individual outlier samples that have high divergence.

Positional Bias: A possible cause for permutation sensitivity may be systematic bias, where the most obvious form of bias is a global preference for a specific option. To measure if there is any systematic preference for certain positions irrespective of the option, one can look at the average probability mass associated with each position, $\omega_k \in \{\mathtt{A},\mathtt{B},\mathtt{C},\mathtt{D}\}$, over all permutations, σ:

$$\mathtt{P}(\omega_k;\boldsymbol{\theta}) = \mathbb{E}_{\{\boldsymbol{q},\boldsymbol{c},\mathcal{A}\},\sigma}[\mathtt{P}(\omega_k|\boldsymbol{q},\boldsymbol{c},\mathrm{A}_\sigma;\boldsymbol{\theta})] \tag{6.49}$$

This marginalised distribution looks at the probability of the k-th option irrespective of how the options have been presented. If this positional distribution is non-uniform, the natural interpretation is that the underlying LLM has a preference towards a particular position and is biased. Therefore, a measure of *positional bias* can be defined as the divergence between the positional and uniform distribution:

$$\mathtt{pb}(\boldsymbol{\theta}) = \frac{1}{2}\sum_{k=1}^{K}\left|\mathtt{P}(\omega_k;\boldsymbol{\theta}) - \frac{1}{K}\right| \tag{6.50}$$

Note that positional bias is more relaxed than permutation sensitivity; a system that is permutation insensitive guarantees having no positional bias, while the reverse is not true.

6.5 Experiments

The experiments in this section focus on identifying whether zero-shot prompted LLMs display bias and whether the bias can be mitigated with post-processing techniques. The first set of experiments examines zero-shot classifiers for general text classification tasks. Here, the aim is to determine the effectiveness of zero-shot text classifiers, whether they are susceptible to label-word bias, and whether reweighting (§6.3) can alleviate such biases. The second set of experiments then examines the abilities of zero-shot LLMs when applied to MCQA tasks. The analysis aims to identify whether prompted LLMs exhibit positional bias

and whether weight debiasing (§6.3) and/or permutation debiasing (§6.2.4) mitigates such biases. Lastly, we examine whether the student-teacher distillation framework (§6.4) can be used to enable practical permutation debiasing.

Datasets

For the analysis, experiments are conducted on several standard NLP datasets across the tasks discussed in Section 4.2.1, covering sentiment classification, natural language inference (NLI), paraphrase detection, and multiple choice question answering (MCQA):

- For sentiment classification, **IMDb** [200], Rotten Tomatoes (**RT**) [237], and **Amazon** polarity [126] are considered. All datasets assess binary sentiment classification, where each review has to be classified as positive sentiment or negative sentiment (with the datasets previously introduced in Section 5.5.1).
- For NLI, the Stanford Natural Language Inference (**SNLI**) [26] and Multi-Genre Natural Language Inference (**MNLI**) [336] are used. SNLI contains pairs of sentences where the relationship is classified as entailment, contradiction, or neutral, using human-written English sentences. MNLI is an extension of SNLI, where sentence pairs are collected from a diverse range of domains and genres, covering both spoken and written text.
- For paraphrase detection, the Quora Question Pairs (**QQP**) [260] dataset is used. The objective of the task is to identify duplicate questions, with binary labels indicating whether the pair of questions are semantically equivalent.
- For MCQA, we use **RACE++** [160, 175], **CosmosQA** [130], **ReClor** [343] and **ARC-EASY** [46]. The first three datasets are multiple-choice reading comprehension exams (previously introduced in Section 5.5.2), while ARC-EASY contains multiple-choice science exam questions drawn from grade-school level assessments with no contextual passage.

As the focus of this chapter is on zero-shot classification, the models are not trained on any training data and are only evaluated on each dataset's test examples. For both QQP and Amazon, we randomly sample 10k examples from the test set (as the original test sets have around 400k examples). Table 6.1 presents the test set statistics for the datasets. Most datasets are relatively balanced over the classes, with the exception of QQP, where the number of non-paraphrases is double the amount of paraphrases. Datasets such as RACE++ and MNLI also exhibit mild imbalances.

Dataset	Test Examples	K	Class Distribution (%)
IMDb	25,000	2	50.0% / 50.0%
RT	1,840	2	50.0% / 50.0%
Amazon	10,000	2	50.0% / 50.0%
QQP	10,000	2	63.1% / 36.9%
SNLI	10,000	3	34.3% / 32.7% / 33.0%
MNLI	9,820	3	35.4% / 31.9% / 32.7%
RACE++	5,642	4	21.8% / 25.9% / 27.2% / 25.2%
CosmosQA	2,985	4	25.1% / 25.0% / 25.0% / 24.9%
ReClor	1000	4	25.2% / 25.2% / 24.9% / 24.6%
ARC-EASY	2,376	4	25.1% / 24.6% / 26.7% / 23.6%

Table 6.1 Data statistics for the test sets, presenting a number of test examples, number of classes (K), and class distributions (%).

Instruction-Following Models

We select and investigate two different open-source LLM families. The first is FlanT5 [42], an encoder-decoder T5 [266] system that has been further instruction-tuned (§3.3.3) on the FLAN collection [193], a diverse set of over 1,800 NLP tasks. The second is Llama2-chat [314], a decoder-only pre-trained transformer (§3.2.2) that is further instruction-tuned (§3.3.3) on 27,540 annotations of multi-turn dialogue, specifically designed to improve its conversational capabilities. As this section examines zero-shot prompting, 'Llama2' will specifically refer to the chat instruction-following variant and not the pre-trained language model. We examine various sizes of both FlanT5 (220M, 770M, 3B, 11B) and Llama2 (7B, 13B). Although numerous other NLP foundation models are available (§3.2), FlanT5 and Llama represent some of the most capable open-source instruction-following LLMs, particularly at the time of conducting experiments.

6.5.1 Debiasing by Reweighting for Text Classification

The experiments in this section will examine zero-shot text classifiers for sentiment analysis, NLI and paraphrase detection, investigating the system's robustness to design choices such as prompts and label words. As discussed in Section 6.1.3, LLMs can exhibit biases that can cause performance degradation. In this section, we focus on label word bias and analyse how the choice of label words may affect downstream performance. In theory, synonyms should be interchangeable and not affect performance. However, LLMs are pre-trained with

language modelling tasks over large unsupervised datasets (§3.1) and therefore less common words, such as "phenomenal", may have lower likelihoods than common words, such as "good", even if both equally fit the context. To investigate approaches to mitigate label word bias, we apply weight reweighting (§6.3) with the following three methods considered:

- **optimal (§6.3.1)**, where supervised data is used to find the weights that maximise the task accuracy. This approach requires labelled data and, therefore, is less practical but serves as an upper bound of the performance possible by weight reweighting.
- **prior-match (§6.3.2)**, where unsupervised data is used to find weights that ensure a balanced class prior over all system predictions. The unlabelled data is the input test text data without using any labels.
- **null-norm (§6.3.3)**, which is a zero-resource approach that requires no data, originally proposed by Zhao et al. [355]. In this method, the weights are set to the reciprocal of the LLM probability associated with each label word given the null input $\emptyset$, where we use the empty string as the null input.

To determine the LLMs label word bias, we consider a set of synonyms for each class in a given task and observe the performance variations when a random combination of label words is used. Table 6.2 presents the prompts evaluated for sentiment classification and NLI, while Table 6.3 shows the respective label words (i.e. sets of synonyms) used. By measuring the performance fluctuations across different prompts and label words, we can determine the level of label bias present, as systems with no label-word bias should have similar performance across all synonym permutations. Additionally, by evaluating the increase in average performance after applying the different reweighting debiasing schemes, we can identify whether our approaches effectively debias the systems and yield more robust performance across prompt-classifier settings.

Sentiment Classification Prompts	Textual Entailment Prompts
classify the following review:	is the second text an entailment of the first text?
how was the movie?	does the second text directly follow from the first text?
which word best describes the text?	are the texts related?
what is the sentiment?	does text 1 imply text 2?
what is the reviewer's verdict?	can text 2 be logically derived from text 1?
is the following movie good or bad?	does the hypothesis logically follow the premise?

Table 6.2 Prompts used for sentiment classification and textual entailment.

Class	Synonyms				
Sentiment Classification					
Positive	good	great	amazing	fantastic	positive
Negative	bad	terrible	poor	horrible	negative
Textual Entailment					
Entailment	yes	correct	follows	yeah	
Neutral	maybe	unclear	potentially	neutral	
Contradiction	no	incorrect	contradiction	nope	

Table 6.3 Selected label words used for sentiment classification and textual entailment tasks, representing synonyms for each category.

Class Bias

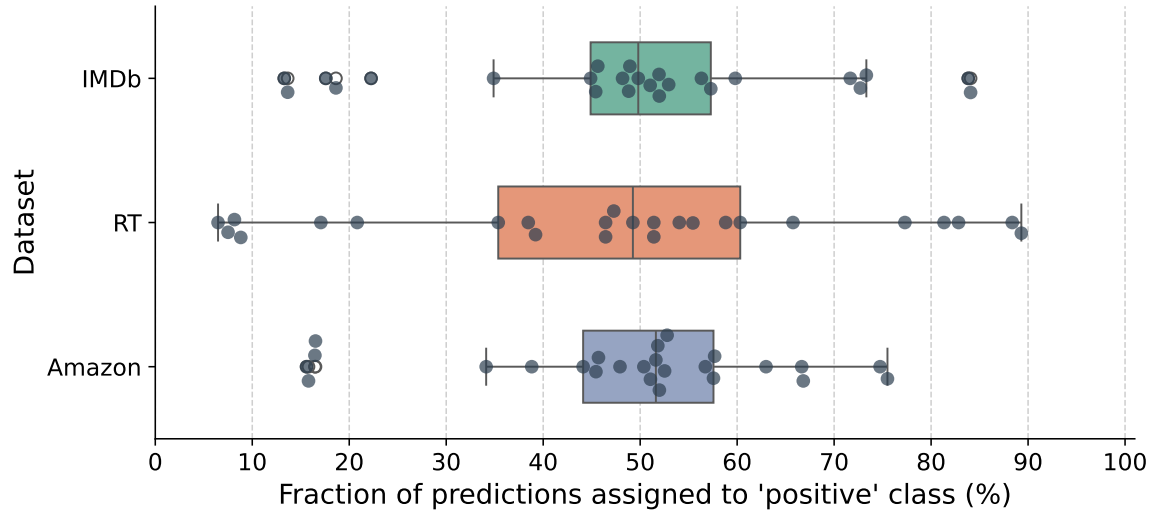


Fig. 6.2 Boxplots of the class bias of FlanT5-770M when prompted for zero-shot sentiment classification. Each point represents a particular choice of label words, where the x-axis denotes the proportion of predictions assigned to the positive class. The datasets are balanced, so any deviation from 50% represents a class bias.

The first experiment examines whether prompt-based classifiers are sensitive to the choice of label words and if this results in class bias where common words have higher associated likelihoods. Figure 6.2 presents box plots showing the class distributions of predictions for FlanT5-770M when applied to sentiment classification tasks. As the sentiment classification datasets are all fully balanced, we would expect half of the system's predictions to be for

the positive class and the other half for the negative class. A class bias would cause the fraction of positive predictions to deviate significantly from 50%. Figure 6.2 demonstrates that selecting a particular permutation of label words (e.g., "good" and "horrible") can lead to considerable class bias, where one class has a significantly higher prior probability than the other. For instance, in the rotten-tomatoes (RT) dataset, certain selected label words result in 90% of the samples being predicted as positive and only 10% as negative. The large spread in implicit priors indicates that label word bias is likely to be a significant factor that may substantially impact the accuracy of the prompt-based classifier.

Performance improvement by Reweighting

Method	Inputs	Labels	IMDb	RT	Amazon	SNLI	MNLI	QQP
	FlanT5-770M							
baseline	✗	✗	85.4±12.7	78.8±14.0	86.0±13.8	45.2±13.7	43.5±11.3	65.4±14.0
null-norm	✗	✗	92.1±3.2	89.1±3.8	95.0±1.8	75.2±10.4	66.1±9.7	77.4±6.6
prior-match	✓	✗	93.1±3.3	90.9±1.6	96.0±0.8	78.5±9.3	69.8±9.7	79.1±2.4
optimal	✓	✓	93.5±2.7	91.2±1.5	96.1±0.7	79.4±8.2	70.8±8.6	82.3±2.8
	FlanT5-220M							
baseline	✗	✗	82.1±11.1	70.1±11.7	83.4±12.8	37.4±6.1	37.0±4.3	52.5±11.4
null-norm	✗	✗	87.5±3.2	78.5±4.8	91.1±2.3	41.8±3.8	40.2±4.1	53.9±10.2
prior-match	✓	✗	89.1±2.4	80.8±2.9	92.0±1.3	44.7±6.3	41.8±3.8	58.5±5.5
optimal	✓	✓	89.3±2.0	81.2±2.9	92.1±1.4	47.6±5.9	43.5±3.7	65.3±2.9
	Llama2-7B							
baseline	✗	✗	85.8±8.7	78.4±10.3	86.4±10.3	35.1±2.7	36.9±3.2	51.0±11.6
null-norm	✗	✗	87.4±6.9	83.2±6.6	90.7±6.4	37.9±5.4	39.4±3.7	51.8±8.4
prior-match	✓	✗	90.5±3.1	86.3±3.7	93.1±2.4	39.5±5.4	41.2±3.1	52.6±1.9
optimal	✓	✓	90.8±2.8	86.7±3.6	93.2±2.4	42.8±4.6	42.8±2.3	66.8±0.4

Table 6.4 Average dataset accuracy and standard deviations, over all prompts and label words. **baseline** and **null-norm** are zero-resource classification methods, **prior-match** uses the text inputs but not labels, while **optimal** is an oracle approach that uses the labels to search for the best thresholds. Results shown for FlanT5-770M, FlanT5-220M and Llama2-7B.

Table 6.4 shows the performance of FlanT5 and Llama2-7B when used as prompt-based classifiers for classification tasks. We present the average accuracy across all prompts and label word permutations, as well as the standard deviation across all settings (which shows the performance fluctuations). We compare performance for the baseline prompt classifier, as well as the three reweighting approaches (optimal, prior-match, null-norm), which use varying amounts of data to estimate the values of $\alpha_{1:K}$.

Table 6.4 demonstrates that prompt-based classifiers are highly sensitive to the choice of label words. When reweighting is not applied (baseline), the performance fluctuates significantly, with standard deviations exceeding ten percentage points on many datasets for both FlanT5 and Llama2. However, prior-matching effectively mitigates label word bias, and this unsupervised reweighting approach results in consistent performance gains. For FlanT5-770M, the average accuracy increases between 6.7% to 12.1% for sentiment classification, 13.7% for QQP and over 25% for NLI. Even for QQP, where the dataset has twice as many non-paraphrases as paraphrases, prior-matching (which assumes a uniform label distribution) yields performance improvements of 14% and 6% for FlanT5-770M and FlanT5-220M, respectively (though only mild improvements for Llama2-7B).

Not only does prior-matching perform well, but prior-matching achieves similar accuracies as when using the optimal weights. The optimal weights are determined by using the classification labels and searching for the weights that give the highest possible accuracy, serving as an upper bound. Nonetheless, the unsupervised prior-matching method often achieves accuracies close to those from the optimal weights, only diverging beyond 3% for QQP, where the dataset has a significant class imbalance. This further highlights that label word bias severely impacts the classifier, and correcting this implicit class prior alone can account for a lot of the performance fluctuations observed across settings, yielding near-optimal reweighting performance.

Finally, the data-free null-input reweighting (null-norm) can achieve reasonable zero-resource debiasing. When proposed by [355], they mainly focused on a few-shot setting where a large source of bias was the few-shot examples provided. We verify that even in zero-shot settings, null-input reweighting can be effective at improving the expected performance and reducing sensitivity to prompts or label words. The performance gap between prior-matching and null-norm is also usually within 3%, and so the motivation of using the null-input to approximate the class prior for the task appears effective and practical in data-free settings.

Prompt Robustness

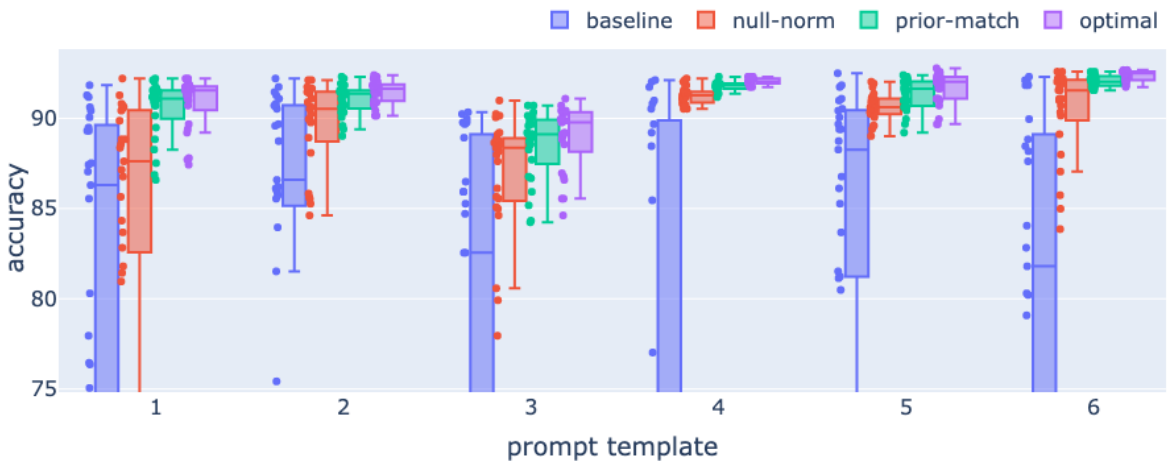


Fig. 6.3 Boxplots of the accuracy of all label-word sets for RT for all 6 prompts.

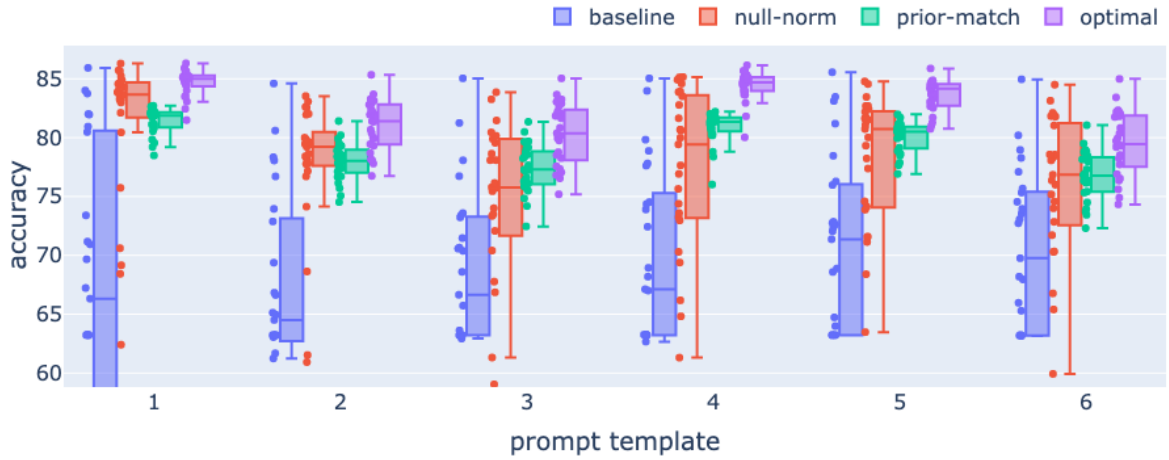


Fig. 6.4 Boxplots of the accuracy of all label-word sets for QQP for all 6 prompts.

The previous results demonstrated that the choice of label words can severely impact the classification performance of the system but that debiasing via reweighting can alleviate this bias, reducing the variance and improving the expected accuracy. However, the choice of the prompt may also largely influence the observed performance [297, 360]. Particular prompts may better describe the task or better interact with selected label words, and therefore, we may want to confirm that reweighting is effective across different prompts. Figures 6.3 and 6.4 present the accuracies of different settings at the prompt level. Here, for a given prompt (1-6), each point represents the accuracy when using this prompt with a particular permutation of selected label words (e.g. good-terrible or amazing-bad). We do this for all

possible combinations of label words, then provide the box plot to depict the 25th, 50th and 75th percentile of performance, and repeat analysis for the performances using the different reweighting approaches.

As previously seen in Table 6.4, the boxplots demonstrate that standard prompting predictions are highly sensitive to the prompt-classifier settings. In some settings, performance is reasonable, though, in other settings, the model can suffer from label word bias and have poor accuracies. Some prompts appear more robust and have settings with higher accuracies than settings from other prompts. However, debiasing predictions using reweighting, whether through data-free, unsupervised or supervised methods, results in robust performance where all settings perform reasonably well for the task. For example, prior-matching on rotten tomatoes results in accuracies above 85% for all prompt-classifier settings, while for QQP, all prior-matching settings achieve accuracies above 72%.

Weight Alignment

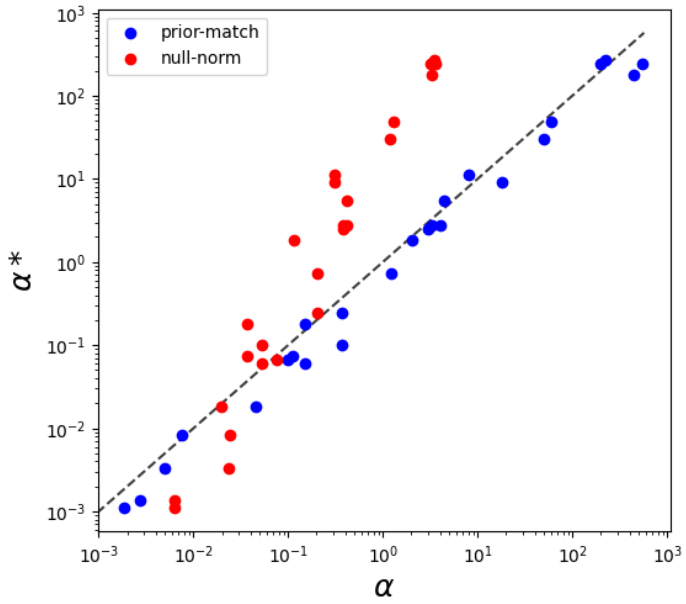


Fig. 6.5 Scatter plot of the values of the optimal weights $\boldsymbol{\alpha}^*$ against the prior match weights $\bar{\boldsymbol{\alpha}}$ (blue) and the approximation via null-input (red) for FlanT5-large on Amazon

As discussed in section 6.3, better parameter estimates can be derived based on data availability. In a supervised setting, the optimal weights that maximise accuracy can be determined. For unlabelled data, weights that result in a uniform prior can be computed. In cases where no data is available, the zero-resource approximation can be applied, which approximates the

prior using a null input. In the next experiments, we analyse how the weights $\boldsymbol{\alpha}$ compare across different search methods.

Figure 6.5 shows a scatter plot comparing the weights found by the optimal threshold search $\boldsymbol{\alpha}^*$, with weights from the unsupervised and data-free method. First, we observe that the values of α can be quite large for a particular label word set, in some cases with the weights scaling the probabilities by a factor of 100. This highlights that the level of bias caused by the label word bias can be very substantial. However, a fixed scaling factor can account for this bias and yield good robust performance across settings (seen by the performance boosts observed in the previous results). Furthermore, we see a clear linear relationship between optimal and prior-match, illustrating that accounting for label word bias is almost equivalent to maximising accuracy but achieved in an unsupervised setting. The null-norm approach is also well correlated with the optimal thresholds, although the steeper gradients imply that the null-norm weights underestimate the level of bias. This might be because the null input is an unnatural input text, so the entropy over the tokens may be larger than expected, which underestimates the level of bias.

6.5.2 Multiple Choice Question Answering

The results in the previous section focused on label word bias when LLMs were applied to text classification tasks. The next form of bias investigated is positional bias in multiple choice question answering (MCQA). As discussed in section 6.2.4, MCQA tasks exhibit permutation invariance where the ordering of the options should not alter the correct answer. The experiments in this section analyse the degree of permutation bias present in zero-shot prompt classifiers when applied to MCQA tasks.

	RACE++			CosmosQA			ReClor			ARC-EASY		
	acc	pb	ps	acc	pb	ps	acc	pb	ps	acc	pb	ps
FlanT5-3B												
baseline (biased)	86.7	0.01	0.05	85.7	0.01	0.06	54.8	0.02	0.12	85.3	0.03	0.08
prior-matching	86.5	0.03	0.06	85.6	0.01	0.06	54.0	0.01	0.12	85.9	0.01	0.08
perm-debias	87.3	0.00	0.00	86.1	0.00	0.00	54.2	0.00	0.00	86.8	0.00	0.00
FlanT5-11B												
baseline (biased)	88.8	0.02	0.05	85.8	0.03	0.07	57.0	0.05	0.15	89.5	0.02	0.07
prior-matching	88.3	0.02	0.06	86.0	0.01	0.06	57.8	0.02	0.14	89.3	0.02	0.07
perm-debias	88.9	0.00	0.00	87.4	0.00	0.00	59.6	0.00	0.00	90.2	0.00	0.00
Llama2-7B												
baseline (biased)	61.2	0.16	0.33	52.2	0.15	0.37	38.8	0.31	0.49	76.2	0.08	0.22
prior-matching	61.9	0.02	0.28	54.1	0.01	0.32	40.8	0.01	0.36	76.1	0.02	0.21
perm-debias	68.3	0.00	0.00	60.8	0.00	0.00	48.6	0.00	0.00	83.7	0.00	0.00
Llama2-13B												
baseline (biased)	71.3	0.10	0.21	63.4	0.09	0.27	49.6	0.15	0.33	82.7	0.03	0.14
prior-matching	71.8	0.02	0.19	65.1	0.01	0.24	50.2	0.03	0.29	83.1	0.01	0.14
perm-debias	74.6	0.00	0.00	70.2	0.00	0.00	53.2	0.00	0.00	87.9	0.00	0.00

Table 6.5 Accuracy (acc), positional bias (pb) and permutation sensitivity (ps) for various LLMs when prompted for Multiple Choice Question Answering.

Table 6.5 displays the accuracy, positional bias (Equation 6.50) and permutation sensitivity (Equation 6.48) for FlanT5 and Llama systems. Given that MCQA tasks tend to be more complex than sentiment classification or NLI, the analysis is extended to larger model sizes of FlanT5 and Llama2. The first row in each block presents the baseline performance when the LLM is prompted in the standard prompt-classifier setting. The performance difference between models on different tasks can be explained by instruction-training data, model size, and task difficulty. FlanT5 was aligned through instruction tuning (§3.3.3) on the FLAN collection [193], which includes extensive and diverse tasks, including MCQA datasets such as CosmosQA. In contrast, Llama2 was instruction-tuned on a limited number of conversational interactions, with 100× fewer examples than FLAN. Therefore, it's unsurprising that FlanT5, which was directly trained on many MCQA datasets, outperforms Llama2 even though Llama2 may have more model parameters. Further, task difficulty can influence performance; ReClor, a graduate-level multiple-choice reasoning exam, is more challenging than ARC-EASY, a middle school science exam (where there is no contextual passage). Therefore, all models have accuracies below 60% on ReClor, but all achieve accuracies above 75% for ARC-EASY. Additionally, increasing the number of parameters within the same

family improves performance, with FlanT5-11B performing better than FlanT5-3B, while Llama2-13B outperforms Llama2-7B.

Beyond MCQA performance, the table also highlights that LLMs may be highly sensitive to the permutation of the input options. Llama2 has a permutation sensitivity of over 0.2 for many tasks; if provided with two options, this is equivalent to predicting a distribution of [0.4, 0.6] but then flipping the distribution to [0.6, 0.4] when the options are reordered. Similarly, the positional bias of 0.1 to 0.3 highlights that the marginalised distribution over positions is non-uniform and has a systematic preference for certain positions over others. FlanT5, though, is more permutation invariant and has much smaller values of permutation sensitivity and positional bias. This is likely due to its supervised instruction tuning on MCQA tasks and its implicit exposure to this property of MCQA tasks.

We further observe that permutation debiasing yields significant improvements in MCQA performance. For LLama2, the performance gains can be large and often within 5-10%. Even for FlanT5, which had low permutation sensitivity, we still observe mild performance improvements of 1-2%. Permutation debiasing guarantees zero permutation sensitivity and positional bias, and the performance improvement implies that being aware of this invariance may help system performance. Prior-matching, which minimises the positional bias[2], does not alone resolve the permutation sensitivity. In some cases, prior-matching can improve performance and sensitivity, though in other tasks, it can be notably worse than permutation debiasing. This implies that positional bias alone does not account for the observed permutation sensitivity. It may also be noted that a loose correlation between permutation sensitivity and accuracy can be observed across tasks and models.

6.5.3 Efficient Debiasing

The previous results in Table 6.5 demonstrated that permutation debiasing was an effective debiasing approach which, by design, completely eliminates permutation sensitivity. Accounting for and correcting this permutation sensitivity led to considerable boosts in performance. However, a limitation is the $K!$ model calls required for full permutation debiasing, which can be prohibitive due to the high latency when deployed in real-world scenarios. Section 6.4 discussed the teacher-student distillation framework, where the debiased teacher's decisions can be distilled to a smaller student to maintain reasonable inference costs. In the next experiments, we will examine the abilities of both distillation students (§6.4.1) and error-correction students (§6.4.2). We baseline the student's performance against the teacher's performance, where the following teacher setups are considered:

[2]In Table 6.5 the positional bias is not always zero for prior-matching as we find the weights by using only the original permutations, but the metrics are computed over all 24 permutations

- **Expected biased black-box**: In this setup, it's assumed that we do not have access to the output probability distribution over options. Therefore, all that can be done is to sample a single teacher decision from the model when using a single arbitrary ordering of the options for the MCQA question,

$$\tilde{\boldsymbol{a}} \sim \mathtt{P}(\boldsymbol{a}|\boldsymbol{q},\boldsymbol{c},\mathrm{A}_\sigma;\boldsymbol{\theta}) \tag{6.51}$$

The expected black box performance is the expected performance of the system over all questions and permutations. Given a dataset $\mathcal{D} = \{(\boldsymbol{q}^{(i)},\boldsymbol{c}^{(i)},\boldsymbol{a}^{(i)}_{1:K},y^{(i)})\}_{i=1}^{M}$, where $y^{(i)} \in \{\boldsymbol{a}_1,...\boldsymbol{a}_K\}$ is the correct answer for the question, the expected biased black box performance can be calculated as,

$$\mathtt{ebb}(\boldsymbol{\theta};\mathcal{D}) = \frac{1}{M\cdot K!}\sum_{i=1}^{M}\sum_{j=1}^{K!}\mathtt{P}\left(y^{(i)}|\boldsymbol{q}^{(i)},\boldsymbol{c}^{(i)},\mathrm{A}^{(i)}_{\sigma_j};\boldsymbol{\theta}\right) \tag{6.52}$$

Where $\mathrm{A}^{(i)}_{\sigma_j}$ is the j-th permutation of the options $a^{(i)}_{1:K}$, such that there are K! total possible permutations.

- **Expected biased white-box**: In this setup, it's assumed that we have access to the output probability distribution of the teacher system, but that permutation debiasing is not applied. Therefore, the prediction is the option with the highest associated probability for a single arbitrary ordering of the options,

$$\hat{\boldsymbol{a}} = \underset{\boldsymbol{a}\in\boldsymbol{a}_{1:K}}{\arg\max}\,\mathtt{P}(\boldsymbol{a}|\boldsymbol{q},\boldsymbol{c},\mathrm{A}_\sigma;\boldsymbol{\theta}) \tag{6.53}$$

Similar to the black-box scenario, the accuracy calculated is the expected performance over all questions and permutations:

$$\mathtt{ebw}(\boldsymbol{\theta};\mathcal{D}) = \frac{1}{M\cdot K!}\sum_{i=1}^{M}\sum_{j=1}^{K!}\mathbb{1}\left(y^{(i)} = \underset{\boldsymbol{a}\in\boldsymbol{a}^{(i)}_{1:K}}{\arg\max}\,\mathtt{P}\left(\boldsymbol{a}|\boldsymbol{q}^{(i)},\boldsymbol{c}^{(i)},\mathrm{A}^{(i)}_{\sigma_j};\boldsymbol{\theta}\right)\right) \tag{6.54}$$

- **Permutation debiased white-box**: The final setting considered is when it's assumed that we have access to the output probability distribution of the teacher system and also perform permutation debiasing. The prediction is the option with the highest

associated probability after ensembling the probabilities from all permutations,

$$\hat{\boldsymbol{a}} = \underset{\boldsymbol{a} \in \boldsymbol{a}_{1:K}}{\arg\max} \left(\frac{1}{K!} \sum_{j=1}^{K!} \mathtt{P}(\boldsymbol{a}|\boldsymbol{q}, \boldsymbol{c}, \mathrm{A}_{\sigma_j}; \boldsymbol{\theta}) \right) \tag{6.55}$$

And the performance is the associated accuracy of the system over the dataset,

$$\mathtt{pdw}(\boldsymbol{\theta}; \mathcal{D}) = \frac{1}{M} \sum_{i=1}^{M} \mathbb{1} \left(y^{(i)} = \underset{\boldsymbol{a} \in \boldsymbol{a}_{1:K}^{(i)}}{\arg\max} \left(\frac{1}{K!} \sum_{j=1}^{K!} \mathtt{P}(\boldsymbol{a}|\boldsymbol{q}^{(i)}, \boldsymbol{c}^{(i)}, \mathrm{A}_{\sigma}^{(i)}; \boldsymbol{\theta}) \right) \right) \tag{6.56}$$

We distil the decisions of the permutation-debiased white-box teacher onto a student encoder-only transformer. For the student 'proxy' models, we consider both RoBERTa [185] and DeBERTa-v3 [110], and use both the base (110M) and large (330M) size. The input text to the student system is the same as that to the teacher, though for error correction, we further provide the biased teacher decision by appending text to the end of the input prompt. E.g. If the sampled biased teacher prediction was '`A`', then we concatenate '`Prediction: A`' to the end of the input text. We distill using the entire RACE++ training dataset for 2 epochs and train 4 seeds per RACE++ setting, reporting the average performance.

	type	acc	pb	ps
Teachers				
permutation debiased white-box		68.3	0.00	0.00
expected biased white-box		61.2	0.16	0.33
expected biased black-box		58.4	-	
Students				
RoBERTa-base (110M)	d	26.7	0.03	0.03
RoBERTa-base (110M)	ec	61.4	0.03	0.18
DeBERTa-base (110M)	d	26.9	0.02	0.03
DeBERTa-base (110M)	ec	64.1	0.01	0.15
RoBERTa-large (330M)	d	26.9	0.04	0.04
RoBERTa-large (330M)	ec	68.0	0.03	0.12
DeBERTa-large (330M)	d	47.9	0.03	0.06
DeBERTa-large (330M)	ec	68.1	0.03	0.13

Table 6.6 Performance of a student trained to emulate the debiased teacher, measured by task accuracy (acc), positional bias (pb) and permutation sensitivity (ps). The students are either distilled (d) or trained to correct the distribution of a single biased black-box teacher decision (ec). Llama2-7B is used as the teacher.

Table 6.6 shows the performance of the students when trained to emulate the teacher debiased decisions, with students either distilled (§6.4.1) or trained to perform error-correction (§6.4.2). It's observed that for RACE++, which is a challenging task that standard encoder-only transformers struggle to learn even with supervised training [268], the distilled student is not powerful enough to alone capture the teacher's task abilities. Although these systems have very low permutation sensitivity, this occurs as the model is very uncertain and always predicts a near-uniform distribution. However, the error correction students can effectively leverage a single-biased teacher decision to predict the estimated general debiased distributions. Here, the error correction student systems are more robust to the permutations and have lower permutation sensitivity (0.33 vs 0.12-0.18), although are not fully permutation invariant. This occurs as the error-correction student uses information from the teacher's prediction, and if the teacher has strong permutation sensitivity, then the student will also be dependent on the ordering due to the sampled teacher's prediction. Interestingly, the error correction students consistently perform better than just copying the sampled decision from the black-box teacher (accuracy of 58.4 vs accuracy of 61.4-68.1), illustrating that the students can capture useful information about the underlying teacher's prediction space.

Black-Box Training Efficiency

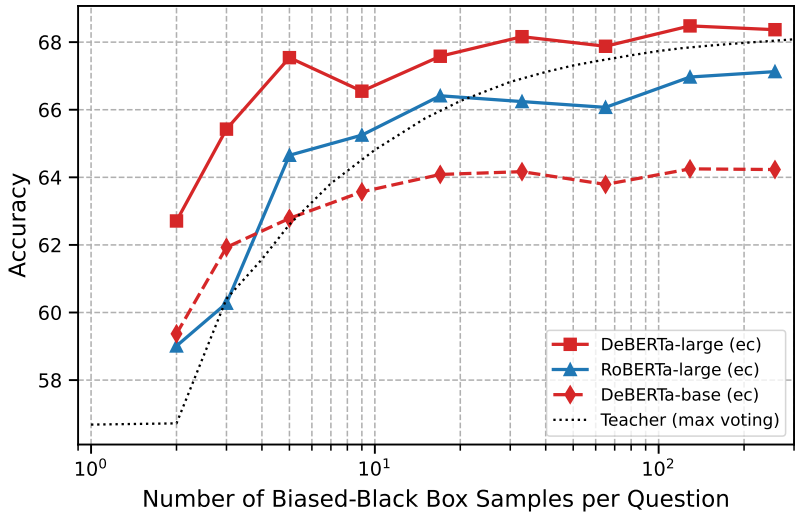


Fig. 6.6 RACE++ performance of error correction students when using M black-box samples to approximate the debiased distribution (§6.4.3)

The previous results applied the teacher-student training framework, where the student system is trained using predictions from the debiased teacher. Permutation debiasing, though, assumes access to the teacher's probabilities, which may not always be available. In black cases where the probabilities of the outputs are not available, one can sample M teacher predictions and use a Monte Carlo approximation to approximate the underlying true distribution (§6.4.3). The next experiments investigate the sample efficiency of the framework in black-box settings. Figure 6.6 displays the RACE++ performance of an error correction student when trained using M black-box teacher samples per example.

The curve illustrates that teacher-student training does not require an excessive number of black-box samples, with performance saturating at 32 samples per example. Interestingly, when using only a few samples, DeBERTa-large can outperform the max-voting performance of the teacher used to distil the student. The max-voting teacher performance is represented by the dotted line in Figure 6.6, which uses the mode of the M samples as the prediction. To train error-correction students, at least two teacher samples are required per question, one to act as the input to the student system and the rest to perform the Monte Carlo estimate of the debiased teacher distribution. The improved performance over the teacher max-voting implies that by applying teacher-student training, the student can infer the systematic biases present in the teacher and yield corrected distributions using many noisy approximations.

6.6 Chapter Summary

This chapter analysed the bias present in zero-shot LLM classifiers and proposed approaches for mitigating the influence of the bias. For label word bias, where the choice of label words can cause implicit class priors, we considered a reweighting strategy to rescale the probabilities of classes. Three variants were considered depending on data limitations, including labelled data, unlabelled data, and zero-resource, which all improved the performance of zero-shot text classifiers and reduced the perceived class bias. Further, we considered permutation debiasing and proposed the student-teacher distillation framework, a practical realisation for inference-efficient debiasing. Experimental results demonstrated that LLMs are prone to exhibiting positional bias, which can impact performance and reliability, but that permutation debiasing and our efficient variants can mitigate such biases.

Chapter 7

Comparative Assessment

Chapter 4 discussed two applications for NLP foundation models: text classification and text assessment. The previous chapter highlighted that when applied for zero-shot text classification, instruction-following foundation models demonstrate biases such as label word bias and positional bias. This chapter now shifts the focus to text assessment, with two primary objectives: first, to determine whether instruction-following LLMs can be better leveraged for NLG assessment. Second, to analyse and account for any biases present.

Traditional NLG assessment methods (§4.2) are either general but reference-based (§4.2.3), which may fail to capture particular properties, or tailored to specific attributes (§4.2.4), which requires labelled data and bespoke solutions that may limit their generalisability. With the advent of instruction-following LLMs, recent methods have been proposed to prompt LLMs to perform NLG assessment in a zero-shot or few-shot fashion (§4.2.5). These methods aim to provide effective assessment that is both general and low-resource. For example, in prompt-scoring, which is motivated by absolute human assessment, the LLMs are prompted to directly assess the quality of texts on a rubric-based scale [326, 182, 357].

With the insight that for humans, it is often easier to determine which of two like objects is better than it is to score items on an absolute scale [161, 29], this chapter investigates whether LLMs can instead be leveraged for comparative assessment. We propose LLM Comparative Assessment[1], where assessment is done by considering pairwise comparisons between texts and then aggregating the results. This chapter first discusses the LLM comparative assessment methodology and how it can be applied to general NLG assessment. Section 7.1 first introduces the method of LLM comparative assessment, while Section 7.2 extends the framework to use the product of experts, which makes the processes significantly more

[1]Concurrent work by Qin et al. [259] and Zheng et al. [357] also explore using LLMs as judges for pairwise decisions, though with different objectives (information retrieval and evaluating open-ended dialogue systems). Neither work considers ranking texts specifically for NLG assessment.

efficient and practical. Section 7.3 then provides experimental results that examine the impact of positional bias and showcase the effectiveness of LLM comparative assessment.

7.1 LLM Comparative Assessment

LLM comparative assessment is motivated by the law of comparative judgement (§4.3). Here, items are ranked by selecting two items, getting a judge to decide which of the two is better, and repeating this for many different pairs of candidates [251]. However, deploying human assessors at scale can be cost-prohibitive, time-consuming, and logistically challenging, particularly when specialised expertise is needed. Given the impressive instruction-following [234, 42] capabilities of LLMs such as GPT-4 [2] and open-sourced variants [42, 314], we explore whether LLMs can serve as LLM judges to determine which of two texts better displays a particular attribute. This section first discusses what makes the task challenging and later proposes how LLM comparative assessment can be performed.

7.1.1 Challenges in Modelling Comparative Decisions

Comparative judgment can be considered a more intuitive form of assessment with higher validity than absolute scoring [28, 142, 251]. Comparing like-for-like items is often seen as a purer form of assessment, as instead of relating items to an internalised standard [89], two similar items are directly compared. Therefore, it may be worthwhile to investigate whether LLMs can be leveraged for pairwise comparative assessment.

Given that assessment can be used in critical applications, it is crucial that methods are as reliable as possible with minimal impact from bias. However, LLMs are known to exhibit specific biases that can affect downstream tasks (§6.1.3), and additionally, comparative judgment has recognised practical limitations. Therefore, when prompting LLMs to make pairwise judgements for NLG assessment, it may be important to remain aware of potential challenges and risks:

- **Positional bias**: When making pairwise judgements, the LLM judge would be given two options and prompted to select the option that best reflects the assessed attribute. When the two inputs are provided to the model, there will be an arbitrary ordering where one option will appear before the other in the prompt. Experiments in the previous chapter (§6.5.2) demonstrated that instruction-following LLMs can exhibit significant positional bias, where the ordering of the options may highly influence its decision. This can reduce the reliability and efficacy of the method, and therefore the impact of positional bias should be analysed and minimised.

- **Style bias**: An important aspect of assessment is fairness. Fair assessment is open-minded, such that diverse perspectives and styles are accepted and treated similarly provided the targeted attribute is demonstrated. LLMs, though, can be biased and prefer responses of particular styles, and for example have shown to prefer texts similar to those that it would have generated [184, 357]. Therefore, we want to minimise the impact that self-bias may have and ensure that different styles are not severely penalised.
- **Computational inefficiency**: The total number of possible pairwise comparisons scales with $O(N^2)$. For comparative judgment applications, comparisons can also be judged multiple times by different pools of judges [267]. These costs, even when applied with automatic assessment approaches, can scale impractically when a large number of candidates are assessed. Therefore these costs should be considered, and the developed solution should aim to minimise the number of comparisons made, where each comparison should be as computationally efficient as possible.

To address these risks, we will explore methods for analysing and mitigating positional bias, consider approaches to reduce the number of required pairwise LLM computations (§7.2.1), and in experiments, use independent LLM judges to minimise the risk of self-bias.

7.1.2 LLM Comparative Assessment

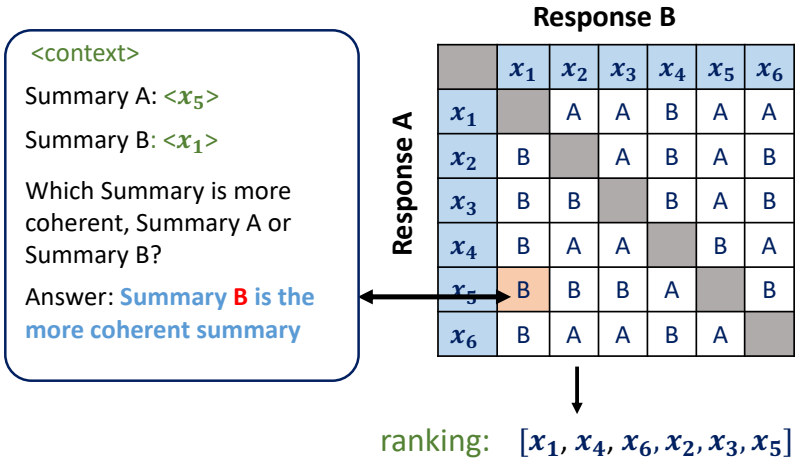


Fig. 7.1 LLM Comparative Assessment prompts an LLM to compare candidates in a pairwise manner, where the comparisons are then converted into scores or ranks.

Having previously discussed the beneficial properties of zero-shot, reference-free and general text evaluation (§4.2.5) as well as the perceived advantages of comparative judgements (§4.3), this section now combines the two principles and proposes LLM comparative assessment.

For a particular task $\mathcal{T}$, let $\boldsymbol{c}$ be the context (e.g., a dialogue, article, or set of semantic triples), $\boldsymbol{x}_{1:N}$ a set of N candidate texts generated conditioned on the context, and $s^*_{1:N} \in \mathbb{R}$ the respective ground truth scores of each text for the assessed attribute (e.g., fluency, coherence, or relevance). An NLG assessment system f predicts scores $\hat{s}_{1:N}$,

$$\hat{s}_{1:N} = f(\boldsymbol{x}_{1:N}, \boldsymbol{c}) \tag{7.1}$$

The primary objective is for the predicted scores, $\hat{s}_{1:N}$, to correlate highly with the true scores, $s^*_{1:N}$. However, in many practical applications, the goal is not necessarily for the scores to match exactly but for the texts to be ranked in the correct order. Therefore, a more typical objective is for the predicted ranks, $\hat{r}_{1:N}$, to match the true ranks, $r^*_{1:N}$, where scores $s_{1:N}$ can be converted to ranks $r_{1:N}$ following:

$$r_i = 1 + \sum_{j=1}^{N} \mathbb{1}(s_j > s_i) \tag{7.2}$$

In LLM comparative assessment (illustrated in Figure 7.1), instead of directly scoring each input text $\boldsymbol{x}_i$, the LLM performs the intermediary task of determining which of two sampled input responses is better. These can then be aggregated to predict overall scores and then text rankings. The next sections will discuss the low-level details of how LLM comparative assessment can be designed for NLG assessment, including the process for obtaining LLM judgements, aggregation methods to derive final rankings and computational efficiency considerations.

7.1.3 Prompt Classifier Set Up

We explore framing the LLM comparative assessment judge as a zero-shot prompt classifier (§4.1.3). The prompt classifier is used as it ensures a definitive output decision and directly provides probabilities associated with each class. Responses could alternatively be freely generated from the LLM, which would no longer restrict the output text sequence and allow the model to provide its reasoning. However, free generation requires a separate system to map output texts to decisions and would not capture probabilistic information.

For a given task and assessment attribute, the prompt $\mathcal{P}(\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{c})$ takes two competing texts as inputs, $\boldsymbol{x}_i, \boldsymbol{x}_j$ along with the context $\boldsymbol{c}$, and generates a prompt template that requests

the LLM to determine which of the two texts better demonstrates the assessed attribute, e.g.,

$$\begin{aligned}\mathcal{P}(\boldsymbol{x}_i,\boldsymbol{x}_j,\boldsymbol{c}) = \textit{`Context: <}\boldsymbol{c}\textit{>}\\ \textit{Which Summary is more coherent, Summary A or Summary B?}\\ \textit{Summary A: <}\boldsymbol{x}_i\textit{>}\\ \textit{Summary B: <}\boldsymbol{x}_j\textit{>'}\\ \textit{Summary \_}\end{aligned}$$

The prompt is designed to be very simple and is not overly prompt-engineered. Although more complicated prompts, such as those with few-shot examples or more detailed descriptions, may yield better performance, we focus on having a general and effective approach that does not require excessive tweaking or bespoke design. Instead, the above simple prompt can be easily adapted to diverse tasks and domains, by simply replacing keywords in the prompt template, e.g.,

$$\begin{aligned}\mathcal{P}(\boldsymbol{x}_i,\boldsymbol{x}_j,\boldsymbol{c}) = \textit{`Dialogue: <}\boldsymbol{c}\textit{>}\\ \textit{`Which Response is more engaging, Response A or Response B?}\\ \textit{Response A: <}\boldsymbol{x}_i\textit{>}\\ \textit{Response B: <}\boldsymbol{x}_j\textit{>}\\ \textit{Response \_}\end{aligned}$$

The comparative judgment decisions can then be determined by looking at the LLM probabilities associated with each label word. The label words[2] (z_1, z_2) can be selected to match the format of the prompt, where for the above prompts,

$$z_1 = \texttt{A} \qquad\qquad z_2 = \texttt{B} \tag{7.3}$$

The LLM can then be used to determine the probability that $\boldsymbol{x}_i$ is better than $\boldsymbol{x}_j$ for the particular attribute, where following the prompt-classifier setup, the probability associated to the LLM comparative judgements, p_{ij}, can be calculated as,

$$p_{ij} = \texttt{P}(\boldsymbol{x}_i \succ \boldsymbol{x}_j|\boldsymbol{c},\mathcal{P},z_{1:2};\boldsymbol{\theta}) = \frac{\texttt{P}_{lm}(z_1|\mathcal{P}(\boldsymbol{x}_i,\boldsymbol{x}_j,\boldsymbol{c});\boldsymbol{\theta})}{\texttt{P}_{lm}(z_1|\mathcal{P}(\boldsymbol{x}_i,\boldsymbol{x}_j,\boldsymbol{c});\boldsymbol{\theta}) + \texttt{P}_{lm}(z_2|\mathcal{P}(\boldsymbol{x}_i,\boldsymbol{x}_j,\boldsymbol{c});\boldsymbol{\theta})} \tag{7.4}$$

[2]Here the label words are assumed to be a single token, e.g. `A/B`, but one can alternatively use label sequences $\boldsymbol{z}$. For simplicity, we assume that label words are used, although using sequences does not influence the analysis.

Note that p_{ij} denotes the probability that the LLM believes that the first text, $\boldsymbol{x}_i$, is better than the second text, $\boldsymbol{x}_j$. This depends on the input ordering of the input texts, and if a positional bias is present (as discussed in Section 7.1.1) the model may provide inconsistent judgements where $p_{ij} \neq (1-p_{ji})$.

In the Thurstonian model [310] (§4.3.1), which serves as the basis for comparative judgement, comparative decisions of the assessors are assumed to be distributed normally around the true quality difference of the two items.

$$\mathtt{P}(\boldsymbol{x}_i \succ \boldsymbol{x}_j | \boldsymbol{c}) = \Phi\left(\frac{s_i^* - s_j^*}{\sqrt{2\sigma^2}}\right) \tag{7.5}$$

Therefore, for a given pair of items, the result for a given comparison may be independently drawn many times from the assumed underlying distribution using different assessors or by considering multiple games. Given n_{ij} victories for $\boldsymbol{x}_i$ and n_{ji} victories for $\boldsymbol{x}_j$, the probability of victory can then be estimated as,

$$\mathtt{P}(\boldsymbol{x}_i \succ \boldsymbol{x}_j | \boldsymbol{c}) \approx \frac{n_{ij}}{n_{ij} + n_{ji}} \tag{7.6}$$

In LLM comparative assessment, instead of sampling from the true distribution, we directly model the distribution $\mathtt{P}(\boldsymbol{x}_i \succ \boldsymbol{x}_j | \boldsymbol{c}; \boldsymbol{\theta})$ using the prompt-based classifier. This highlights an advantage of LLM comparative assessment, where each comparison provides an approximation for the true underlying distribution, effectively providing infinite samples using a single LLM call (although this probability may be inaccurate).

7.1.4 Positional Bias

LLM comparative assessment can be viewed as an MCQA task where there are two options, `A` or `B`. As discussed in Section 6.2.4, these tasks are order invariant, and a consistent LLM comparative assessment system should satisfy the property that:

$$\mathtt{P}(\boldsymbol{x}_i \succ \boldsymbol{x}_j | \boldsymbol{c}; \boldsymbol{\theta}) = 1 - \mathtt{P}(\boldsymbol{x}_j \succ \boldsymbol{x}_i | \boldsymbol{c}; \boldsymbol{\theta}) \qquad \forall i, j \tag{7.7}$$

where for convenience the dependence on the prompt $\mathcal{P}$ and label words $z_{1:2}$ is dropped. Equation 7.7 states that for comparative assessment, if the LLM satisfies the invariance property, then the probability it assigns to $\boldsymbol{x}_i$ being better than $\boldsymbol{x}_j$ should be one minus the probability it assigns to $\boldsymbol{x}_j$ being better than $\boldsymbol{x}_i$. The experiments in Section 6.5.2, however, highlighted the susceptibility of prompted LLMs to have permutation sensitivity and positional bias. Therefore, one can perform a similar analysis and measure both permutation

sensitivity and position bias. For comparative assessment, where there are only two options, the permutation sensitivity has form:

$$\texttt{ps}(\boldsymbol{\theta}) = \mathbb{E}_{\{\boldsymbol{c},\boldsymbol{x}_i,\boldsymbol{x}_j\}}\left[\left|\texttt{P}(\boldsymbol{x}_i \succ \boldsymbol{x}_j|\boldsymbol{c};\boldsymbol{\theta}) - (1-\texttt{P}(\boldsymbol{x}_j \succ \boldsymbol{x}_i|\boldsymbol{c};\boldsymbol{\theta}))\right|\right] \tag{7.8}$$

While for position bias, we first marginalise over all contexts and questions to find the prior for the first position being preferred,

$$\texttt{P}(\texttt{A};\boldsymbol{\theta}) = \mathbb{E}_{\{\boldsymbol{c},\boldsymbol{x}_i,\boldsymbol{x}_j\}}\left[\texttt{P}(\boldsymbol{x}_i \succ \boldsymbol{x}_j|\boldsymbol{c};\boldsymbol{\theta})\right] \tag{7.9}$$

The permutation sensitivity, which measures the total variation of the position distribution with the uniform distribution, can then be calculated as:

$$\texttt{pb}(\boldsymbol{\theta}) = \left|\texttt{P}(\texttt{A};\boldsymbol{\theta}) - 0.5\right| \tag{7.10}$$

To correct for any permutation sensitivity and positional bias, permutation debiasing (§6.2.4) was shown to be an effective debiasing approach. Permutation debiasing averages the probabilities from both permutations, which for comparative assessment results in the debiased prediction $\tilde{p}_{ij}$ taking form:

$$\tilde{p}_{ij} = \frac{p_{ij} + (1-p_{ji})}{2} \tag{7.11}$$

7.1.5 Comparisons to Ranks

Comparative decisions only provide pairwise preferences, but the objective of NLG assessment is to generate a final ranking/scoring on the perceived quality of the texts. As inconsistencies can arise in LLM predictions, it may not be possible to find a final ranking that is consistent with all the pairwise judgements. For example, consider the scenario with three texts where the LLM predicts that $\boldsymbol{x}_1 \succ \boldsymbol{x}_2$, $\boldsymbol{x}_2 \succ \boldsymbol{x}_3$ and $\boldsymbol{x}_3 \succ \boldsymbol{x}_1$. In this scenario, any proposed ranking will contradict at least one of these comparisons. Therefore, the objective is not to find a ranking that is perfectly consistent with the pairwise decisions but, instead, a ranking that is as consistent as possible with the observations.

With no computational restrictions, one can compute all $N \cdot (N-1)$ comparisons between all pairs of texts, which provides the most information from the LLM. A straightforward method to convert comparisons to a final ranking is to calculate the **average probability** of each text. Here, the predicted score is the mean probability associated with a given text over

all its comparison, calculated as:

$$\hat{s}_i = \frac{1}{N-1} \sum_{j=1, j\neq i}^{N} \frac{p_{ij} + (1 - p_{ji})}{2} \tag{7.12}$$

This approach considers all comparisons involving $\boldsymbol{x}_i$, whether it appears in the first or second position, and averages all probabilities. By utilising all possible pairwise comparisons, both permutations will be evaluated, and hence, this method inherently performs permutation debiasing, avoiding positional bias and ensuring the rankings are not influenced by the input ordering of the texts.

As an alternative, one can consider the 'hard' version of the average probability, which is the **win ratio**. Here, each probability is first converted into the predicted outcomes $\hat{y}_{ij} \in \{0, 1\}$, by selecting the most likely outcome:

$$\hat{y}_{ij} = \begin{cases} 1, & \text{if } p_{ij} > 0.5 \\ 0, & \text{otherwise} \end{cases} \tag{7.13}$$

The score associated with the text is then the proportion of comparisons the respective text has won:

$$\hat{s}_i = \frac{1}{N-1} \sum_{j=1, j\neq i}^{N} \frac{\hat{y}_{ij} + (1 - \hat{y}_{ji})}{2} \tag{7.14}$$

This approach can be applied when probabilistic information is not available but when binary decisions $\hat{y}_{ij}$ can be sampled directly from an LLM. When all comparisons are used, calculating the win ratio will be fair and not cause any text to gain an unfair advantage from the ordering, as both permutations are used. However, positional bias may impact the win ratio more significantly than the average probability, as a strong positional bias may cause many decisions rounded to a victory for a position (e.g. `A`), while the average probability may preserve the degree of preference within each comparison.

Once we have obtained the predicted scores for each text, $\hat{s}_{1:N}$, whether through hard decisions or average probabilities, we can convert these scores to predicted ranks, $\hat{r}_{1:N}$. As discussed previously, this can be simply performed by counting the number of texts that have a higher predicted score:

$$\hat{r}_i = 1 + \sum_{j=1}^{N} \mathbb{1}(\hat{s}_j > \hat{s}_i) \tag{7.15}$$

7.2 Product of Expert View of Comparative Assessment

The previous section (§7.1) introduced LLM comparative assessment, where LLM comparative judges are used with the law of comparative judgement framework. One of the challenges of LLM comparative assessment (§7.1.1) is the associated computational costs when all $N \cdot (N-1)$ comparisons are considered. As the number of candidate texts grows, due to the $O(N^2)$ scaling of the total number of comparisons, the approach becomes impractical in real-world applications.

A straightforward way of reducing the computational cost would be to only use a subset of all the possible comparisons. However, the previous metrics of the win ratio and average probability (§7.1.5) don't take into account the strength of the opponent played, and therefore, the final rankings might be skewed based on the strength of the opposition drawn. For example, if a text is only compared to the best texts in the set, it may lose all comparisons and be assigned a low score, even if it's better than many other candidates it was never compared against. This section, therefore, considers how to best aggregate all available comparative information into the final ranking, and determine an inference-efficient methodology for predicting accurate quality scores.

7.2.1 PoE framework of Comparative Assessment

For LLM comparative assessment, each comparison C_k has form (i, j, p_{ij}), where i and j refer to the index of the candidate texts selected and p_{ij} the associated LLM probability. Each comparison C_k provides not only information about which text is preferred but also 'the degree of victory'; a high probability means that the quality score s_i is likely to be significantly higher than that of s_j, while a probability near 0.5 means that the two texts are likely of similar quality. Hence, the outcome of each comparison can be interpreted as providing information on the score difference, $\mathrm{p}(s_i - s_j | p_{ij})$. This can be used to write the probability of a set of scores in the form of a Product of Experts (PoE) [118, 333]:

$$\mathrm{p}(s_{1:N}|\mathcal{C}_{1:K}) = \frac{1}{Z} \prod_{i,j \in \mathcal{C}_{1:K}} \mathrm{p}(s_i - s_j | p_{ij}) \tag{7.16}$$

where Z is a normalisation constant to ensure a valid pdf. A Product of Experts (PoE) combines the information gained from many individual models (the experts) by taking their product and normalising the result. It offers a theoretical yet intuitive approach of combining information from multiple different sources, elegantly aggregating all information into a single output pdf over all possible sets of scores. We explore using PoE to frame

LLM comparative assessment due to the flexibility of the PoE framework and its ability to probabilistically combine information from each expert.

Unlike the win ratio or average probability, this approach yields a distribution that takes the opponent's strength into account. The probability of a particular set of scores measures how consistent it is with the observed comparisons, which may be an effective method to aggregate all available comparative information. However, in order to have an analytical expression for the distribution over scores $s_{1:N}$, the particular form of the experts, $\mathrm{p}(s_i - s_j | p_{ij})$, have to be selected. This is a design choice, and any sensible distribution can be used. In the next section we first analyse Gaussian experts, a standard and simple choice of expert which has several favourable properties.

7.2.2 Linear Gaussian Experts

Within the Products of Experts framework of comparative assessment, the form of the experts has to be selected, which defines how the LLM probabilities p_{ij} are expected to impact the score difference $s_i - s_j$. Within PoE set-ups, a popular choice of experts is having Gaussian experts. Gaussian experts yield convenient properties, such as having a simple expression for the overall score distribution, which yields a closed-form solution [345].

If the underlying distribution of the experts is assumed to be Gaussian with the mean $f_\mu(p_{ij})$ and variance $f_\sigma(p_{ij})$ only dependent on the probability, such that,

$$\mathrm{p}(s_i - s_j | p_{ij}) = \mathcal{N}\big(s_i - s_j; f_\mu(p_{ij}), f_\sigma(p_{ij})\big) \tag{7.17}$$

then the PoE distribution will take the form,

$$\mathrm{p}(s_{1:N} | \mathcal{C}_{1:K}) = \frac{1}{Z} \prod_{i,j \in \mathcal{C}_{1:K}} \mathcal{N}\big(s_i - s_j; f_\mu(p_{ij}), f_\sigma(p_{ij})\big) \tag{7.18}$$

By representing the scores in vector form, $\mathbf{s} = [s_1, \ldots, s_N]^\mathsf{T}$, one can further simplify the expression to,

$$\mathrm{p}(\mathbf{W}\mathbf{s} | \mathcal{C}_{1:K}) = \mathcal{N}\left(\mathbf{W}\mathbf{s}; \boldsymbol{\mu}, \mathrm{diag}(\boldsymbol{\sigma}^2)\right) \tag{7.19}$$

Where $\mathbf{s}$ is the N-dimensional column vector of $s_{1:N}$, $\boldsymbol{\mu} \in R^K$ and $\boldsymbol{\sigma}^2 \in R^K$ is a vector of the means and variances,

$$\boldsymbol{\mu} = [f_\mu(p_{ij}^{(1)}), f_\mu(p_{ij}^{(2)}), \ldots f_\mu(p_{ij}^{(K)})]^\mathsf{T} \qquad \boldsymbol{\sigma}^2 = [f_\sigma(p_{ij}^{(1)}), f_\sigma(p_{ij}^{(2)}), \ldots f_\sigma(p_{ij}^{(K)})]^\mathsf{T} \tag{7.20}$$

and $\mathbf{W} \in R^{K \times N}$ is a matrix representing the set of comparisons, such that for the k^{th} comparison between i and j, $\mathbf{W}_{ki}=1$, $\mathbf{W}_{kj}=-1$, and $\mathbf{W}_{km}=0 \ \forall m \neq i, j$. For illustrative purposes, consider the case where there are 4 texts, $\boldsymbol{x}_{1:4}$, and all possible comparisons are computed. In this scenario, the structure of $\mathbf{W}$, $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}^2$ would be[3],

$$\mathbf{W} = \begin{bmatrix} 1 & -1 & 0 & 0 \\ 1 & 0 & -1 & 0 \\ 1 & 0 & 0 & -1 \\ 0 & 1 & -1 & 0 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & -1 \end{bmatrix} \qquad \boldsymbol{\mu} = \begin{bmatrix} f_\mu(p_{12}) \\ f_\mu(p_{13}) \\ f_\mu(p_{14}) \\ f_\mu(p_{23}) \\ f_\mu(p_{24}) \\ f_\mu(p_{34}) \end{bmatrix} \qquad \boldsymbol{\sigma}^2 = \begin{bmatrix} f_\sigma(p_{12}) \\ f_\sigma(p_{13}) \\ f_\sigma(p_{14}) \\ f_\sigma(p_{23}) \\ f_\sigma(p_{24}) \\ f_\sigma(p_{34}) \end{bmatrix} \tag{7.21}$$

Where p_{ij} denotes the output LLM comparative probability when comparing $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, e.g. p_{13} is the comparison between $\boldsymbol{x}_1$ and $\boldsymbol{x}_3$. As defined above, any shift of the scores $\boldsymbol{s}$ will yield the same output probability (Equation 7.19). To address this, an additional expert on the first element can be added such that,

$$\mathrm{p}(s_1|\mathcal{C}_0) = \mathcal{N}(0, \sigma_0^2) \tag{7.22}$$

prepending an extra row to all of $\mathbf{W}$, $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}^2$, yielding $\tilde{\mathbf{W}}$, $\tilde{\boldsymbol{\mu}}$ and $\tilde{\boldsymbol{\sigma}}^2$ respectively. The distribution takes a similar form,

$$\mathrm{p}(\tilde{\mathbf{W}}\mathbf{s}|\mathcal{C}_{1:K}) = \mathcal{N}(\tilde{\mathbf{W}}\mathbf{s}; \tilde{\boldsymbol{\mu}}, \mathrm{diag}(\tilde{\boldsymbol{\sigma}}^2)) \tag{7.23}$$

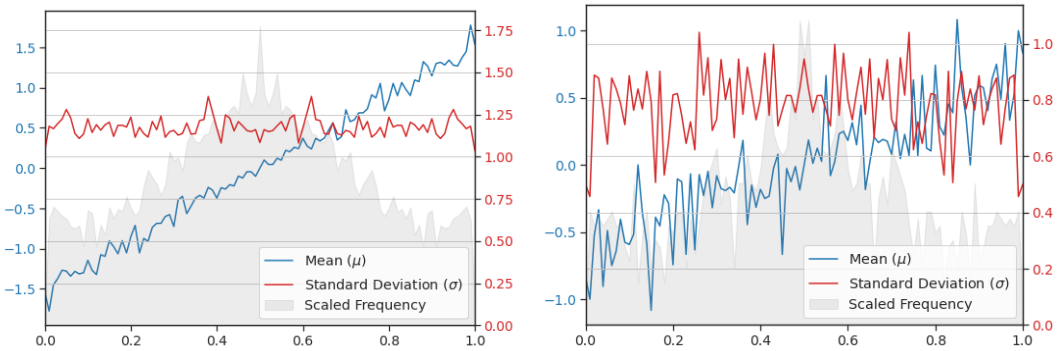
which can be rearranged (shown in Appendix A.1.1) to provide the probability for a given set of scores,

$$\mathrm{p}(s_{1:N}|C_{1:K}) = \mathcal{N}\left(\mathbf{s}; (\tilde{\mathbf{W}}^\mathsf{T}\tilde{\boldsymbol{\Sigma}}^{-1}\tilde{\mathbf{W}})^{-1}\tilde{\mathbf{W}}^\mathsf{T}\tilde{\boldsymbol{\Sigma}}^{-1}\tilde{\boldsymbol{\mu}} \ , \ (\tilde{\mathbf{W}}^\mathsf{T}\tilde{\boldsymbol{\Sigma}}^{-1}\tilde{\mathbf{W}})^{-1}\right) \tag{7.24}$$

where $\tilde{\boldsymbol{\Sigma}} = \mathrm{diag}(\tilde{\boldsymbol{\sigma}}^2)$. The maximum probability of a Gaussian distribution is at its mean, and hence Equation 7.24 demonstrates that for Gaussian experts, the maximum probability solution, $\hat{s}_{1:N}$, will be,

$$\hat{s}_{1:N} = \underset{s_{1:N}}{\arg\max}\ \mathrm{p}(s_{1:N}|\mathcal{C}_{1:K}) = (\tilde{\mathbf{W}}^\mathsf{T}\tilde{\boldsymbol{\Sigma}}^{-1}\tilde{\mathbf{W}})^{-1}\tilde{\mathbf{W}}^\mathsf{T}\tilde{\boldsymbol{\Sigma}}^{-1}\tilde{\boldsymbol{\mu}} \tag{7.25}$$

[3]For simplicity, we illustrate the case where each comparison between x_i and x_j is only performed once. In practice, one may consider the probabilities from both permutations, p_{ij} and p_{ji}, which would double the number of rows.



(a) FlanT5-3B, SummEval COH (b) Llama2-13B, TopicalChat CNT

Fig. 7.2 Expected score difference and variance given the LLM probability. For a particular LLM, all comparisons over all the contexts of the dataset are assessed. The plot presents the number of counts for each LLM probability when binned to the nearest 0.01 (denoted as frequency), as well as the true score difference and standard deviation for each bin, calculated using the gold-standard annotator labels.

Linear Gaussian Assumptions

A drawback with the current general Gaussian experts is that producing $\tilde{\boldsymbol{\mu}}$ and $\tilde{\boldsymbol{\sigma}}^2$ requires knowledge of both functions $f_\mu(p_{ij})$ and $f_\sigma(p_{ij})$ for all $p_{ij} \in [0,1]$. This is not available without a large quantity of human-annotated data for the selected task and model, which renders the approach impractical for zero-shot applications.

To enable a practical solution applicable in zero-shot settings, one can make two assumptions on the Gaussian experts: 1) that the variance is constant irrespective of the predicted probability $f_\sigma(p_{ij}) = \sigma^2$, and 2) that the mean scales linearly with the probability $f_\mu(p_{ij}) = \alpha \cdot (p_{ij} - \beta)$. As can be seen in Figure 7.2, these assumptions appear to be reasonable and hold well when LLMs make pairwise judgements (the experimental set-up will be discussed later in Section 7.3). These assumptions then simplify the method and yield similar expressions for the PoE, however since the matrix of the variances can now be expressed as $\tilde{\boldsymbol{\Sigma}} = \sigma^2 \mathbf{I}$, the marginalised score distribution can be simplified to,

$$\mathrm{p}(s_{1:N}|C_{1:K}) = \mathcal{N}\left(\mathbf{s}; \alpha \cdot (\tilde{\mathbf{W}}^\mathsf{T}\tilde{\mathbf{W}})^{-1}\tilde{\mathbf{W}}^\mathsf{T}\tilde{\boldsymbol{\mu}}\ ,\ \frac{1}{\sigma^2}(\tilde{\mathbf{W}}^\mathsf{T}\tilde{\mathbf{W}})^{-1}\right) \tag{7.26}$$

where $\tilde{\boldsymbol{\mu}} = [0, p_{ij}^{(1)} - \beta, ..., p_{ij}^{(K)} - \beta]^\mathsf{T}$ and $\mathbf{s} = [s_1, \ldots, s_N]^\mathsf{T}$. This introduces three parameters, β, α and σ. The values of α and σ are insignificant as they only determine the relative spacing and influence the subjective scale used to score the texts, and for simplicity, can be set to $\alpha = 1$ and $\sigma = 1$. β, though, is a parameter that defines the 'bias' that should be

applied to the probabilities. If there is a position bias, such that p_{ij} is typically larger than $(1-p_{ji})$, then we can increase the value of β to account for this position bias. When there is no difference in quality, a system with no positional bias would be expected to return an output probability of 0.5. Therefore, a sensible is for $\beta = 0.5$, particularly if the probabilities p_{ij} are the permutation debiased parameters (though Section 7.2.6 will detail how β can be set in situations where there may be positional bias). Using the linear Gaussian Experts yields the predicted solution,

$$\hat{\mathbf{s}} = (\tilde{\mathbf{W}}^{\mathsf{T}}\tilde{\mathbf{W}})^{-1}\tilde{\mathbf{W}}^{\mathsf{T}}\tilde{\boldsymbol{\mu}} \tag{7.27}$$

When using the full set of comparisons, it can be shown that the solution is equivalent to using average probabilities, as presented in appendix A.1.2.

7.2.3 Soft Bradley-Terry Experts

Section 7.2.1 introduced the PoE framework for comparative assessment, where a crucial design choice is the form of the individual experts, $\mathtt{p}(s_i - s_j | p_{ij})$. The previous section proposed Gaussian experts, which had favourable properties such as an intuitive marginalised score distribution, a closed-form solution, and a clear relationship with the average probability solution. Nonetheless, the framework is not limited to Gaussian experts, and other forms of experts are also possible.

To motivate an alternative expert, we consider the Bradley–Terry model [27, 346], a standard method in traditional comparative judgment. This model, which yields similar results to the Turnstone model [310] but with simpler optimisation [106], assumes that the outcome depends solely on the difference of scores, $\mathtt{P}(y_{ij}|s_i - s_j) = \sigma(s_i - s_j)$, where $\sigma(x)$ is the sigmoid function, $\sigma(x) = \frac{1}{1+\exp(-x)}$. As discussed in Section 4.3.2, the Bradley-Terry model treats the scores as parameters of the model, and optimises the likelihood of observing the comparisons $\mathcal{C}_{1:K}$ under the Bradley-Terry model. This likelihood can be defined as:

$$\mathtt{P}(\mathcal{C}_{1:K}|s_{1:N}) = \prod_{\{i,j\},y_{ij}\in\mathcal{C}_{1:K}} \sigma(s_i - s_j)^{y_{ij}}(1 - \sigma(s_i - s_j))^{1-y_{ij}} \tag{7.28}$$

where $y_{ij} \in \{0,1\}$ denotes the outcome of the comparison between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$. The predicted scores $\hat{s}_{1:N}$ are then the scores that maximise the likelihood of the observations:

$$\hat{s}_{1:N} = \arg\max_{s_{1:N}} \mathtt{P}(\mathcal{C}_{1:K}|s_{1:N}) \tag{7.29}$$

The Bradley-Terry can't directly be used as an expert as it operates with hard binary decisions and models the outcome given the score difference, not the score difference given the outcome. However, in comparative assessment, there can be a pool of judges (or, in a sports context, multiple games between teams) where each comparison is made several times. In such settings, assuming m_{ij} independent comparisons between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, let n_{ij} be the number of times $\boldsymbol{x}_i$ is judged better than $\boldsymbol{x}_j$. The probability of observing n_{ij} victories can be modelled using a binomial distribution:

$$\mathtt{P}(n_{ij}|m_{ij}, s_{1:N}) = \binom{m_{ij}}{n_{ij}} \big(\sigma(s_i - s_j)\big)^{n_{ij}} \big(1 - \sigma(s_i - s_j)\big)^{(m_{ij} - n_{ij})} \tag{7.30}$$

The likelihood of the observations can then be expressed as:

$$\mathtt{P}(\mathcal{C}_{1:K}|s_{1:N}) = \prod_{i,j} \mathtt{P}(n_{ij}|m_{ij}, s_{1:N}) \tag{7.31}$$

$$= \prod_{i,j} \binom{m_{ij}}{n_{ij}} \big(\sigma(s_i - s_j)\big)^{n_{ij}} \big(1 - \sigma(s_i - s_j)\big)^{(m_{ij} - n_{ij})} \tag{7.32}$$

As the number of comparisons becomes large and $m_{ij} \to \infty$, the empirical fraction of victories converges to the true probability, $\frac{n_{ij}}{m_{ij}} \to p_{ij}$, where p_{ij} is the true underlying probability that $\boldsymbol{x}_i$ is judged better than $\boldsymbol{x}_j$. In this limit, assuming all comparisons are made an equal number of times, the optimal scores that maximise Equation 7.32 will be equivalent to the scores which maximise:

$$\hat{s}_{1:N} = \arg\max_{s_{1:N}} \prod_{i,j} \big(\sigma(s_i - s_j)\big)^{p_{ij}} \big(1 - \sigma(s_i - s_j)\big)^{(1 - p_{ij})} \tag{7.33}$$

Therefore, we propose the soft Bradley-Terry expert, which has the form:

$$\mathtt{p}(s_i - s_j|p_{ij}) = \frac{1}{Z_{ij}} \sigma(s_i - s_j)^{p_{ij}} (1 - \sigma(s_i - s_j))^{1 - p_{ij}} \tag{7.34}$$

$$= \frac{1}{Z_{ij}} \cdot \frac{e^{p_{ij}(s_i - s_j)}}{1 + e^{(s_i - s_j)}} \tag{7.35}$$

Defined within the range $p_{ij} \in (0, 1)$, where $Z_{ij} = \pi / \sin(p_{ij}\pi)$ is a normalisation constant to ensure a valid probability density function. The soft Bradley-Terry expert directly incorporates probabilities and, instead of using the likelihood of the outcomes, models the probability of the score difference $\mathtt{p}(s_i - s_j|p_{ij})$. Therefore, these experts can be applied within the PoE framework. Although the limiting behaviour of the Bradley-Terry model (Equation 7.33)

results in estimating the true underlying scores[4] $s^*_{1:N}$, using soft Bradley-Terry experts allow us to exploit the developed optimisation techniques of Bradley-Terry to efficiently find the solution. Although no closed-form solution exists when using soft Bradley-Terry experts, similar to Bradley-Terry, Zermello's algorithm [346] can be applied to iterate the solution until convergence is reached.

7.2.4 Laplacian Approximation of Bradley-Terry

The previous section (§7.2.3) introduced the soft Bradley-Terry expert, which was an extension of the original Bradley-Terry model. The score distribution under the soft Bradley-Terry takes the form,

$$\mathrm{p}(s_{1:N}|\mathcal{C}_{1:K}) = \frac{1}{Z}\prod_{i,j\in\mathcal{C}_{1:K}} \sigma(s_i - s_j)^{p_{ij}}(1-\sigma(s_i-s_j))^{1-p_{ij}} \tag{7.36}$$

which unlike the previous PoE using Gaussian Experts (§7.2.2), is difficult to analyse. Therefore, this section considers how to approximate the distribution into a form that's more convenient to analyse. A standard approach is to apply a Laplace approximation, which approximates the original distribution as a Gaussian, $\tilde{\mathrm{p}}(s_{1:N}|\mathcal{C}_{1:K})$. Let $\mathbf{s} = [s_1, \ldots, s_N]^\mathsf{T}$ be the vector of scores. The Laplace approximation is given by:

$$\mathrm{p}(\mathbf{s}|\mathcal{C}_{1:K}) \approx \tilde{\mathrm{p}}(\mathbf{s}|\mathcal{C}_{1:K}) = \mathcal{N}\left(\mathbf{s};\, \hat{\mathbf{s}},\, \mathbf{A}^{-1}\right) \tag{7.37}$$

where $\hat{\mathbf{s}}$ is the maximum probability estimate of the the original distribution $\mathrm{p}(\mathbf{s}|\mathcal{C}_{1:K})$, and $\mathbf{A}^{-1}$ is the covariance matrix of the approximating Gaussian. The log of this approximation can be expressed as:

$$\log\tilde{\mathrm{p}}(\mathbf{s}|\mathcal{C}_{1:K}) = -\frac{1}{2}(\mathbf{s}-\hat{\mathbf{s}})^\mathsf{T}\mathbf{A}(\mathbf{s}-\hat{\mathbf{s}}) - \frac{N}{2}\log(2\pi) - \frac{1}{2}\log\left(\det\left(\mathbf{A}^{-1}\right)\right) \tag{7.38}$$

which represents a quadratic approximation of the log-likelihood around its mode. Therefore, the elements of $\mathbf{A}$ can be calculated as:

$$a_{ij} = -\frac{\partial^2}{\partial s_i \partial s_j}\log\mathrm{p}(\mathbf{s}|\mathcal{C}_{1:K})\bigg|_{\mathbf{s}=\hat{\mathbf{s}}} \tag{7.39}$$

[4] Assuming the comparative probabilities follow the assumptions of the Bradley-Terry model

By differentiating the log score distribution when using soft Bradley-Terry experts, the diagonal elements, $\mathbf{A}_{kk}$, and off-diagonal elements $\mathbf{A}_{km}$, are given by,

$$a_{kk} = \mathbb{1}(k=1) + \sum_{i,j \in \mathcal{C}} \big[\mathbb{1}(i=k) + \mathbb{1}(j=k)\big] \cdot \sigma(\hat{s}_i - \hat{s}_j) \cdot \sigma(\hat{s}_j - \hat{s}_i) \tag{7.40}$$

$$a_{km} = -1 \cdot \Big(\sum_{i,j \in \mathcal{C}} \big[\mathbb{1}(i=k)\mathbb{1}(j=m) + \mathbb{1}(j=k)\mathbb{1}(i=m)\big] \cdot \sigma(\hat{s}_i - \hat{s}_j) \cdot \sigma(\hat{s}_j - \hat{s}_i) \Big) \tag{7.41}$$

Similar to the Gaussian expert, a static expert is introduced on the first element to avoid the redundant axis. The form of the inverse covariance matrix for the Laplace approximation is closely related to the form of the inverse covariance matrix for the linear Gaussian experts, $\mathbf{A} = \tilde{\mathbf{W}}^\mathsf{T}\tilde{\mathbf{W}}$, which has form,

$$a_{kk} = \mathbb{1}(k=1) + \sum_{i,j \in \mathcal{C}} \big[\mathbb{1}(i=k) + \mathbb{1}(j=k)\big] \tag{7.42}$$

$$a_{km} = -1 \cdot \sum_{i,j \in \mathcal{C}} \big[\mathbb{1}(i=k)\mathbb{1}(j=m) + \mathbb{1}(i=m)\mathbb{1}(j=k)\big] \tag{7.43}$$

The only difference between the inverse covariance matrix when using linear Gaussian experts and the Laplacian approximation of soft Bradley-Terry experts is the multiplication by the term $\sigma(\hat{s}_i - \hat{s}_j) \cdot \sigma(\hat{s}_j - \hat{s}_i)$.

7.2.5 Comparison Selection

The previous theory detailed how to determine the predicted scores $\hat{s}_{1:N}$ given a set of observed comparisons $\mathcal{C}_{1:K}$. While these comparisons have been assumed to be randomly selected from the set of all possible comparisons, an alternative approach is to strategically select them. By prioritising comparisons that provide the most information, it might be possible to achieve better assessment performance while maintaining the same number of comparisons. One of the useful properties of using the Products of Experts is that one has an analytical form of the distribution over the final predictions. This allows for a selection routine where comparisons are selected to maximise the probabilities of the observations. Assuming the marginalised score distribution has a Gaussian form,

$$\mathrm{p}(\mathbf{s}|\mathcal{C}_{\mathbf{1:K}}) = \mathcal{N}(\mathbf{s};\, \boldsymbol{\mu}^*,\, \boldsymbol{\Sigma}^*) \tag{7.44}$$

The solution with the highest probability is the mean $\boldsymbol{\mu}^*$. The associated probability is,

$$\mathtt{p}(\boldsymbol{\mu}^*|\mathcal{C}_{1:K}) = \frac{\sqrt{\mathtt{det}\left(\boldsymbol{\Sigma}^{*-1}\right)}}{(2\pi)^{\frac{N}{2}}} \tag{7.45}$$

To minimise uncertainty, one may want to select the set of comparisons which maximise the expected probability of the solution. The comparison matrix $\tilde{\mathbf{W}}^*$ that maximises the probability of the solution is equivalent to the comparison matrix that maximises the determininant of the inverse covariance matrix,

$$\tilde{\mathbf{W}}^* = \arg\max_{\tilde{\mathbf{W}}} \mathtt{p}(\boldsymbol{\mu}^*|\mathcal{C}_{1:K}) \equiv \arg\max_{\tilde{\mathbf{W}}} \mathtt{det}(\boldsymbol{\Sigma}^{*-1}) \tag{7.46}$$

Although it may not be possible to find the exact solution of the expression, one can run a greedy approach of iteratively estimating the comparisons which will greedily maximise the probability of the solution. For the Linear Gaussian Experts, it was shown that $\mathbf{A} = \tilde{\mathbf{W}}^\mathsf{T}\tilde{\mathbf{W}}$ (Equation 7.26). Given the current comparisons $\tilde{\mathbf{W}}$, adding an additional comparison (i, j) is equivalent to adding an extra row $\mathbf{r} \in \mathbb{R}^N$ to $\tilde{\mathbf{W}}$, where $\mathbf{r}_i = 1$, $\mathbf{r}_j = -1$ and $\mathbf{r}_l = 0 \ \ \forall l \neq i, j$. By applying the matrix determinant lemma [91], the determinant of the matrix after adding the row $\mathbf{r}$ can be expressed as:

$$\mathtt{det}\left([\tilde{\mathbf{W}};\mathbf{r}]^\mathsf{T}[\tilde{\mathbf{W}};\mathbf{r}]\right) = \mathtt{det}(\tilde{\mathbf{W}}^\mathsf{T}\tilde{\mathbf{W}} + \mathbf{r}\mathbf{r}^\mathsf{T}) = \mathtt{det}(\tilde{\mathbf{W}}^\mathsf{T}\tilde{\mathbf{W}})(1 + \mathbf{r}^\mathsf{T}(\tilde{\mathbf{W}}^\mathsf{T}\tilde{\mathbf{W}})^{-1}\mathbf{r}) \tag{7.47}$$

The sparse form of $\mathbf{r}$ allows us to simplify the expression, such that when using linear Gaussian experts, the next greedy optimal comparison can be selected as,

$$\hat{i}, \hat{j} = \arg\max_{i,j} (\mathbf{r}^\mathsf{T}(\tilde{\mathbf{W}}^\mathsf{T}\tilde{\mathbf{W}})^{-1}\mathbf{r}) \tag{7.48}$$

$$\equiv \arg\max_{i,j} \left(\left[\mathbf{A}^{(k)^{-1}}\right]_{ii} + \left[\mathbf{A}^{(k)^{-1}}\right]_{jj} - 2 \cdot \left[\mathbf{A}^{(k)^{-1}}\right]_{ij} \right) \tag{7.49}$$

Note the covariance matrix $\mathbf{A}^{(k+1)}$ can be updated efficiently from $\mathbf{A}^{(k)}$ (See Appendix A.1.3). Furthermore, the greedy-selected comparisons do not depend on any of the observed LLM probabilities p_{ij} and depend only on the number of candidate texts N and the number of comparisons K.

For the Laplacian approximation of the soft Bradley-Terry experts, the inverse covariance matrix has elements as described in Equations 7.40 and 7.41. Hence, one can note that adding

a further comparison (i, j) has influence on $\mathbf{A}^{(k)}$,

$$\mathbf{A}^{(k+1)} = \mathbf{A}^{(k)} + \sigma(\hat{s}_i - \hat{s}_j) \cdot \sigma(\hat{s}_j - \hat{s}_i) \cdot \mathbf{r}\mathbf{r}^\mathsf{T} \tag{7.50}$$

By again applying the matrix determinant lemma, one can express the determinant after adding the next comparison as follows:

$$\begin{aligned} \det\left(\mathbf{A}^{(k+1)}\right) &= \det(\mathbf{A}^{(k)} + \sigma(\hat{s}_i - \hat{s}_j) \cdot \sigma(\hat{s}_i - \hat{s}_j) \cdot \mathbf{r}\mathbf{r}^\mathsf{T}) && (7.51) \\ &= \det(\mathbf{A}^{(k)}) \left(1 + \sigma(\hat{s}_i - \hat{s}_j) \cdot \sigma(\hat{s}_j - \hat{s}_i) \cdot (\mathbf{r}^\mathsf{T} {\mathbf{A}^{(k)}}^{-1} \mathbf{r})\right) && (7.52) \end{aligned}$$

Maximising the the expression is equivalent to maximising,

$$\begin{aligned} \hat{i}, \hat{j} &= \arg\max_{i,j} \left(\sigma(\hat{s}_i - \hat{s}_j) \cdot \sigma(\hat{s}_j - \hat{s}_i) \cdot \mathbf{r}^\mathsf{T} {\mathbf{A}^{(k)}}^{-1} \mathbf{r}\right) && (7.53) \\ &= \arg\max_{i,j} \left(\sigma(\hat{s}_i - \hat{s}_j) \cdot \sigma(\hat{s}_j - \hat{s}_i) \left(\left[\mathbf{A}^{(k)^{-1}}\right]_{ii} + \left[\mathbf{A}^{(k)^{-1}}\right]_{jj} - 2 \cdot \left[\mathbf{A}^{(k)^{-1}}\right]_{ij}\right)\right) && (7.54) \end{aligned}$$

Therefore selecting comparisons under the Laplacian approximation of the soft Bradley-Terry model leads to a selection process that is similar to that when using linear Gaussian expert, but with an additional multiplication term of $\sigma(\hat{s}_i - \hat{s}_j) \cdot \sigma(\hat{s}_j - \hat{s}_i)$. Intuitively, this selection mechanism favours close comparisons, such that the comparisons between texts of similar quality are expected to reveal the most information. This approach is now also dynamic and depends on all the previous LLM probabilities. Although this may yield a more powerful selection scheme, this also means that the approach has to be embedded in the LLM comparative assessment framework. Furthermore, the score predictions $\hat{s}_{1:N}$ must be solved for each next new comparison (which is solved through iterating the solution until convergence) and the efficient matrix updates (§A.1.3) are no longer applicable. Therefore, this selection mechanism is more computationally expensive to run than that of the linear Gaussian selection scheme, and practically, the effectiveness of this approach will depend on the tradeoff between this computational expense and the cost of an LLM forward pass.

7.2.6 Mitigating Bias

As previously discussed in Section 7.1.4, LLMs can have inconsistent outputs where $p_{ij} \neq (1 - p_{ji})$ and may have positional bias such that the system prefers one position over another. An option to overcome this would be to perform permutation debiasing, where the probabilities from both permutations are combined into a single probability $\tilde{p}_{ij} = \frac{1}{2} \cdot (p_{ij} + (1 - p_{ji}))$. This resulting probability is then guaranteed to be consistent across both permutations.

An alternative would be to directly accommodate for bias within the experts. A simple method to do so would be to introduce a bias parameter γ that shifts the experts such that $\mathtt{p}_\gamma(s_i - s_j|p_{ij}) = \mathtt{p}(s_i - s_j - \gamma|p_{ij})$. The extension of the Bradley-Terry model that accounted for home advantage (§4.3.2) had a similar form:

$$\mathtt{P}(\boldsymbol{x}_i \succ \boldsymbol{x}_j) = \sigma(s_i - s_j + \gamma) = \frac{\exp(s_i - s_j + \gamma)}{1 + \exp(s_i - s_j + \gamma)} \tag{7.55}$$

where γ is treated as a further parameter of the model that can also be optimised when finding the maximum likelihood solution [71]. However within the PoE framework, we consider determining the values of γ by noting that the expected score difference between two randomly sampled texts is zero, $\mathbb{E}[s_i - s_j] = 0$. This section derives methods of setting γ for the Gaussian and Bradley-Terry experts, such that the experts can automatically account for positional bias.

The linear Gaussian experts (§7.28) introduced the parameter β, which defined the probability offset. Setting γ is equivalent to setting the β parameter, and it can be shown (see Appendix A.1.4) that the value of β that mitigates positional bias is the average LLM comparative probability:

$$\beta = \mathbb{E}_{\{\boldsymbol{c},\boldsymbol{x}_i,\boldsymbol{x}_j\}}\left[\mathtt{P}(\boldsymbol{x}_i \succ \boldsymbol{x}_j|\boldsymbol{c};\boldsymbol{\theta})\right] \tag{7.56}$$

I.e. $\beta = \mathbb{E}[p_{ij}] \approx \frac{1}{K}\sum_{k=1}^{K} p_{ij}^{(k)}$, where $p_{ij}^{(k)}$ are probabilistic samples from the LLM judge.

For the soft Bradley-Terry expert, the expectation $\mathbb{E}[s_i - s_j] = 0$ becomes undefined due to infinities; if the model returns a probability of 1, this implies that $\boldsymbol{x}_i$ always wins against $\boldsymbol{x}_j$ and the true score difference is expected to be infinite. However, for experts that are unstable or for which the expectation is analytically intractable, one can instead ensure that the mode of the expert's distribution occurs when the score difference is 0. For the soft Bradley-Terry expert, this yields the solution:

$$\gamma = -1 \cdot \log\left(\frac{\mathbb{E}_{\{\boldsymbol{c},\boldsymbol{x}_i,\boldsymbol{x}_j\}}\left[\mathtt{P}(\boldsymbol{x}_i \succ \boldsymbol{x}_j|\boldsymbol{c};\boldsymbol{\theta})\right]}{1 + \mathbb{E}_{\{\boldsymbol{c},\boldsymbol{x}_i,\boldsymbol{x}_j\}}\left[\mathtt{P}(\boldsymbol{x}_i \succ \boldsymbol{x}_j|\boldsymbol{c};\boldsymbol{\theta})\right]}\right) \tag{7.57}$$

I.e., $\gamma = -1 \cdot \log\left(\frac{\mathbb{E}[p_{ij}]}{1+\mathbb{E}[p_{ij}]}\right) = -\text{logit}(\mathbb{E}[p_{ij}]) \approx \text{logit}\left(\frac{1}{K}\sum_{k=1}^{K} p_{ij}^{(k)}\right)$ (derived in detail in Appendix A.1.4).

7.3 Experimental Results

This chapter introduced LLM comparative assessment, a reference-free zero-shot NLG evaluation approach that can be easily generalised to new domains. The experiments in this section will investigate 1) the degree to which bias may manifest in the assessment, 2) the general effectiveness of the approach compared to alternative bespoke and zero-shot approaches, and 3) whether the approach can be made more inference-efficient by framing it within a product of experts framework.

Datasets

To ensure the effectiveness and generalisability of the approach, we consider standard NLG assessment datasets (§4.2.1) covering a wide range of NLG generation tasks, domains and attributes. The following datasets with gold-standard human assessment ranks are:

- **SummEval** [67] is a summary evaluation benchmark of 100 passages, each with 16 machine-generated summaries. Each summary is evaluated for coherency (`COH`), consistency (`CON`), fluency (`FLU`), and relevancy (`REL`).
- **TopicalChat** with the USR annotations [215] is for benchmarking dialogue evaluation. It includes 60 dialogue contexts and six system responses per context. These responses are assessed on coherency (`COH`), continuity (`CNT`), engagingness (`ENG`), and naturalness (`NAT`).
- **WebNLG** [81] is for benchmarking data-to-text evaluation methods. It contains 223 semantic triple groups, each paired with outputs from 8 triple-to-text generation systems. These texts are evaluated for fluency (`FLU`), grammar (`GRA`) and semantic equivalence (`SEM`).
- **CMCQRD** [223] is a dataset for evaluating the difficulty of multiple choice reading comprehension questions. The dataset contains 658 questions, each annotated with perceived question difficulty.
- **HANNA** [36] is a story evaluation evaluation dataset, containing 1056 machine-generated stories annotated by humans on coherency (`COH`), complexity (`CMP`) and surprisingness (`SUR`).

Evaluation

The datasets $\mathcal{D}$ contains a set of unique input contexts $\boldsymbol{c}$, N system-generated responses $\boldsymbol{x}_{1:N}$ per context and corresponding manual assessment ranks $r_{1:N}$,

$$\mathcal{D} = \left\{ \boldsymbol{c}^{(j)}, \boldsymbol{x}_{1:N}^{(j)}, r_{1:N}^{(j)} \right\}_{j=1,\ldots,M} \tag{7.58}$$

where M is the number of unique contexts and N is the number of different generative systems. Given the system assessment rankings. The NLG assessment systems are evaluated by measuring the average response-level Spearman correlations with the ground-truth rankings:

$$\rho = \frac{1}{M} \sum_{j=1}^{M} \text{SCC} \left(\hat{r}_{1:N}^{(j)}, r_{1:N}^{(j)} \right) \tag{7.59}$$

The Spearman correlation is used over the Pearson correlation to better suit the ranking tasks prevalent in NLG assessment. Further, response-level correlations are calculated instead of system-level correlations, as the latter is an easier task that often results in higher correlations, often near 1. Using response-level correlations allows us to discern meaningful differences in the effectiveness of various assessment methods.

Models

Consistent with the results of the previous chapter, we similarly use both FlanT5 [42] and Llama2-chat [314] as the LLM judges in LLM comparative assessment, as well as Mistral-7B [139]. These systems were the most capable open-source instruction-following LLMs at the point of conducting experiments in March 2024 and also represent a range of model architectures and training approaches. This diversity allows us to evaluate the generalisability of our findings across different LLM and training paradigms, allowing us to evaluate effectiveness and identify potential biases and limitations across different model types.

Baseline Approaches

To validate the effectiveness of comparative assessment against other NLG evaluation methods, we employ a range of standard NLG assessment systems as baselines. These systems, previously detailed in Section 4.2, include:

- **BERTScore** [349], which evaluates summaries based on their similarity to human reference summaries, measured in the embedding space.

- **UniEval** [358], which convert NLG evaluation into Boolean QA. This method uses pre-defined schemes for selected aspects (e.g., coherence) and generates synthetic data to fine-tune a T5 system for NLG assessment. References are used for particular aspects (e.g. relevancy), and schemes/systems are bespoke for a particular attribute. We use the 'continual' system where the same model has been continually trained for all the attributes (which is the setting that demonstrates the best performance.)

- **MQAG** [203], which represents information using multiple-choice questions and measures the consistency in answer distribution between conditioning on the summary and the context. The method is used reference-free.

- **GPTScore** [76], which evaluates texts using conditional language model probabilities. By conditioning the language model on a task instruction and the context, GPTScore assumes that high-quality texts will be assigned a higher probability than poor-quality texts.

- **Prompt-Scoring**, which, for a particular attribute, asks an LLM to assess the response quality between 1-10. Prompt-scoring is used as the main baseline to compare comparative assessment as both are zero-shot prompting approaches. The prompt-scoring prompts are also designed to be simple and not overly engineered, e.g.:

$$\mathcal{P}(\boldsymbol{x},\boldsymbol{c}) = \text{`}\textit{Provide a score between 1 and 10 that measures the text's coherency.}$$
$$\textit{Context: } \boldsymbol{<c>}$$
$$\textit{Text: } \boldsymbol{<x>}\text{'}$$

- **G-Eval** [182], which is an extension of prompt-scoring, calculates the expected score over a score range (e.g. 1-5 normalised by their probabilities). We use the same detailed prompts that were used in the original paper.

7.3.1 Analysis of Positional Bias

The first set of experiments in this chapter analyses whether, similar to multiple-choice question answering (§6.5.2), LLM comparative assessment is impacted by positional bias. We replicate the experiments of Section 6.5.2 for comparative assessment, evaluating the accuracy, permutation sensitivity (Equation 7.8) and positional bias (Equation 7.10) for individual pairwise comparisons.

	COH			CON			FLU			REL		
	acc	pb	ps	acc	pb	ps	acc	pb	ps	acc	pb	ps
FlanT5-3B												
baseline	69.2	0.16	0.20	81.1	0.06	0.12	62.5	0.12	0.17	64.5	0.10	0.17
prior-matching	70.5	0.03	0.15	80.5	0.03	0.11	64.3	0.03	0.13	64.1	0.02	0.15
perm-debias	71.7	0.00	0.00	82.0	0.00	0.00	65.7	0.00	0.00	65.4	0.00	0.00
FlanT5-11B												
baseline	61.6	0.42	0.42	70.5	0.37	0.38	55.6	0.44	0.44	62.8	0.39	0.39
prior-matching	67.3	0.02	0.16	77.8	0.00	0.14	58.9	0.03	0.16	65.3	0.05	0.16
perm-debias	68.9	0.00	0.00	79.7	0.00	0.00	61.4	0.00	0.00	67.0	0.00	0.00
Llama-7B												
baseline	62.8	0.09	0.30	64.1	0.45	0.49	58.0	0.33	0.42	58.3	0.53	0.56
prior-matching	62.5	0.07	0.29	62.8	0.13	0.29	59.6	0.10	0.30	62.0	0.01	0.27
perm-debias	64.8	0.00	0.00	66.2	0.00	0.00	59.7	0.00	0.00	65.7	0.00	0.00
Llama-13B												
baseline	62.3	0.36	0.38	71.8	0.18	0.23	58.9	0.56	0.56	63.8	0.38	0.40
prior-matching	65.8	0.06	0.23	72.9	0.01	0.17	61.6	0.03	0.24	65.5	0.11	0.24
perm-debias	68.6	0.00	0.00	73.1	0.00	0.00	63.5	0.00	0.00	66.3	0.00	0.00

Table 7.1 Accuracy (acc), permutation bias (pb) and permutation sensitivity (ps) for various LLMs when prompted for comparative assessment on SummEval.

Table 7.1 presents an analysis of the performance and positional bias when various LLMs are applied for LLM comparative assessment on different attributes of SummEval, a dataset for Summary assessment. We observe that the LLMs are able to determine which of two texts better demonstrates an assessed attribute, with accuracies usually within the range of 60-80%, which is well above the random performance of 50%. However, LLM judges exhibit considerable positional bias. For example, on coherency, FlanT5-11B has a positional bias of 0.42, which indicates that over all comparisons, one position is preferred 42% of the times more than the other (e.g. 71% vs 29%). Positional bias can also be observed on most models and attributes, which highlights that the systems are sensitive to the input orderings, and positional bias may be a vital consideration when applying LLM comparative assessment.

Additionally, systems may follow task invariances for some tasks, but the same invariance may not be maintained for other tasks. FlanT5-3B and FlanT5-11B were previously shown to be robust to changes in input orderings, having low permutation sensitivity and positional bias when applied to MCQA tasks (§6.5.2). This was hypothesised to be attributed largely to the pre-training tasks, where MCQA datasets were present during the instruction-tuning stage of FlanT5. However, for LLM comparative assessment, FlanT5 demonstrates notable permutation sensitivities and positional biases. This is particularly noticeable for FlanT5-11B, which shows biases comparable to those of Llama2. This highlights that though systems

may account for positional invariance in one domain, this capability may not extend to other domains where positional invariance is also relevant.

Further, as observed in MCQA tasks, permutation debiasing yields large accuracy improvements for many tasks. Permutation debiasing can improve accuracies by 5-10 percentage points, which is seen across various assessment attributes and models. With two options, permutation debiasing is simple and only requires averaging the probabilities from both permutations of the comparison (e.g. A vs B and B vs A) and is not as computationally expensive as in MCQA tasks. The results highlight that bias is a crucial factor that must be considered and addressed in LLM comparative assessment. This can be achieved by performing permutation debiasing, where both permutations are computed and averaged. Alternatively, when individual decisions are used, it becomes important to consider methods to account for positional bias, motivating the experts proposed in Section 7.2.6 for the PoE framework of LLM comparative assessment.

Debiasing via student distillation

		COH		CON		FLU		REL	
	type	acc	ps	acc	ps	acc	ps	acc	ps
Teachers									
permutation debiased white-box		68.9	0.00	79.7	0.00	61.4	0.00	67.0	0.00
biased white-box		61.6	0.42	70.5	0.38	55.6	0.44	62.8	0.39
expected biased black-box		58.5	-	65.5	-	54.3	-	58.8	-
Students									
RoBERTa-base (110M)	d	61.5	0.05	70.5	0.07	61.2	0.05	60.9	0.06
RoBERTa-base (110M)	ec	66.4	0.04	71.7	0.06	61.6	0.03	61.9	0.05
DeBERTa-base (110M)	d	62.6	0.03	67.1	0.04	62.1	0.03	63.0	0.05
DeBERTa-base (110M)	ec	66.0	0.03	71.1	0.04	64.1	0.03	62.1	0.06
RoBERTa-large (330M)	d	64.8	0.05	67.6	0.06	62.7	0.05	62.1	0.05
RoBERTa-large (330M)	ec	66.7	0.05	72.0	0.05	63.3	0.04	63.6	0.05
DeBERTa-large (330M)	d	65.1	0.04	71.5	0.04	64.9	0.03	63.2	0.03
DeBERTa-large (330M)	ec	66.1	0.03	70.9	0.02	64.8	0.03	63.3	0.04

Table 7.2 Performance of a student trained to emulate the debiased teacher on SummEval. Reported are task accuracy (acc) and permutation sensitivity (ps). The students are either directly distilled (d, §6.4.1) or trained to correct the distribution of a single biased black-box teacher decision (ec, §6.4.2). FlanT5-11B is used as the teacher.

To investigate the generalisability of the student-teacher distillation framework (§6.4) beyond MCQA, we examine whether student models can be trained to replicate the permutation-

debiasing decisions in LLM comparative assessment. The same experimental set-up and student systems are used as the results in Section 6.5.3, although for SummEval, we now take the first 80 contexts as the training data and the last 20 contexts as the test set. For comparative assessment, permutation debiasing now only requires two calls, although using a smaller encoder-teacher yields considerable computational efficiency savings in distillation.

When applied to LLM comparative assessment, the distilled students are shown to be highly effective, performing similarly to the debiased teacher. Unlike in MCQA, where distilled students struggle to model the teacher's distribution and error correction is required to achieve good performance, the distilled students achieve promising performance for comparative assessment. This may be because comparative assessment is simpler than MCQA and only involves binary decisions rather than comparing several options. Hence, the capacity of the student model may alone be sufficient for the pairwise judgements. For example, the distilled DeBERTa-large student achieves an accuracy of 65.1 and permutation sensitivity of 0.04 when assessing coherency, while the biased white-box teacher has an accuracy of 61.6 and permutation sensitivity of 0.42. This highlights the benefits of the debiased students, which perform better and are more robust to permutations. Similar observations are seen across models and attributes, where notably, the student models have an order of magnitude fewer parameters and do not use any labelled data.

Additionally, the error correction students perform better than the distillation students. Performance is significantly better than the sampled black box teacher decisions, with the DeBERTa-large error-correction student achieving accuracies 5-10% above the expected biased black-box performance. Lastly, we observe that the size and ability of the students can be an important factor, with DeBERTa typically outperforming RoBERTa and RoBERTa-large outperforming RoBERT-base. However, for DeBERTa, the performance appears to saturate with size, with the error-correcting DeBERTa-base achieving accuracies within 1.2% of DeBERTa-large for all attributes.

7.3.2 Effectiveness of LLM comparative Assessment

The previous results demonstrated that LLMs can perform comparative assessments with reasonable consistency with human judgements and that positional bias is an important factor to consider. For NLG assessment tasks, the primary objective is not only to determine which of two texts is better but to rank all texts from best to worst for an assessed attribute. The system must conduct numerous pairwise comparisons between texts and then aggregate these results to generate the final predicted rankings. Our next set of experiments examines the effectiveness of LLM comparative assessment for NLG evaluation, comparing performance to existing baselines and, in particular, the prompt-scoring approach. In these experiments,

we focus on exploring the potential of the approach rather than its computational efficiency. Consequently, all $N \cdot (N-1)$ comparisons are computed, which will also implicitly correct for bias since both permutations are considered for each comparison.

Approach	COH	CON	FLU	REL
Baselines				
BERTScore (w/ Ref)	25.9	19.7	23.7	34.7
MQAG	17.0	28.8	19.3	16.6
UniEval (continual)	57.5	44.6	44.9	42.6
GPTScore FlanT5-3B	47.0	43.6	42.1	34.4
GPTScore FlanT5-11B	45.6	43.8	42.4	34.3
ChatGPT scoring	45.1	43.2	38.0	43.9
Prompt-Scoring				
FlanT5-3B	**14.5**	**19.8**	**3.9**	**15.2**
FlanT5-11B	0.7	11.2	3.2	5.7
Llama2-chat-7B	8.6	9.0	1.8	7.8
Llama2-chat-13B	9.9	6.9	1.2	9.2
G-Eval				
FlanT5-3B	10.5	29.1	9.8	23.8
FlanT5-11B	19.2	29.3	20.7	35.8
Llama2-chat-7B	28.2	29.4	**23.0**	27.4
Llama2-chat-13B	**53.2**	**33.7**	16.5	**38.3**
Comparative Assessment				
FlanT5-3B	**51.2**	**47.1**	**32.5**	44.8
FlanT5-11B	44.2	37.2	30.2	43.4
Llama2-chat-7B	27.9	24.6	20.2	35.6
Llama2-chat-13B	40.9	39.9	30.8	**45.3**

Table 7.3 Spearman correlation coefficient, SCC, for **SummEval**, averaged over both prompts per system (for prompt-scoring and comparative). ChatGPT performance is quoted from Wang et al. [326], which use more detailed scoring prompts.

Table 7.3 analyses the effectiveness of LLM comparative assessment for summary evaluation on the **SummEval** dataset. The results demonstrate that comparative assessment is an effective NLG assessment approach, with LLMs comparative assessment providing rankings that correlate well with the manual gold-standard ranks. With comparative assessment, the moderate-sized LLMs achieve Spearman correlations usually within the range of 30-50 SCC, with the approach generalising across models and attributes. A further observation is that moderate-sized LLMs struggle to predict independent assessment scores through zero-shot prompt-scoring. FlanT5-3B, which achieved the best prompt-scoring performance, achieved a Spearman correlation of 19.8 SCC when assessing consistency and correlations between 3-15 SCC for the other attributes. When used for comparative assessment, the same LLM achieved a correlation of 47.1 SCC for consistency assessment and 30-52 SCC for the other attributes.

Interestingly, prompt-scoring is more effective with larger systems, as demonstrated by ChatGPT, which achieves correlations between 38-45 SCC. This perforance gap suggests that prompt-scoring is an emergent ability only larger LLMs can perform effectively. However, by framing evaluation within the simpler intermediate task of judging two texts, comparative assessment appears to better harness the abilities of less powerful models, with FLanT5-3B comparative assessment even outperforming ChatGPT prompt-scoring in 3/4 attributes.

Additionally, G-Eval, which uses task-specific detailed prompts and continuous scores, yields significant improvements over prompt-scoring. However, in 12/16 settings, LLM comparative assessment outperforms G-Eval, and often by quite large margins. Beyond the prompt-scoring baselines, comparative assessment performs competitively against other bespoke baselines. The best comparative assessment LLM (FlanT5-3B) achieves correlations competitive with other zero-shot methods, outperforming the equivalent GPTScore by an average SCC of 2.1 across all attributes. Performance is also substantially higher than BERTScore which uses referenced-based evaluation (average increase of 17.9 SCC), as well as MQAG which is designed to assess information consistency using multiple choice question answering (average increase of 23.5 SCC). Although UniEval does perform better than LLM comparative assessment (on average 3.5 SCC higher), UniEval was designed for bespoke tasks and is fine-tuned on synthetic data created for particular attributes. When applied to out-of-domain settings, UniEval can have considerable performance degradation (shown later in Table 7.5), while in contrast, comparative assessment is zero-shot and general.

Approach	COH	CNT	ENG	NAT
Baselines				
GPTScore GPT3	56.9	32.9	49.6	52.4
ChatGPT scoring	54.7	57.7	37.9	58.0
UniEval (continual)	61.3	-	60.5	44.4
Prompt-Scoring				
FlanT5-3B	**31.9**	**28.8**	17.4	**23.7**
FlanT5-11B	15.3	8.0	4.3	24.3
Llama2-chat-7B	16.4	17.0	20.6	21.4
Llama2-chat-13B	21.7	19.9	**31.4**	23.2
Comparative Assessment				
FlanT5-3B	49.4	**49.4**	37.3	47.4
FlanT5-11B	**54.3**	42.2	54.7	**54.2**
Llama2-chat-7B	28.9	33.7	36.1	30.3
Llama2-chat-13B	32.4	43.2	**55.5**	33.5

Table 7.4 Spearman correlations for NLG assessment approaches on **TopicalChat**.

Approach	FLU	GRA	SEM
Baselines			
BLEU	36.3	34.7	50.3
METEOR	44.3	42.9	62.7
UniEval (continual)	21.7	16.3	-
Prompt-Scoring			
FlanT5-3B	**30.8**	**32.7**	**38.5**
FlanT5-11B	-0.7	6.9	20.8
Llama2-chat-7B	3.8	2.4	17.0
Llama2-chat-13B	1.8	0.5	5.6
Comparative Assessment			
FlanT5-3B	40.6	41.4	12.8
FlanT5-11B	41.4	44.8	52.4
Llama2-chat-7B	22.9	37.8	-5.3
Llama2-chat-13B	**44.9**	**45.1**	**53.5**

Table 7.5 Spearman correlations for NLG assessment approaches on **WebNLG**.

To further investigate the applicability of LLM comparative assessment in general settings beyond summarisation, we analyse the performance of comparative assessment on both dialogue response generation (TopicalChat) and data-to-text generation (WebNLG). Table 7.4 presents the findings for **TopicalChat**, which align with those observed for SummEval. LLM comparative assessment again outperforms the correlations seen from prompt-scoring, where using prompt-scoring FlanT5-11B achieves an average of 13.0 SCC (across all attributes), while comparative assessment achieves a much stronger correlation of 51.4 SCC. This trend holds across all models, with improvements of 20.4 SCC for FlanT5-3B, 38.4 SCC for FlanT5-11B, 13.4 for Llama2-7B, and 17.1 SCC for Llama2-13B.

For **WebNLG**, which evaluates data-to-text generation, the context is highly abstract and consists of a list of triples in the form of (object, relation, subject). This makes semantic assessment challenging, as the LLM must parse and comprehend these triples. Table 7.5 highlights that understanding semantic triples is an emergent ability in LLMs. For grammar and fluency, the correlations are similar across systems, with FlanT5-3B achieving a correlation of 41.4 SCC for grammar and FlanT5-11B 44.8 SCC. However, for semantic understanding, the larger models significantly outperform the smaller ones. FlanT5-3B and LLama2-7B achieve correlations of 12.8 SCC and -5.3 SCC respectively, while FlanT5-11B and Llama2-13B achieve correlations of 52.4 SCC and 53.5 SCC, respectively.

7.3.3 PoE Framework for Comparative Assessment

The previous experiments validated the effectiveness of comparative assessment. However, it was previously noted that using all $N(N-1)$ comparisons can be computationally inefficient and cause limitations when applied to practical use cases where there are many assessed texts. To deal with this limitation, we introduced the PoE framework for LLM comparative assessment (§7.2), which elegantly leverages all available information from a subset of completed comparisons to get efficient scores. Several different methods of mapping comparisons to scores are considered, categorised into binary decision-based (win-rato, BT, PoE-g) or probability-based (avg-prob, PoE-BT, PoE-g):

- **win-ratio**: This calculates the number of comparisons won by the respective text, which is used as the quality score.
- **BT**: This approach finds the scores that maximise the likelihood of the comparisons under the Bradley-Terry model [27], where the solution is found by Zermelo [346] with a convergence threshold of $1e^{-4}$.

- **avg-prob**: This computes the average probability associated with the text over all of its comparisons
- **PoE-BT**: Our proposed PoE framework using the soft Bradley-Terry experts, which is the Bradley-Terry model extended to soft probabilities (§7.2.3). Models are similarly found using an updated version of Zermelo to handle soft probabilities.
- **PoE-g** Our proposed PoE framework using the Gaussian expert with the linear mean and constant variance assumptions (§7.2.2). This approach has an analytic solution.
- **PoE-g-hard** An extension of the POE-gaussian framework, however, using hard binary decisions and not soft probabilities.

For each context, we do pairwise comparisons using the LLM on the full set of $N \cdot (N-1)$ comparisons. We then randomly select a subset of K comparisons from these outcomes. The first set of experiments considers the symmetric setup, where the probability for each comparison is calculated using permutation debiasing (i.e. averaging both permutations), though later experiments will consider the impact of bias in the non-symmetric setup. This process is repeated 100 times for a particular number of total comparisons, K, and we calculate both the mean and standard deviation of performance over the entire dataset.

7.3.4 Summeval

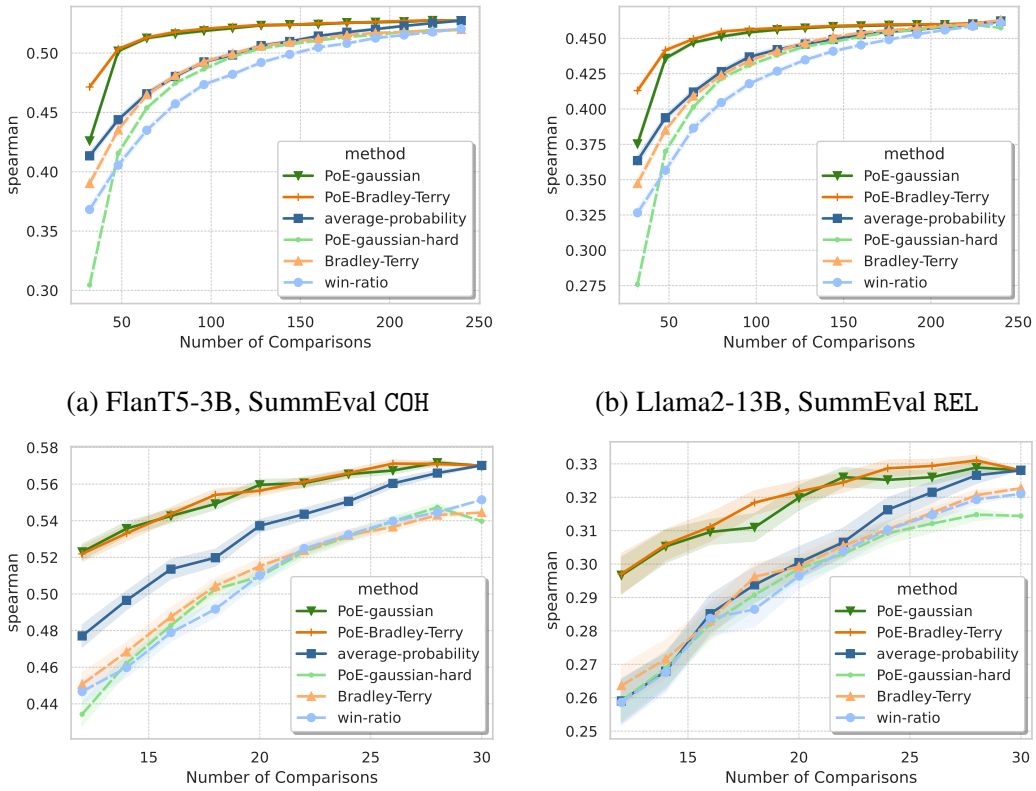
		decisions only			probabilities		
system	K	win-ratio	BT	PoE-g-hard	avg-prob	PoE-BT	PoE-g
Llama2-7B	48	21.6±0.8	23.4±0.7	22.5±0.7	24.0±0.7	26.8±0.5	26.6±0.5
	240	27.8±0.0	27.9±0.0	27.6±0.0	28.4±0.0	28.4±0.0	28.4±0.0
Llama2-13B	48	30.8±0.7	33.1±0.7	31.6±0.7	33.7±0.6	37.7±0.4	37.3±0.4
	240	39.3±0.0	39.3±0.0	39.2±0.0	39.3±0.0	39.3±0.0	39.3±0.0
Mistral-7B	48	29.7±0.8	31.9±0.7	30.5±0.6	31.1±0.7	33.2±0.6	32.8±0.6
	240	38.1±0.0	38.1±0.0	38.0±0.0	37.7±0.0	37.7±0.0	37.7±0.0
Flant5-3B	48	34.1±0.8	36.6±0.6	34.9±0.7	38.4±0.6	42.6±0.4	42.4±0.4
	240	43.6±0.0	43.6±0.0	43.4±0.0	44.3±0.0	44.3±0.0	44.3±0.0
Flant5-11B	48	31.2±0.8	33.4±0.7	32.0±0.7	34.7±0.7	38.5±0.4	38.4±0.4
	240	40.0±0.0	40.0±0.0	39.7±0.0	40.5±0.0	40.5±0.0	40.5±0.0

Table 7.6 Spearman Correlations for SummEval, averaged over all attributes (`COH`, `CON`, `FLU`, `REL`). K is the number of comparisons made, where $K = 240$ is the full set of comparisons.

Here, we investigate whether the Product of Experts framework can yield reasonable performance for SummEval when a subset of comparisons are used. Table 7.6 presents the results, where it's observed that when considering the full set of comparisons ($K = 240$), the correlations from the average probability is only marginally better than using the win ratio (within 1 SCC). However, when using 20% of the comparisons ($K = 48$), the average probability performs better than win ratio and yields gains of 3-4 SCC. This highlights that leveraging soft probabilistic information is advantageous when using only a subset of comparisons.

However, using the PoE solution yields significant gains in efficient settings. Even when only using hard decisions, for $K = 48$, both the Bradley-Terry model (BT) and the PoE Gaussian with hard decisions (PoE-g-hard) have mild performance gains over the win ratio. Nevertheless, the real benefits are seen when using PoEs with soft probabilities, with both POE-BT and PoE-g significantly outperforming the average probability. With these methods, when using only 20% of the comparisons, one can achieve performance close to when using the full comparison set (in four out of five cases within 2 SCC), where using the same comparisons, win ratio would result in degradations of up to 10 SCC. The findings hold across the different models and SummEval attributes. Further, both the Gaussian and the soft BT experts have similar performing solutions. With the full set of comparisons, the Gaussian PoE solution can be shown to be equivalent to the average probability, while the BT PoE may yield a different solution. Nonetheless, the performance for both PoE-BT and PoE-g are comparable across most models and datasets, as well as for both the hard and soft setups. The Gaussian solution, though, has the benefit of a convenient closed-form solution.

Finally, Figure 7.3 shows more detailed performance curves for a few selected models and attributes by sweeping K from $K = N$ to the full set of comparisons, $K = N \cdot (N-1)/2$. The curves show that the performance improves smoothly when the number of comparisons is increased, with the convergence rates considerably better with the PoE methods.



(a) FlanT5-3B, SummEval COH

(b) Llama2-13B, SummEval REL

(c) FlanT5-11B, TopicalChat, ENG

(d) Llama2-7B, TopicalChat, ENG

Fig. 7.3 Efficiency curves when sweeping K, the number of comparisons per context, where at each K the comparisons are randomly drawn 100 times. Average performance with 95% confidence is displayed.

HANNA and CMCQRD

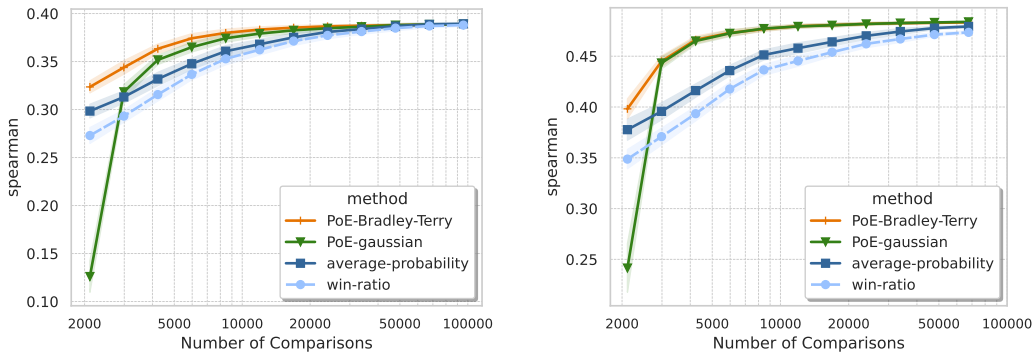
The previous experiments demonstrated that the PoE framework yields significant performance boosts on SummEval when a subset of the comparisons are used. However, SummEval has 16 generated summaries per context, and while considering all possible 240 comparisons may be computationally costly, it is feasible. However, some tasks have a much larger number of assessed texts, and for HANNA and CMCQRD, where there is no context dependence, the number of candidate texts is $N = 1056$ and $N = 658$ respectively. This would result in a total of over 200,000 comparisons, which would be very impractical to assess. To investigate whether the PoE framework yields practical advantages in such settings, Table 7.7 presents performance when using $\alpha \cdot N$ comparisons. Across all models and datasets, POE-BT consistently performs better than average probability. Additionally, PoE-BT converges to its final performance faster, with the average performance difference between 5 and 50 comparisons

0.8 SCC apart, while for average probability, the discrepancy is 2.5 SCC. Note that only Llama2 and Mistral were considered due to Flant5's maximum token length of 512.

		CMCQRD DIF		HANNA COH		HANNA CMP		HANNA SUR	
system	K	avg-prob	PoE-BT	avg-prob	PoE-BT	avg-prob	PoE-BT	avg-prob	PoE-BT
Llama2-7B	$5N$	31.9±1.2	33.4±0.9	39.2±0.9	41.3±0.4	45.7±1.0	47.9±0.4	32.8±1.1	34.1±0.4
	$10N$	33.8±1.1	34.4±0.8	40.3±0.9	41.4±0.3	46.9±0.8	48.2±0.3	33.6±0.8	34.3±0.3
	$20N$	34.8±0.7	35.0±0.6	41.1±0.5	41.6±0.2	47.6±0.4	48.3±0.2	34.1±0.5	34.5±0.2
	$50N$	35.3±0.4	35.3±0.2	41.4±0.2	41.6±0.1	48.0±0.1	48.3±0.1	34.4±0.2	34.5±0.0
Llama2-13N	$5N$	30.0±1.3	31.2±1.0	39.9±0.7	41.3±0.5	51.7±0.9	54.6±0.3	34.6±1.1	36.9±0.7
	$10N$	31.5±1.0	31.9±0.5	41.2±0.4	41.8±0.3	53.4±0.6	54.9±0.2	36.0±0.8	37.2±0.4
	$20N$	32.2±0.7	32.3±0.4	41.8±0.5	41.9±0.3	54.3±0.4	55.1±0.1	36.8±0.6	37.5±0.2
	$50N$	32.6±0.3	32.6±0.2	42.1±0.3	42.1±0.1	54.9±0.2	55.1±0.1	37.2±0.3	37.6±0.1
Mistral-7B	$5N$	38.9±1.2	40.7±0.5	36.6±1.1	38.3±0.6	47.3±0.8	49.9±0.4	24.2±1.1	25.5±0.9
	$10N$	40.7±1.1	41.1±0.2	37.9±0.7	38.6±0.4	49.0±0.6	50.6±0.3	25.3±0.7	26.0±0.5
	$20N$	41.1±0.6	41.2±0.2	38.7±0.5	38.8±0.2	50.1±0.5	50.9±0.2	25.9±0.6	26.2±0.3
	$50N$	41.2±0.3	41.2±0.1	38.9±0.2	38.9±0.1	50.7±0.2	51.0±0.1	26.0±0.3	26.1±0.1

Table 7.7 Spearman correlations for CMCQRD and HANNA for specific attributes. $K \in \{5N, 10N, 20N, 50N\}$ is the total number of symmetric comparisons made, e.g. $5N$ refers to each sample being in 5 comparisons.

Figure 7.4 illustrates the full efficiency curves for several models and attributes. PoE-BT typically performs best, and though PoE-g often performs similarly to PoE-BT, PoE-g can struggle in very low information regions. In all cases, the PoE methods appear to converge to their solution within $10 \cdot N$ comparisons, significantly fewer than the total number of $N \cdot (N-1)$ comparisons.

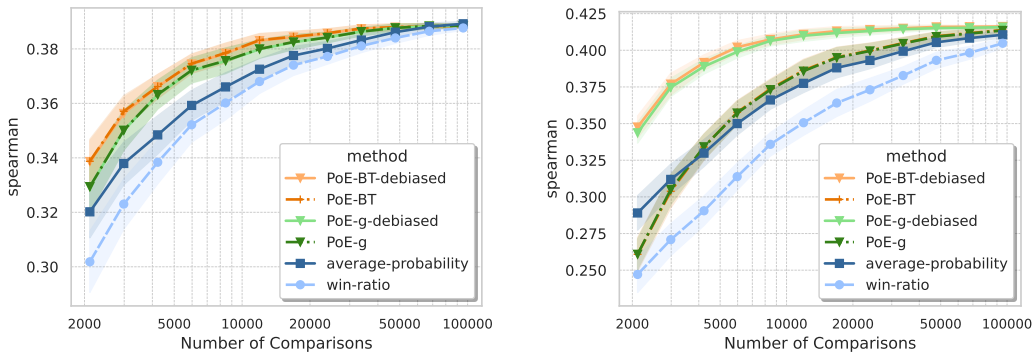


(a) Mistral-7B, HANNA COH

(b) Llama2-7B, HANNA CMP

Fig. 7.4 Efficiency curves when sweeping K, the number of comparisons per context, where at each K the comparisons are randomly drawn 20 times. Average performance with 95% confidence is displayed.

7.3.5 Non-Symmetric Comparions



(a) Mistral-7B, HANNA COH

(b) Llama-7B, HANNA COH

Fig. 7.5 Efficiency curves in the non-symmetric set up.

To minimise the influence of positional bias, in all previous experiments, both permutations of any comparison were evaluated and averaged. Although this reduces bias, one may gain more information by having a more diverse set of comparisons and by using a non-symmetric setup where we don't enforce both permutations to be sampled. In such cases, we can use debiased experts, which were proposed in Section 7.2.6. To investigate whether permutation debiased comparisons are required and whether our debiasing experts are effective, Figures 7.5a and 7.5b present performance curves in the non-symmetric setup for Mistral-7B and Llama-7B, respectively. Mistral-7B has minimal positional bias with $E_{i\neq j}[p_{ij}] = 0.51$, while Llama-7B has considerable bias with $E_{i\neq j}[p_{ij}] = 0.78$. For Llama2-7B, the debiased experts, $\mathrm{p}_\gamma(s_i - s_j|p_{ij})$, yield large performance gains and performance does not converge quickly without it. For Mistral-7B, the debiasing parameter has little influence, as expected, since γ will be near 0. Note that although Llama2-7B is more biased, it has better judgement capabilities and achieves better correlations than Mistral-7B once debiased. Lastly, Figure 7.6 compares non-symmetric debiased performance with symmetric performance and illustrates that the two perform similarly, albeit with slightly different characteristics. Non-symmetric often does better in the low number of comparisons region, symmetric sometimes marginally better after, and performance is similar when more comparisons are made.

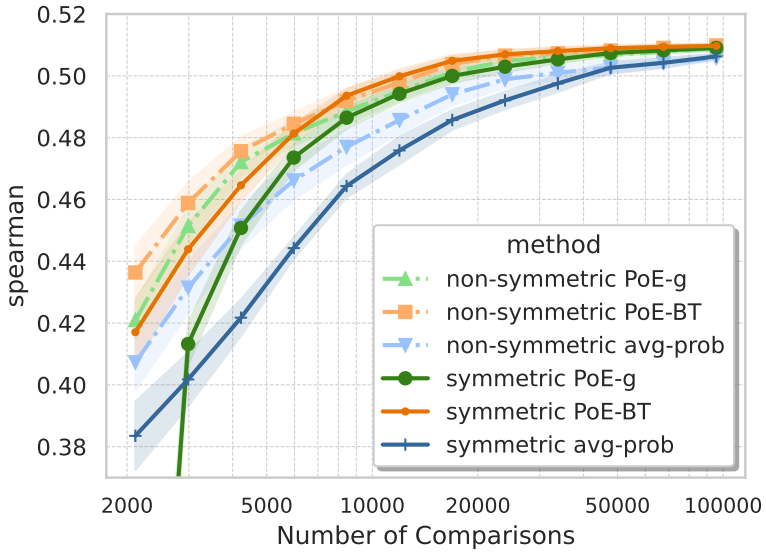


Fig. 7.6 Mistral-7B, HANNA COH, symmetric vs non-symmetric

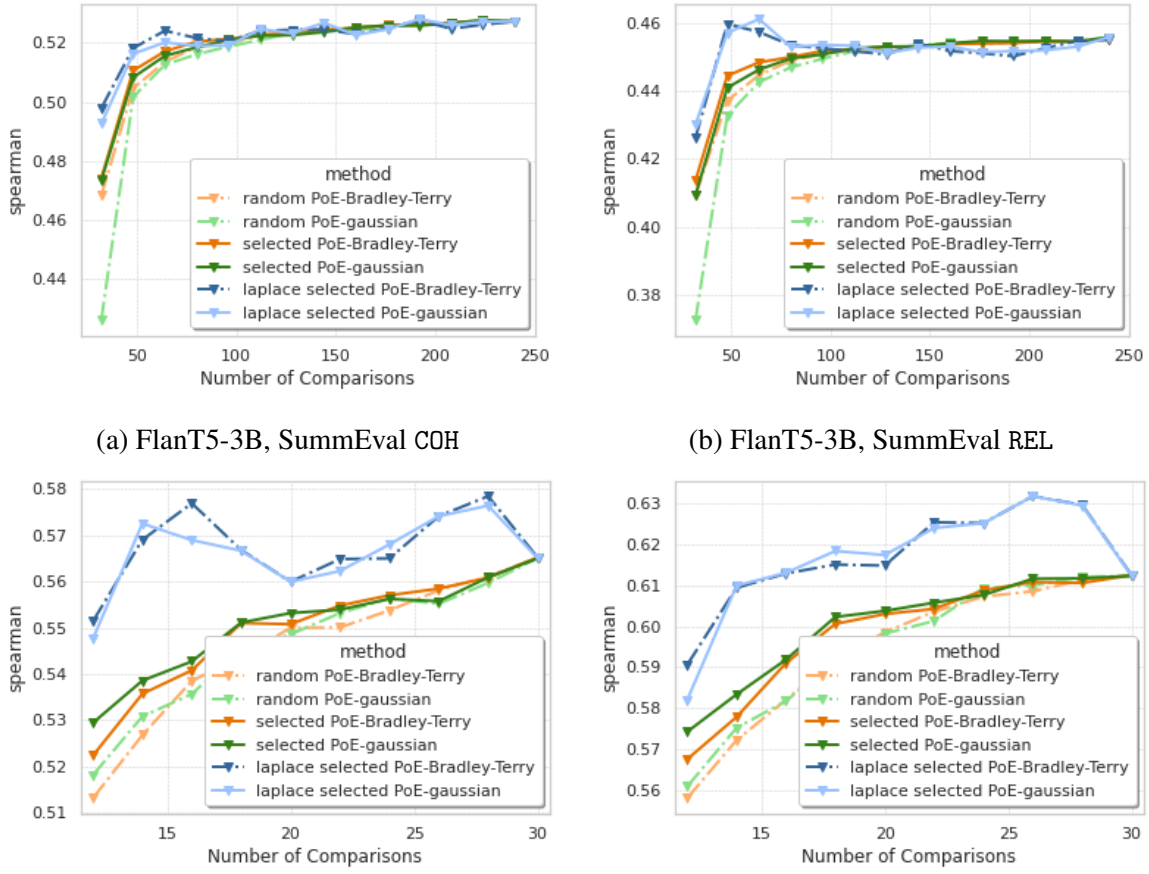
7.3.6 Selection Sensitivity

system	method	win-ratio	avg-prob	PoE-BT	PoE-g
Llama2-7B	random	21.6±0.8	24.0±0.7	26.8±0.5	26.6±0.5
	selected	23.0±0.8	24.5±0.6	27.3±0.4	27.2±0.4
Llama2-13B	random	30.8±0.7	33.7±0.6	37.7±0.4	37.3±0.4
	selected	32.4±0.7	34.6±0.6	38.2±0.3	38.0±0.4
Mistral-7B	random	29.7±0.8	31.1±0.7	33.2±0.6	32.8±0.6
	selected	31.4±0.7	32.2±0.6	34.0±0.5	33.9±0.5
Flant5-3B	random	34.1±0.8	38.4±0.6	42.7±0.4	42.4±0.4
	selected	36.0±0.6	39.3±0.6	43.2±0.4	42.9±0.3
Flant5-11B	random	31.2±0.8	34.7±0.7	38.4±0.4	38.4±0.4
	selected	33.1±0.6	35.7±0.7	39.2±0.4	39.0±0.4

Table 7.8 SummEval Spearman correlations when using the greedy optimal set of comparisons, for $K = 48$.

The previous results used random comparisons from all possible comparisons. An alternative would be to pre-select a set of comparisons that maximises the information gained from a fixed number of comparisons. Section 7.2.5 discusses how for the Gaussian-POE, this can be

achieved with a practical greedy approximation. Table 7.8 illustrates that at the operating point of $K = 48$, pre-selecting the comparisons can provide further performance boosts, with the average performance of the probabilistic PoE approaches consistently increasing by 0.5 SCC for all approaches, at no extra cost. Further, although the theory was derived using the Gaussian assumptions, the performance boosts are seen for all methods, with the largest gains for the win ratio. Additionally, Figure 7.7 shows that the performance gains are substantial with fewer comparisons, but as the number of comparisons increases, the performance difference between random and optimal selection becomes negligible. Finally, selecting the comparisons based on the Laplace-approximation of the Bradley-Terry expert (§7.2.5) yields better performance when a small number of comparisons are considered; however, it is more computationally expensive as the BT solution has to be determined at each timestep[5].



(a) FlanT5-3B, SummEval COH

(b) FlanT5-3B, SummEval REL

(c) FlanT5-11B, TopicalChat, COH

(d) FlanT5-11B, TopicalChat, CON

Fig. 7.7 FlanT5-3B, SummEval COH, selected vs random

[5]As a result, in Figure 7.7 the Laplace-selected curves are shown for only a single run of comparison selection, which provides a noisier estimate of the convergence properties.

7.4 Chapter Summary

This chapter introduced LLM comparative assessment, which prompts LLMs to make judgements between which of two responses is of greater quality. Experiments established that LLM comparative assessment performs better than many existing baselines and alternative zero-shot approaches, highlighting that LLM comparative assessment is an effective zero-shot, zero-resource text quality evaluation approach. It was further illustrated that LLM comparative judges exhibit significant positional bias, but by averaging the probabilities from both permutations, this bias can be mitigated and improve the reliability of the systems. Lastly, we proposed the Products of Experts framework, which enabled comparative assessment to be performed with significantly fewer comparisons.

Chapter 8

Bias in Zero-Shot Audio Classifiers

The preceding chapters have all focused on NLP foundation models, analysing their abilities, limitations and biases. Chapter 5 demonstrated the risk of spurious correlations, where models fine-tuned for specific applications learn to leverage non-generalisable features, while chapters 6 and 7 demonstrated that instruction-tuned LLMs also exhibit biases when prompted for downstream tasks, such as position bias and label word bias. Beyond NLP, foundation models have also had a significant impact in other domains and modalities, such as computer vision [55, 108, 262] and audio processing [287, 11, 128, 263]. This Chapter extends the analysis of zero-shot prompted foundation models to speech foundation models. Several speech foundation models exist, including those that are pre-trained using self-supervised learning (SSL) tasks [287, 11], as well as ASR foundation models [263, 255]. Section 8.1 first provides an overview of the systems, while 8.2 describes various approaches of applying such systems to audio classification, split into supervised fine-tuning and zero-shot approaches. Section 8.3 then introduces our novel method of prompting ASR systems for text classification. Section 8.4 discusses the risk of word bias for prompted audio models, extending the mitigation approaches introduced for NLP (§6.3) to the speech domain. Finally, Section 8.5 presents our experimental results, demonstrating that ASR foundation models can achieve zero-shot classification abilities but that debiasing approaches can yield significant performance improvements.

8.1 Audio Foundation Models

Chapter 3 discussed how crucial elements that led to the current effective NLP foundation models were breakthroughs in architecture (such as the introduction of the transformer §2.2), effective SSL pre-text tasks (§3.1.1) and the increasing scale of model size and training data. Within the audio processing domain, there has been growing interest in developing

foundational audio models that utilise large-scale datasets to learn robust audio representations to enable better abilities for downstream tasks [11, 128, 263]. The transformer architecture, though initially applied within NLP, has also proven to be highly effective for audio processing tasks. An increasing number of current SoTA audio foundation models are transformer-based [11, 255, 263], leveraging either the encoder-only transformer architecture (§2.2.1) or the encoder-decoder transformer architecture[1] (§2.2.3). This section examines influential transformer-based foundation models that have advanced the field of audio processing. We focus on three key models: Wav2Vec 2.0 [11], the Multilingual Multimodal Speech (MMS) [255] and Whisper [263].

8.1.1 Wav2Vec 2.0

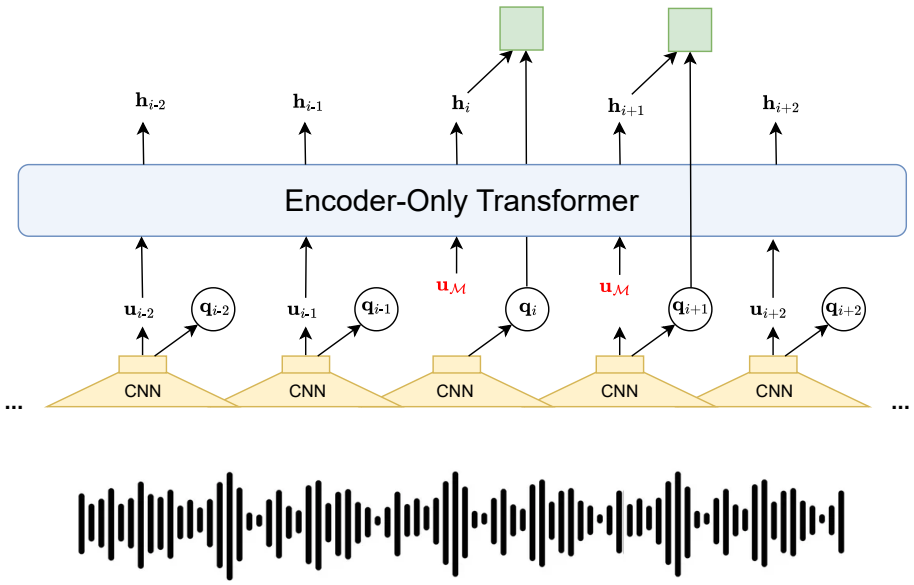


Fig. 8.1 Diagram of Wav2Vec 2.0 pre-training approach.

Wav2Vec [287] was one of the earliest works that directly applied SSL to speech processing. The first version of Wav2Vec used a convolutional neural network (CNN) architecture (§2.1.2) and was pre-trained to predict future audio representations from past contexts. Although Wav2Vec demonstrated the potential of learning useful speech representations from unlabeled data, it was Wav2Vec 2.0 [11] which outperformed existing state-of-the-art systems and marked a significant advance in applying SSL to audio processing.

[1] Due to the mismatch between input and output modalities in many audio tasks, decoder-only transformers are less prevalent

Wav2Vec 2.0 uses an encoder-only transformer architecture (§2.2.1) and combines masked prediction [51] and contrastive learning [35] to pre-train audio systems. The Wav2Vec 2.0 pre-training process is illustrated in 8.1, and has the following steps:

1. **Feature Encoder**: The input to Wav2Vec 2.0 is the raw audio signal, represented as $s_{1:T}$. In this section, where the focus is the audio modality, the input no longer represents discrete tokens and instead is a discrete-time representation of a continuous audio signal, typically sampled at 16 kHz. Each $s_i \in \mathbb{R}$ represents the amplitude of the waveform at time step i. The first stage of Wav2Vec 2.0 is the feature encoder, a multi-layer convolutional neural network (CNN) that processes the raw waveform and produces a sequence of latent speech representations, $\mathbf{u}_{1:N}$,

$$\mathbf{u}_{1:N} = \text{FeatureEncoder}(s_{1:T}) \tag{8.1}$$

   Here, N, the number of frames, is a direct function of T, the number of input timesteps, but is substantially smaller than T due to the temporal downsampling performed by the feature encoder. The feature encoder consists of several blocks, each containing temporal convolution, layer normalisation, and a GELU activation function. The convolutional layers process the audio signal with increasing receptive fields, resulting in feature vectors ($\mathbf{u}_{1:N}$), each corresponding to a frame representing approximately 25ms of audio.

2. **Quantisation**: Each latent vector $\mathbf{u}_i \in \mathbb{R}^n$ from the feature encoder is then quantised into $\tilde{\mathbf{q}}_i \in \mathbb{R}^n$ using product quantisation [138], achieved by selecting the closest sub-representations from multiple codebooks for each vector $\mathbf{u}_i$. Assume there are G codebooks, $\mathcal{B}_g$, where each codebook contains V entries. Each entry $\mathbf{e} \in \mathbb{R}^{n/G}$ is a n/G dimensional vector. The latent vector $\mathbf{u}_i$ is split into G sub-vectors, $\mathbf{u}_{i,g}$, where $\mathbf{u}_{i,g} \in \mathbb{R}^{n/G}$ represents the portion of $\mathbf{u}_i$ from the $(g-1)\frac{n}{G}$-th element to the $(g)\frac{n}{G}$-th element. For the g-th sub-vector $\mathbf{u}_{i,g}$, the closest entry $\mathbf{e}_{i,g}$ is selected from the g-th codebook $\mathcal{B}_g$,

$$\mathbf{e}_{i,g} = \arg\min_{\mathbf{e} \in \mathcal{B}_g} \|\mathbf{u}_{i,g} - \mathbf{e}\| \tag{8.2}$$

   The selected vectors $\mathbf{e}_{i,1}, \mathbf{e}_{i,2}, \ldots, \mathbf{e}_{i,G}$ are then concatenated and passed through a linear transformation, to produce the final quantised representation, $\mathbf{q}_i \in \mathbb{R}^d$:

$$\tilde{\mathbf{q}}_i = [\mathbf{e}_{i,1} \; ; \; \mathbf{e}_{i,2} \; ; \; \ldots \; ; \; \mathbf{e}_{i,G}] \tag{8.3}$$

$$\mathbf{q}_i = \mathbf{W}\tilde{\mathbf{q}}_i \tag{8.4}$$

where $\mathbf{W} \in \mathbb{R}^{d \times n}$ is a learnable linear transformation matrix. Note that the Gumbel softmax [137] is used to enable the gradients to be propagated through the quantisation.

3. **Masked Representations from Transformers**: Wav2Vec 2.0 is based on the encoder-only transformer architecture (§3.2.1) and similar to BERT [51], the latent vectors are masked. This is done by randomly selecting 6.5% of the feature encoder vectors and masking the subsequent ten timesteps (approximately 49% of all representations are masked). Masking is performed by replacing all masked vectors with a learned feature vector $\mathbf{u}_{\mathcal{M}} \in \mathbb{R}^d$:

$$\tilde{\mathbf{u}}_i = \begin{cases} \mathbf{u}_{\mathcal{M}}, & \text{if } i \in \mathcal{M} \\ \mathbf{u}_i, & \text{otherwise} \end{cases} \tag{8.5}$$

Where $\mathcal{M}$ is the set of masked indices and $\tilde{\mathbf{u}}_{1:N}$ denotes the masked sequence representations. The representations are then passed through a GELU, layer normalisation, and subsequently fed into the layers of the encoder-only transformer. This results in the final contextual audio representations, $\mathbf{h}_{1:N}$, taken from the final layer of the transformer,

$$\mathbf{h}_{1:N} = \text{TransformerEncoder}(\tilde{\mathbf{u}}_{1:N}; \boldsymbol{\theta}) \tag{8.6}$$

where $\boldsymbol{\theta}$ are the parameters of the transformer.

4. **Contrastive Loss**: Wav2Vec 2.0 is then trained in a self-supervised way to distinguish the true quantised latent speech representation from distractors at each time step. The model uses a contrastive loss [104], where given the contextual audio representation $\mathbf{h}_i$ centred on a masked time step i, the model has to identify the correct quantised latent speech representation $\mathbf{q}_i$ from a set of $\kappa+1$ candidate representations within the set $\mathcal{Q}_i$. This set $\mathcal{Q}_i$ includes the true representation $\mathbf{q}_i$, as well as κ distractors which are uniformly sampled from other masked time steps within the same utterance. The contrastive loss $\mathcal{L}_m(\boldsymbol{\theta})$ is then defined as:

$$\mathcal{L}_m(\boldsymbol{\theta}) = -1 \cdot \frac{1}{|\mathcal{M}|} \cdot \sum_{i \in \mathcal{M}} \log \frac{\exp(\text{sim}(\mathbf{h}_i, \mathbf{q}_i)/\tau)}{\sum_{\tilde{\mathbf{q}} \in \mathcal{Q}_i} \exp(\text{sim}(\mathbf{h}_i, \tilde{\mathbf{q}})/\tau)} \tag{8.7}$$

where τ is a temperature hyperparameter that controls the sharpness of the softmax distribution, and the similarity function $\text{sim}(\mathbf{a}, \mathbf{b})$ is the cosine similarity between two

vectors **a** and **b**,

$$\text{sim}(\mathbf{a}, \mathbf{b}) = \frac{\mathbf{a}^\top \mathbf{b}}{\|\mathbf{a}\|\|\mathbf{b}\|} \tag{8.8}$$

There is also an additional diversity loss that is introduced to to maximise the entropy of the averaged softmax distribution over the codebook, encouraging a more uniform use of the codebook entries.

Wav2Vec 2.0 is trained on 960 hours of unlabelled audio from the Librispeech corpus [236] and 53.2k hours of audio from LibriVox (LV-60k) [145]. Wav2Vec 2.0 is released in two sizes: `base`, which has 95 million parameters, and `large`, which has 317 million parameters. Wav2Vec can then be fine-tuned for speech recognition by adding a linear projection on top of the context network and trained with the Connectionist Temporal Classification (CTC) loss [96]. This process will be discussed in greater detail in the next section.

8.1.2 MMS

Wav2Vec 2.0 demonstrated significant advancements in self-supervised learning for audio, providing robust representations of speech. However, without further training, Wav2Vec 2.0 cannot be directly used for audio processing tasks such as automatic speech recognition (ASR). Like BERT in NLP, Wav2Vec 2.0 serves as a foundation model, adaptable to various audio processing tasks through fine-tuning. One notable example is the Massively Multilingual Speech (MMS) system [255], an Automatic Speech Recognition (ASR) foundation model based on Wav2Vec 2.0. MMS significantly expands the original architecture's capabilities to handle multilingual speech recognition tasks efficiently, a process achieved with the following steps:

1. **Character Level Linear Projection Layer:** MMS extends the model architecture by adding a linear projection layer, often referred to as a "head". Assuming K possible output characters, this layer consists of learnable parameters $\mathbf{W} \in \mathbb{R}^{K \times d}$ and $\mathbf{b} \in \mathbb{R}^K$, which are applied to the outputs of the contextualised audio representations, $\mathbf{h}_i \in \mathbb{R}^d$. This linear projection is used to predict the state of the input at frame i, outputting probabilities for each associated character,

$$\mathtt{P}(y_i = \omega_k | \mathbf{h}_{1:N}; \boldsymbol{\theta}) = \frac{\exp(\mathbf{w}_k^\top \mathbf{h}_i + b_k)}{\sum_{j=1}^K \exp(\mathbf{w}_j^\top \mathbf{h}_i + b_j)} \tag{8.9}$$

where y_i represents the character for the i-th frame such that $y_i \in \{\omega_1, \omega_2, ..., \omega_K\}$, and ω_k represent particular characters (e.g. a, b, c, ...). These frame-level predictions are

then combined using a decoding algorithm to produce the final output text transcript (discussed in 3).

2. **Labelled ASR data collection**: After adding the character-level projection layer, MMS can be fine-tuned on labelled audio data. This process begins with collecting relevant transcribed audio data. MMS, a single system designed for multi-language ASR, covers 1406 languages and is trained on 36,800 hours of paired speech data, primarily from recordings of the New Testament in various languages. Many publicly available ASR datasets $\mathcal{D}$ consist of audio recordings paired with text transcriptions:

$$\mathcal{D} = \{s_{1:T}^{(j)}, x_{1:L}^{(j)}\}_{j=1}^{M} \tag{8.10}$$

Where for convenience, T and L are used generally to represent the length of the input audio and reference text respectively, though, in practice, they vary from sample to sample. The absence of time-stamp information poses challenges in training, as the model must learn both the audio-to-text mapping and their alignment. Furthermore, the input sequence length T and corresponding number of frames N are typically much larger than the output sequence length L, further complicating the alignment process. This challenge can be addressed using Connectionist Temporal Classification (CTC) loss.

3. **CTC training**: The Connectionist Temporal Classification (CTC) [96] loss addresses the alignment issue by considering all possible alignments between the input sequence and the output transcription. CTC introduces a special blank symbol, ε, which allows the model to handle repetitions, silences, or uncertain frames. The CTC loss maximises the probability of the correct transcription by marginalizing over all possible alignments, enabling the model to learn transcription and alignment simultaneously:

$$L_{CTC}(\boldsymbol{\theta}) = -1 \cdot \frac{1}{M} \cdot \sum_{j=1}^{M} \log \left(\sum_{\boldsymbol{\pi} \in \mathcal{S}_N(x_{1:L}^{(j)})} \prod_{i=1}^{N} P(\pi_i | \mathbf{h}_{1:N}^{(j)}; \boldsymbol{\theta}) \right) \tag{8.11}$$

where $\mathcal{S}_N$ is the one-to-many mapping of transcriptions to possible alignments of length N, $\boldsymbol{\pi}$ is a selected alignment of $x_{1:L}^{(j)}$ (e.g. $\boldsymbol{\pi} = \{t, h, \varepsilon, \varepsilon, e, ...\}$) and π_i is the symbol at the i-th frame. Despite the exponential number of possible alignments, CTC efficiently computes this loss using the forward-backward algorithm [261], achieving $O(NL)$ time complexity by leveraging dynamic programming techniques.

8.1.3 Whisper

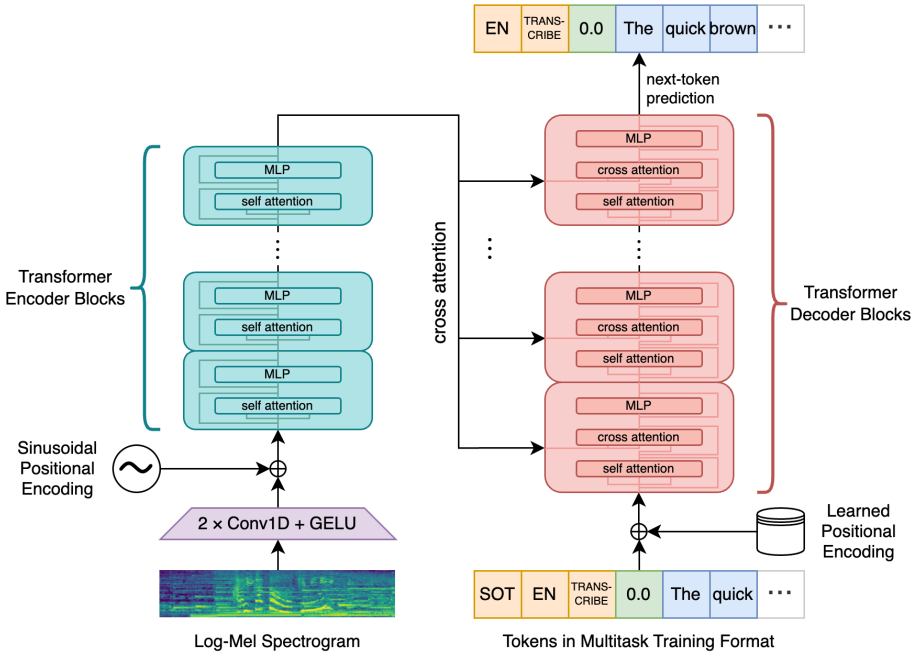


Fig. 8.2 Diagram of Whisper Architecture, taken from the original paper [263].

Another popular ASR foundation model is Whisper [263], which is based on the encoder-decoder transformer (§2.2.3). Unlike Wav2vec 2.0 and the NLP foundation models discussed in this thesis (§3.2), Whisper doesn't use SSL training but rather is trained on vast quantities of weakly supervised multilingual data obtained from the web. Whisper has demonstrated robust generalisation across various languages and accents, and can handle diverse audio processing tasks beyond ASR. Whisper's training process can be summarised as follows:

1. **Input tokenization** Whisper converts raw input audio $s_{1:T}$ into frame-level features $u_{1:N}$. It segments the audio into overlapping 25ms windows, converts each into log-Mel spectrograms [302], and processes these through a two-layer CNN (§2.1.2) to extract salient features, $\mathbf{u}_{1:N}$. These frame features are then fed into the transformer encoder, which uses sinusoidal positional encoding (following the original transformer architecture).

2. **Data Collection** Whisper is trained on a large-scale, diverse dataset $\mathcal{D}$ of 680,000 hours of labelled speech data, primarily sourced from online videos with associated transcripts. Like in MMS, Whisper's dataset lacks precise time-stamp information. This multilingual dataset covers various tasks, including ASR, speech translation, and

language identification, and is also diverse, covering a wide range of speakers, accents and acoustic conditions. The data is "weakly supervised," and the transcriptions are either human-written or generated by other automatic systems. However, the exact details of the training data are not publicly available.

3. **Supervised End-to-End Training** Unlike MMS's encoder-only transformer, which makes frame-level character predictions, Whisper uses an encoder-decoder architecture. This allows the decoder to attend over a longer context of input frames, enabling token-level predictions and resolving alignment issues. Whisper directly models the next output token given the input audio and previously generated tokens, $\mathtt{P}(x_{i+1}|x_{1:i}, s_{1:T}; \boldsymbol{\theta})$. The model is trained with Negative Log-Likelihood (NLL) loss:

$$\mathcal{L}(\boldsymbol{\theta}) = -1 \cdot \frac{1}{M} \cdot \log \left(\prod_{j=1}^{M} \mathtt{P} \left(x_{1:L}^{(j)} | s_{1:T}^{(j)}; \boldsymbol{\theta} \right) \right) \tag{8.12}$$

$$= -1 \cdot \frac{1}{M} \cdot \sum_{j=1}^{M} \sum_{i=1}^{L} \log \left(\mathtt{P} \left(x_{i}^{(j)} | s_{1:T}^{(j)}, x_{1:i-1}^{(j)}; \boldsymbol{\theta} \right) \right) \tag{8.13}$$

This approach allows the model to learn alignment and transcription simultaneously, with the alignment implicitly learned in the cross-attention mechanism. This design enables a single model to handle various tasks and languages efficiently. To differentiate between tasks, Whisper prepends a task-specific instruction to the decoder input (as illustrated in Figure 8.2).

Whisper is available in various model sizes, ranging from 39M parameters (Whisper tiny) to 1.55B parameters (Whisper large). These models come in two variants: English-only and multilingual, except the largest model (1.55B), which is only available in the multilingual version. All Whisper models are trained for automatic speech recognition (ASR) and voice activity detection, with the multilingual models are additionally trained for speech translation and language identification.

8.2 Audio Classification

The previous section discussed several audio-foundation models, detailing their architectures and pre-training methodologies. This section explores how these audio foundation models can be adapted for audio classification tasks. The objective of audio classification is to categorise input audio into predefined classes. This encompasses a diverse range of tasks, including sound event classification [249, 282], speaker emotion detection [198, 31], and

music genre classification [304]. Typical audio classification methods train deep learning systems in a supervised fashion, with state-of-the-art performance recently achieved by fine-tuning pre-trained audio foundation models on task-specific data. Additionally, zero-shot learning approaches have also been explored, where the representations of existing systems are adapted to new classification tasks without additional labelled data, though often with reduced performance. The following section will discuss audio classification methods, detailing both the supervised training and zero-shot representation matching methods.

8.2.1 Supervised Training

Similar to supervised training within NLP (§4.1.2), supervised approaches in audio classification optimise model weights to maximise the likelihood of the training data. Let $\mathcal{D}$ be a labelled audio classification dataset for a particular task:

$$\mathcal{D} = \{s_{1:T}^{(i)}, y^{(i)}\}_{i=1}^{M} \tag{8.14}$$

where $s_{1:T}^{(i)}$ represents the input audio and $y^{(i)} \in \{\omega_1, ..., \omega_K\}$ represents the true label of the audio for the particular task. Let Encode$(\cdot)$ denote a deep learning system that takes input audio $s_{1:T}$ and processes it to an audio vector representation $\mathbf{h}_{\texttt{cls}} \in \mathbb{R}^d$. This vector[2] can be passed to a classification head, which produces the predicted output class probabilities,

$$\mathbf{h}_{\texttt{cls}} = \text{Encode}(s_{1:T}) \tag{8.15}$$

$$\texttt{P}(\omega_k | s_{1:T}; \boldsymbol{\theta}) = \frac{\exp(\mathbf{w}_k^\top \mathbf{h}_{\texttt{cls}} + b_k)}{\sum_{j=1}^{K} \exp(\mathbf{w}_j^\top \mathbf{h}_{\texttt{cls}} + b_j)} \tag{8.16}$$

where $\mathbf{W} \in \mathbb{R}^{K \times d}$ and $\mathbf{b} \in \mathbb{R}^K$ are parameters of the classification head. Once the audio representations $\mathbf{h}$ are obtained, the training objective is typically the NLL loss (which is equivalent to cross-entropy loss):

$$\mathcal{L}(\boldsymbol{\theta}) = -\frac{1}{M} \sum_{i=1}^{M} \log \texttt{P}(y^{(i)} | s_{1:T}^{(i)}; \boldsymbol{\theta}) \tag{8.17}$$

Diverse architectures have been explored for encoding the audio, with the weights either initialised from scratch [248, 115] or using pre-trained weights [153, 92]. An example of leveraging pre-trained weights is the approach of fine-tuning CNN-14 [153] for audio classification. CNN-14 is a 14-layer CNN architecture in the PANNs family (Pre-trained Audio Neural Networks) [153], systems which were trained on the large AudioSet dataset [87]

[2]The `CLS` here stands for classification, and the vector it is not associated with any CLS token.

for the task of audio tagging. Audio tagging is a multi-label classification problem where each audio input can be associated with multiple tags simultaneously. Hence, during pre-training, the system uses the sigmoid activation instead of the softmax and employs the binary cross-entropy loss. CNN-14 can then be fine-tuned for specific audio classification tasks and has been shown to outperform models trained from scratch.

Another promising approach is to leverage the representations from large-scale ASR foundation models such as Whisper [263]. These models, trained on vast amounts of diverse audio data, have been shown to capture rich and generalisable audio features that can benefit downstream audio classification tasks. For instance, Gong et al. [92] propose taking the audio representations from various layers of Whisper and using pooling methods to obtain compact audio representations. One pooling approach is final-layer mean pooling, where the audio representation is obtained by averaging the encoder's final layer output representations across all time frames:

$$\mathbf{h}_{\texttt{cls}} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{h}_i \tag{8.18}$$

where $\mathbf{h}_i$ represents the final-layer hidden state for frame i, and N is the number of frames in the audio sequence.

8.2.2 Representation Matching

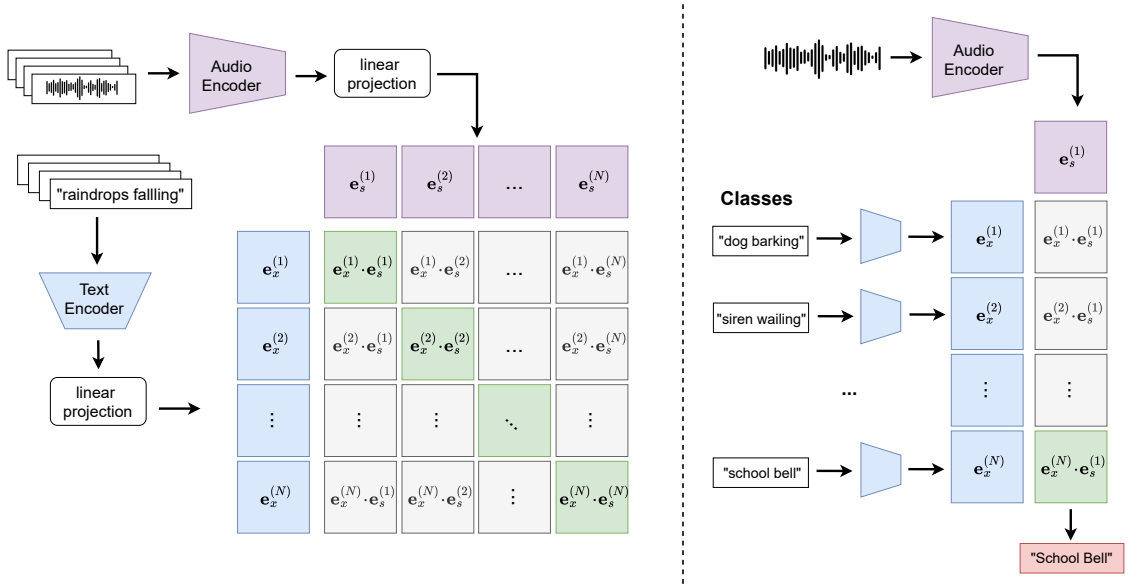


Fig. 8.3 Illustration of CLAP. **left**: illustration the contrastive learning to align the audio and text representations **right**: using CLAP for zero-shot audio classification.

Foundation models are capable of capturing meaningful representations, both within text and audio domains. Leveraging this capability, methods such as CLAP [62] and AudioCLIP [103] achieve zero-shot audio classification by encoding both the audio input and the textual descriptions and then matching the audio representation to the closest text representation, enabling classification without prior training on specific categories. For example, CLAP [62] uses CNN14 [153] as an audio encoder, which converts the input audio $s_{1:T}$ into the audio representation $\mathbf{h}_{\mathbf{s}} \in \mathbb{R}^{d_s}$,

$$\mathbf{h}_{\mathbf{s}} = \text{Encode}_s(s_{1:T}; \boldsymbol{\theta}_s) \tag{8.19}$$

CLAP also uses BERT [51] as a text encoder to encode texts $x_{1:L}$ to the text representation, $\mathbf{h}_{\mathbf{x}} \in \mathbb{R}^{d_x}$:

$$\mathbf{h}_{\mathbf{x}} = \text{Encode}_x(x_{1:L}; \boldsymbol{\theta}_x) \tag{8.20}$$

The idea of representation matching is that the vector representation of the audio $\mathbf{h}_{\mathbf{s}}$ should be close to the text representation of its description. However, measuring the similarity of the vectors is difficult as the text and audio representations are in completely different and unrelated spaces. Therefore, the first stage is to adapt the representations to exist in a joint multimodal space through learnable linear projections $\mathbf{W}_s \in \mathbb{R}^{d \times d_s}$ and $\mathbf{W}_x \in \mathbb{R}^{d \times d_x}$:

$$\mathbf{e}_{\mathbf{s}} = \mathbf{W}_s \mathbf{h}_{\mathbf{s}} \qquad \mathbf{e}_{\mathbf{x}} = \mathbf{W}_x \mathbf{h}_{\mathbf{x}} \tag{8.21}$$

Where $\mathbf{e}_{\mathbf{s}}$ is the projected audio representation and $\mathbf{e}_{\mathbf{x}}$ is the projected text representation. Given a batch $\mathcal{B}$ of audios and matched text classes, $\mathcal{B} = \{(s^{(i)}_{1:T}, x^{(i)}_{1:L})\}_{i=1}^{M}$, the parameters of the models and projection layers, $\boldsymbol{\theta} = [\boldsymbol{\theta}_x; \boldsymbol{\theta}_s; \mathbf{W}_x; \mathbf{W}_s]$, can be learned using a contrastive learning loss where the distance between paired representations is minimised while the distance between all other pairs is maximised:

$$\mathcal{L}(\boldsymbol{\theta}) = -\frac{1}{2M} \sum_{i=1}^{M} \left[\log \left(\frac{\exp\left(\mathbf{e}_{\text{s}}^{(i)} \cdot \mathbf{e}_{\text{x}}^{(i)}\right)}{\sum_{j=1}^{L} \exp\left(\mathbf{e}_{\text{s}}^{(i)} \cdot \mathbf{e}_{\text{x}}^{(j)}\right)} \right) + \log \left(\frac{\exp\left(\mathbf{e}_{\text{x}}^{(i)} \cdot \mathbf{e}_{\text{s}}^{(i)}\right)}{\sum_{j=1}^{L} \exp\left(\mathbf{e}_{\text{x}}^{(i)} \cdot \mathbf{e}_{\text{s}}^{(j)}\right)} \right) \right] \tag{8.22}$$

CLAP is trained on 128,010 audio-text pairs from four distinct datasets. Another zero-shot approach, AudioCLIP [103], extends CLIP to align audio embeddings with other modalities rather than using BERT and CNN-14. This is achieved by introducing an audio processing head and performing similar contrastive learning on AudioSet [87]. Both approaches enable zero-shot solutions for new tasks without requiring retraining. This is accomplished by

selecting label word sequences that reflect the desired classes and then identifying the class with the highest similarity to the audio embedding, as illustrated in Figure 8.3.

8.3 Zero-Shot Classification with ASR Foundation Models

The previous section discussed audio classification methods, covering both supervised training and representation-matching approaches. While representation-matching methods such as CLAP [62] and AudioClip [103] can seemingly adapt to new, unseen tasks, these models were trained on extensive large-scale audio tagging and/or audio description datasets and then evaluated on similar audio-tagging and audio classification tasks. One may therefore question whether the approach can generalise to tasks less similar to the pre-training tasks, and whether the approach is zero-shot.

In light of these limitations, we propose an alternative zero-shot audio classification approach. Our method draws inspiration from NLP, where foundation models trained on vast text data demonstrated emergent capabilities [30]. These systems, which were not instruction-tuned, could be adapted to tasks dissimilar to the training task and still demonstrate abilities in unseen tasks [80, 286, 265]. Current ASR foundation models are trained on vast quantities of data and generalise well to new domains, mirroring earlier NLP foundation models. This section, therefore, proposes a method to leverage existing ASR foundation models for unseen audio classification tasks without training or updating any parameters, achieved through a novel form of prompting.

8.3.1 Zero-shot ASR prompt-classifiers

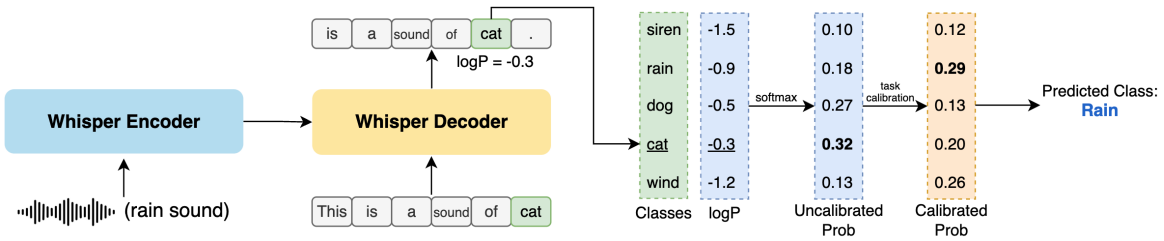


Fig. 8.4 ASR foundation models are leveraged for zero-shot audio classification by prompting the decoder to calculate the log-likelihood of label sequences associated with each class. The log-likelihood for each class is converted to probabilities and post-processed to a predicted class. This process is displayed for Whisper.

Recall that for NLP prompt-based classifiers (§4.1.3), the LLM output probability of a given label word is used as the classification probability. The input text $\boldsymbol{x} = x_{1:L}$ is formatted in a prompt template $\mathcal{P}$, and the probability of generating the respective label word z_k is assumed to be proportional to the probability associated with class ω_k,

$$\mathtt{P}(\omega_k|\boldsymbol{x},\mathcal{P},z_{1:K};\boldsymbol{\theta}) = \frac{\mathtt{P}_{lm}(z_k|\mathcal{P}(\boldsymbol{x});\boldsymbol{\theta})}{\sum_{j=1}^{K}\mathtt{P}_{lm}(z_j|\mathcal{P}(\boldsymbol{x});\boldsymbol{\theta})} \tag{8.23}$$

ASR systems differ from language models and do not use textual inputs. Instead, given an input audio $s_{1:T}$, the ASR system autoregressively calculates the probabilities associated with a particular output text sequence $x_{1:L}$, $\mathtt{P}(x_{1:L}|s_{1:T};\boldsymbol{\theta})$. Hence, for zero-shot *audio* classification, instead of templating the input, we alternatively propose to template the outputs. Given K classification classes, $\mathcal{Y} = \{\omega_1,\ldots,\omega_K\}$, and respective text descriptions of each class, $\{\boldsymbol{z}_1,\ldots,\boldsymbol{z}_K\}$, the probability an audio $s_{1:T}$ is associated to class ω_k can be proportional to the ASR probability of generating the textual description of the class:

$$\mathtt{P}(\omega_k|s_{1:T},\boldsymbol{z}_{1:K};\boldsymbol{\theta}) = \frac{\mathtt{P}(\boldsymbol{z}_k|s_{1:T};\boldsymbol{\theta})}{\sum_{j=1}^{K}\mathtt{P}(\boldsymbol{z}_j|s_{1:T};\boldsymbol{\theta})} \tag{8.24}$$

The label sequences $\boldsymbol{z}_{1:K}$ are generated using a common template $\mathcal{P}(\omega_k)$ for all classes. For example, the template could be ‘‘`This is a sound of _`’’, where the blank is filled with the class name of ω_k. We refer to this template as the ’prompt’, which is applied uniformly across all classes to create the label sequences. The zero-shot audio classification prompting process is depicted in Figure 8.4.

This approach may rely on leveraging the implicit acoustic-semantic associations learned from the ASR training domain. For instance, events such as a dog barking may have corresponding captions describing the action or other individuals present in the scene commenting on the sound. Given that the model must learn sound-class associations from labels derived from the supervised ASR transcripts, this may also raise questions on whether the approach can be considered truly zero-shot. However, given the extensive, diverse, and heterogeneous ASR training data used, these learned associations may generalise across a broad spectrum of tasks, and this approach is arguably more “zero-shot” than existing “zero-shot” approaches which use supervised training on a highly related task.

8.4 Debiasing Predictions

Section 6.5.1 demonstrated that large language models (LLMs) are prone to common-word bias, where frequently occurring words in training data overly influence the LM probabilities,

causing the prompt-based classifier to have an implicit class bias. ASR systems may face similar challenges, where uncommon terms have lower associated probabilities and, therefore, cause an implicit bias when used in our proposed zero-shot prompting approach. This section provides a brief discussion of bias in audio-processing tasks and then explores how the reweighting approaches from Section 6.3 can be applied to zero-shot audio classifiers.

8.4.1 Bias and Class Imbalance in Audio-Processing tasks

Bias is prevalent in many audio processing tasks. One example is within ASR; domain-specific terms may often appear infrequently in the training data, leading to lower probabilities during evaluation and omission from the predicted transcript. Contextual ASR [257, 353] addresses this issue typically by using bias lists, lists of context-specific terms that may be common in the evaluation domain but rare or missing in the training data. Incorporating these terms can be done by deep biasing, where the training is modified to use the context lists as a secondary input [73], e.g. converting the bias list to a context vector which guides the decoder's attention [257]. Alternatively, shallow fusion adjusts the final probability distribution by adding scores to words from the external bias list model, typically applied without retraining the model [353, 73]. Both methods aim to improve ASR performance on domain-specific or rare terms, thereby reducing bias that arises from domain mismatch between training and evaluation data.

Beyond ASR, several audio classification datasets can suffer from label imbalance, which may cause class bias and poor performance in the underrepresented classes. For example, the aim of speech emotion recognition (SER) is to identify the emotions expressed from speech patterns [289], e.g. happiness, anger, sadness, or frustration. However, for natural speech data, neutral samples are much more common than emotionally expressive ones, which results in the neutral emotion being overrepresented in many SER datasets. To overcome the resulting class imbalance, data augmentation techniques often synthetically augment the training data to increase the diversity of emotions present in the training data [33]. Alternatively, alternate evaluation metrics, such as average recall or the F1-score, can be used to give more balanced importance to underrepresented categories, providing a fairer assessment of model performance than traditional accuracy metrics [135].

8.4.2 Debiasing via Reweighting

This section considers how we may be able to adjust for and mitigate any systematic class bias that may arise in the zero-shot audio classifier. The debiasing approach will largely follow the reweighting debiasing approach used for the NLP prompt-based classifiers (§6.3).

In the reweighting debiasing approach, a set of weights $\boldsymbol{\alpha} = [\alpha_{1:K}]$ can be introduced to rescale the class probabilities,

$$\mathtt{P}(\omega_k|s_{1:T}, \boldsymbol{z}_{1:K}, \boldsymbol{\alpha}; \boldsymbol{\theta}) = \frac{\alpha_k \mathtt{P}(\omega_k|s_{1:T}, \boldsymbol{z}_{1:K}; \boldsymbol{\theta})}{\sum_{j=1}^{K} \alpha_j \mathtt{P}(\omega_j|s_{1:T}, \boldsymbol{z}_{1:K}; \boldsymbol{\theta})} \tag{8.25}$$

The weights $\boldsymbol{\alpha}$ can be selected to correct for any systematic class bias of the system. As discussed in Section 6.3, the weights can be optimised depending on the data availability. Given a dataset $\mathcal{D} = \{(s_{1:T}^{(i)}, y^{(i)})\}_{i=1}^{M}$, one can find the weights that maximise the accuracy,

$$\boldsymbol{\alpha}^* = \arg\max_{\boldsymbol{\alpha}} \left(\frac{1}{M} \cdot \sum_{i=1}^{M} \mathbb{1}\left(y^{(i)} = \arg\max_{y \in \mathcal{Y}} \mathtt{P}(y|s_{1:T}{}^{(i)}, \boldsymbol{z}_{1:K}, \boldsymbol{\alpha}; \boldsymbol{\theta})\right) \right) \tag{8.26}$$

However, this approach requires labelled data. Alternatively, the weights can be selected to equalise any implicit class priors that may manifest due to the selection of label sequences $\boldsymbol{z}_{1:K}$. The unsupervised 'prior-matching' approach (6.3.2) aims to ensure that the implicit class prior is uniform across all classes,

$$\bar{\boldsymbol{\alpha}} = \arg\min_{\boldsymbol{\alpha}} \sum_{k=1}^{K} \left| \mathtt{P}(\omega_k|\boldsymbol{z}_{1:K}, \boldsymbol{\alpha}; \boldsymbol{\theta}) - \frac{1}{K} \right| \tag{8.27}$$

where here $\mathtt{P}(\omega_k|z_{1:K}, \boldsymbol{\alpha}; \boldsymbol{\theta})$ is marginalised over the audios, approximated using available unsupervised data,

$$\mathtt{P}(\omega_k|\mathcal{P}, z_{1:K}, \boldsymbol{\alpha}; \boldsymbol{\theta}) \approx \frac{1}{M} \sum_{i=1}^{M} \mathtt{P}(\omega_k|s_{1:T}, \boldsymbol{z}_{1:K}, \boldsymbol{\alpha}; \boldsymbol{\theta}) \tag{8.28}$$

The final approach was the data-free approximation, which used no data. For NLP, we demonstrated that the prior matching approach could be approximated using a null input (§6.3.3). In text classification tasks, the null input was an informative text with no information, e.g. an empty string or the input 'N/A' [355]. For speech-based systems, these text-based null-inputs are not applicable. Therefore, as the null-input[3] $\emptyset_{1:T}$, we propose using acoustic features generated from synthetic Gaussian white noise with $\sigma = 1$. Using the same approximations as in Section 6.3.3, $\bar{\boldsymbol{\alpha}}$ can be estimated following:

$$\bar{\alpha}_k \approx \frac{1}{\mathbb{E}_{s_{1:T}}\left[\mathtt{P}(\boldsymbol{z}_k|s_{1:T}; \boldsymbol{\theta})\right]} \approx \frac{1}{\mathtt{P}(\boldsymbol{z}_k|\emptyset_{1:T}; \boldsymbol{\theta})} \tag{8.29}$$

[3] the length of the null input sequence, T, can be set to the average length of the audios for the task

That is, the weights $\bar{\alpha}_k$ are proportional to the probability of the ASR system generating each respective label sequence $\boldsymbol{z}_k$ when given the null input.

8.5 Experiments

The experiments in this section investigate whether the ASR foundation models discussed in Section 8.1 can be effectively prompted for zero-shot audio classification. We also examine whether the approach is affected by label word bias, a phenomenon which was observed when prompting NLP foundation models. Additionally, this section explores bias mitigation approaches through reweighting (§8.4.2) and whether unsupervised prior-matching, which was effective for NLP tasks, can enhance task performance for zero-shot audio classification.

Datasets

To evaluate the zero-shot audio classification capabilities of ASR foundation models, we select a diverse range of 8 audio classification datasets that span 6 different tasks. These tasks cover a broad spectrum of audio classification challenges, each assessing distinct forms of audio classification. The tasks and corresponding datasets are as follows:

- Sound Event Classification (SEC): The aim of SEC is to detect and identify specific auditory events within an audio clip. Two SEC datasets are used: First, **ESC50** [249], which comprises 50 distinct environmental sounds, including natural phenomena (e.g., rain, sea waves), human non-speech sounds (e.g., crying baby) and exterior urban noises (e.g. helicopter). The second dataset, **UrbanSound8K** [282], focuses exclusively on 10 distinct urban environments (e.g., sirens, car horns, drilling).

- Acoustic Scene Classification (ASC): The objective of ASC is to identify the overall acoustic environment of an audio clip. For ASC, the **TUT2017** dataset [216] is used, which features 15 acoustic scenes spanning both outdoor (e.g., city centre, beach, forest path) and indoor environments (e.g., metro station, library, home).

- Vocal Sound Classification (VSC): This task involves identifying non-verbal human vocal sounds. For this task, the **Vocal Sound** [93] dataset is used, which contains 6 distinct human vocal sound categories (e.g., laughter, crying, coughing).

- Speech Emotion Recognition (SER): The goal of SER is to detect and classify emotions conveyed in speech by analysing vocal features like tone, pitch, and energy. Two SER datasets are used, **RAVDESS** [198] and **CREMA-D** [31], which contain speakers

expressing 8 and 6 different common emotions, respectively, such as happiness, sadness and anger.

- Music Genre Classification (MGC): The objective of MGC is to distinguish between audios of different musical styles and identify the music genre. Here the **GTZAN** [304] dataset is used, which has 10 different music genres covering styles such as classical, jazz and rock.

- Speaker Counting (SC): This task assesses the model's capability to determine the number of distinct speakers in a given audio sample. For SC **LibriCount** [303] is used, which features audio clips with varying speaker counts from 0 to 10.

Complete dataset statistics, including the number of samples, duration of audio clips, and class distribution, are outlined in Table 8.1. Five of the datasets are balanced over the classes (ESC50, TUT2017, Vocal, GTZAN, and LibriCount), while the other three have slightly imbalanced distributions (as illustrated in Figure 8.5). These tasks and datasets allow for a comprehensive evaluation of the ASR foundation models' zero-shot audio classification capabilities across diverse tasks.

Task	Dataset	Utts	Avg. Dur.	K
SEC	ESC50	2,000	5.0	50
	UrbanSound8K	8,732	3.6	10
ASC	TUT2017	1,620	10.0	15
VSC	Vocal Sound	3,594	5.0	6
SER	RAVDESS	1,440	3.7	8
	CREMA-D	7,442	5.0	6
MGC	GTZAN	1,000	30.0	10
SC	LibriCount	5,720	5.0	11

Table 8.1 Test set statistics, displaying the total number of test utterances, the average duration of each audio sample (in seconds), and the number of classes K.

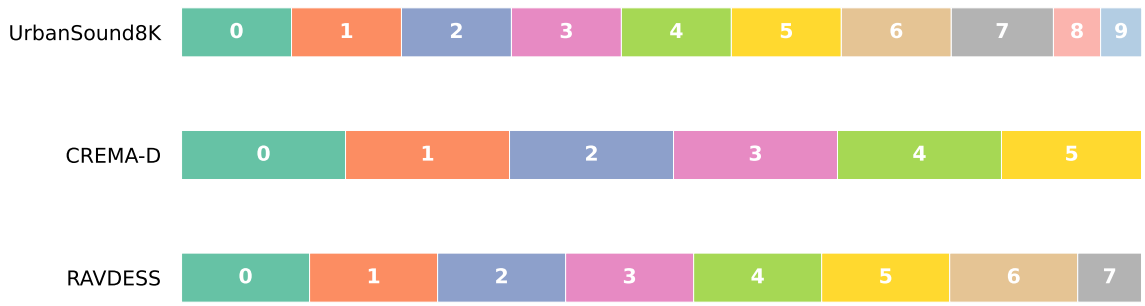


Fig. 8.5 Class distributions for audio classification datasets with non-uniform distributions. Each coloured segment represents the fraction of samples associated with a particular class.

ASR Foundation Models

Two families of ASR foundation models are investigated: Whisper and MMS, which have different architectures and training routines and represent the most capable open-source ASR foundation models at the point of conducting experiments. As described in Section 8.1.3, Whisper [263] is an encoder-decoder ASR system trained on weakly supervised transcripts. We investigate the larger Whisper models: Whisper medium.en (769M parameters) which is English-only, Whisper medium (769M parameters) which is multilingual and Whisper large-v2 (1.6B parameters) which is also multilingual. We also investigate MMS [255] (§8.1.2), an encoder-only ASR system with 965M parameters, which is a Wav2Vec 2.0 system fine-tuned using a CTC loss.

Zero-shot Prompting Set Up

Task	Prompt
SER	The speaker is feeling *class_label*.
MGC	This is an audio of *class_label* music.
SC	In the audio, *class_label* people are speaking.
others	This is a sound of *class_label*.

Table 8.2 Manually designed prompts used for each task. The bottom prompt is used for SEC, ASC and VSC.

The prompt templates used to create the label sequences for each class are shown in Table 8.2, adapted from the prompts used by Elizalde et al. [62]. The output probabilities are converted to class probabilities using three methods (§8.4.2):

- **uncalibrated:** the base probabilities by assuming the class probabilities are proportional to the associated ASR probabilities.
- **prior-matched:** this approach uses prior-matching, where we find the weights that result in a uniform marginalised class distribution. Similar to Section 6.5.1, we calculate the weights over the entire batch of test data (without access to any of the labels).
- **null-input:** the data-free null-input approximation, where synthetic clips of Gaussian white noise (with a variance set to 1) are generated and used to estimate the values of $\boldsymbol{\alpha}$. The synthetic clips are generated to have the same average duration as the task's clips.

The output ASR likelihoods are converted to probabilities using our proposed zero-shot audio classifier (§8.3) and rescaled using each of the above methods.

Baselines

We baseline performance against CLAP and AudioClip (§8.2.2). CLAP aligns a pre-trained text encoder with a pre-trained audio encoder using contrastive learning. BERT [51] is used as the text encoder, and CNN-14 [153] as the audio encoder. The model is trained using a sound event classification dataset and three audio captioning datasets. AudioCLIP [103] follows a similar approach, but instead of BERT and CNN-14, extends CLIP to also incorporate the audio modality by introducing an audio head and performing contrastive learning on AudioSet [87] to align the audio embeddings with the other modalities.

8.5.1 Impact of Class Bias

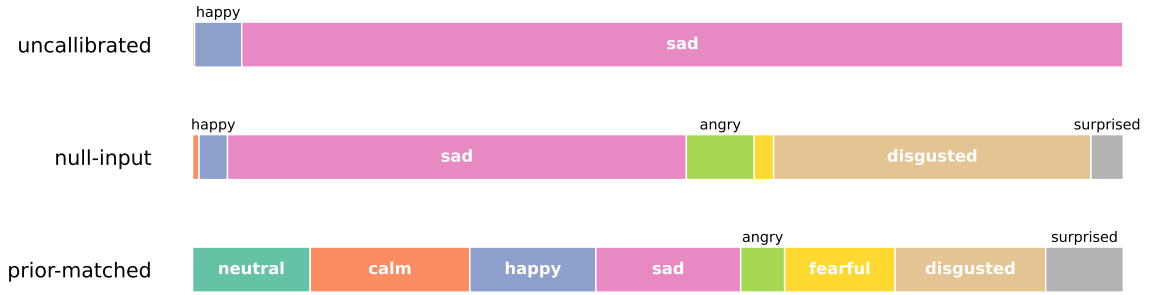
Model	ESC50	US8K	TUT	Vocal	RAVDESS	CREMA-D	GTZAN	LibriCount
Uniform	3.91	2.30	2.71	1.79	2.08	1.79	2.30	2.40
Whisper medium.en								
Uncalibrated	2.73	1.32	0.68	1.41	0.22	0.62	0.92	0.56
Null-Input	3.16	1.73	2.01	1.61	1.28	0.00	0.04	1.57
Prior-matched	3.86	2.28	2.61	1.79	2.02	1.78	2.23	2.38
Whisper large-v2								
Uncalibrated	3.15	1.76	0.76	1.34	0.22	0.56	1.38	0.18
Null-Input	3.29	2.04	2.13	1.50	0.62	0.27	1.78	1.42
Prior-matched	3.89	2.29	2.67	1.79	2.02	1.78	2.27	2.30

Table 8.3 Entropy of the distribution of the predictions when Whisper medium.en and Whisper large-v2 are prompted for various audio classification datasets.

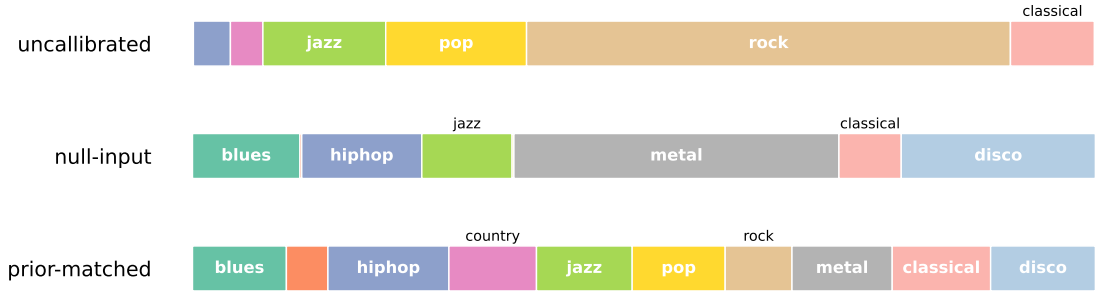
The initial experiments investigate whether prompting ASR foundation models for audio classification (§8.3) results in any significant class bias that may impact task performance. To examine this, Table 8.3 presents the entropy of the resulting class distribution of the predictions from Whisper medium.en and Whisper large-v2. The top row shows the entropy if the decisions are uniform across all classes.

The table demonstrates that the standard 'uncalibrated' approach results in non-uniform distributions where the models have strong preferences towards particular classes. For instance, when applied to ESC50, Whisper medium.en has an entropy of 2.73, which is lower than the entropy of a uniform distribution, 3.91. The resulting class imbalance can be visualised in Figure 8.6, which presents the distribution of Whisper large-v2 predictions for RAVDESS, GTZAN and Vocal. The figure reveals that the resulting predicted distribution can be significantly imbalanced, with classes such as 'sad' dominating for RAVDESS, or 'rock' for GTZAN.

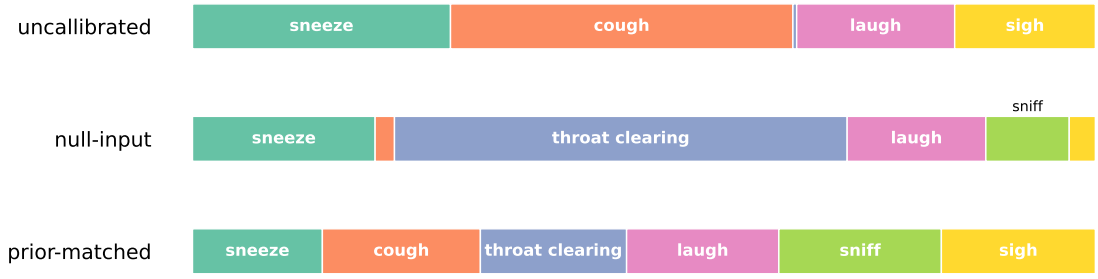
However, prior-matching, as intended, mitigates the resulting class bias and yields a near-uniform distribution over the classes, as seen in Figure 8.6. This is further evident from Table 8.3, where the resulting entropies are similar to those from the uniform distribution. The null-input approach can mitigate the bias and result in more balanced predicted distributions, but unlike in the experiments of Section 6.5.1, the approach is less stable and doesn't consistently adjust the prior in the desired direction.



(a) RAVDESS



(b) GTZAN



(c) Vocal

Fig. 8.6 Class distributions of predictions made by Whisper large-v2 across various datasets. The bar width is proportional to the fraction of decisions assigned to each class.

8.5.2 Zero-Shot Audio Classification Performance

Model	ESC50	US8K	TUT	Vocal	RAVDESS	CREMA-D	GTZAN	LibriCount	**Avg.**
Baselines									
Random	2.0	10.0	6.7	16.7	12.5	16.7	10.0	9.1	10.4
AudioCLIP	69.4	65.3	-	-	-	-	-	-	-
CLAP	82.6	73.2	29.6	49.4	16.0	17.8	25.2	17.9	39.0
Uncalibrated									
MMS large	1.7	9.6	4.9	14.2	13.5	17.2	8.3	8.4	9.7
Whisper medium.en	27.9	39.5	7.2	59.0	15.3	20.9	15.2	8.2	24.2
Whisper medium	29.7	45.8	7.5	44.6	16.7	19.9	28.4	9.4	25.2
Whisper large-v2	38.9	50.5	7.7	60.1	15.1	20.2	38.2	9.2	30.0
Null-Input									
MMS large	2.4	12.6	7.9	13.0	12.7	17.0	14.9	11.9	11.5
Whisper medium.en	36.8	45.8	20.0	68.9	17.2	20.4	10.0	8.9	28.5
Whisper medium	38.3	47.1	15.9	63.0	16.2	20.4	18.6	16.4	29.5
Whisper large-v2	43.8	53.2	22.1	62.4	20.4	18.8	50.8	15.0	35.8
Prior-matched									
MMS large	2.4	10.9	7.6	11.5	12.2	17.2	10.5	11.5	10.5
Whisper medium.en	56.2	60.9	18.3	82.8	29.0	22.6	29.7	9.8	38.7
Whisper medium	57.5	61.6	25.2	82.4	35.0	25.9	48.6	16.3	44.1
Whisper large-v2	65.4	60.4	26.0	84.9	41.7	28.8	60.9	17.3	**48.2**

Table 8.4 Baseline and zero-shot task performance using the audio prompts.

The next experiments examine the accuracy of ASR foundation models when prompted for zero-shot audio classification. Table 8.4 presents the accuracy of Whisper and MMS for various audio classification datasets, comparing their performance to random performance and the relevant zero-shot baselines. The table demonstrates that ASR foundation models can achieve above-random performance when prompted zero-shot for audio classification. All Whisper models achieve above-random accuracies on all tasks apart from LibriCount (which is a very challenging task and requires the model has to count the number of speakers in a clip). The Whisper models achieve average accuracies of 24.2%, 25.2% and 30.0%, significantly higher than the average random performance of 10.4%. MMS, however, proves ineffective for zero-shot audio classification. This can be attributed to the fact that MMS is trained with the CTC loss, where the model learns to independently map the acoustic features of each frame to characters. This may restrict the model to only capture frame-level features and not more global properties of the audio input. In contrast, Whisper's attention mechanism enables it to attend to the entire input sequence, allowing it to capture high-level

audio information across the input. Given these findings, any subsequent analysis will only focus on Whisper's zero-shot audio classification capabilities.

Further applying prior matching, which only requires unlabelled data, can result in substantial performance improvements across all tasks. For example, Whisper large-v2 achieves an accuracy of 65.4% with prior matching on ESC50, considerably higher than its uncalibrated performance of 38.9%. For tasks such as RAVDESS, the model initially appears to be not particularly capable, achieving an accuracy of 15.1%, only marginally above random performance of 12.5%. However, with prior matching, the same model can achieve an accuracy of 41.7%, highlighting that for many tasks, the class bias can potentially mask the model's true capabilities, but debiasing methods such as prior-matching can reveal more competitive performance.

Notably, zero-shot prompting of Whisper-large-v2 with prior matching outperforms CLAP in average accuracy. Despite CLAP being fine-tuned on sound event classification and audio captioning datasets, tasks similar to ESC50 and US8K, Whisper-large-v2 with zero-shot prompting and prior-matching surpasses CLAP by an average of 9.2%. This average accuracy includes the accuracy of ESC50 and US8K, and prompting Whisper results in consistent and substantial improvements across most out-of-domain tasks.

While prior matching requires unlabeled test data and is thus inapplicable to single or few-sample classification scenarios, the null-input approximation offers an alternative zero-resource debiasing approach. This method uses Gaussian noise to approximate the marginalised class prior. Null-input debiasing improves model performance by an average of 4.3%, 4.3%, and 5.8% for Whisper-medium.en, Whisper-medium, and Whisper-large-v2, respectively. These improvements indicate the potential of null-input methods for data-free calibration. However, unlike NLP applications, where null-input debiasing resulted in similar performance to prior-matching, there is a considerable performance gap between the two approaches in the audio domain. This disparity suggests that the null-input approximation may not accurately estimate the underlying marginalised class distribution in audio tasks. One possible explanation is that purely Gaussian noise does not truly mimic a neutral or "silence" scenario. Instead, it may inadvertently resemble static or certain natural background sounds (e.g., wind, hissing), thereby skewing the model's prior distribution in unanticipated ways.

8.5.3 Robustness to Prompts

The preceeding results demonstrated that Whisper can be effectively prompted for zero-shot audio classification, with prior-matching serving as a crucial technique to mitigate class bias and achieve competitive performance. In the realm of NLP, the performance of prompted

models is known to be sensitive to the prompts selected [360, 297]. The experiments in this section investigate whether Whisper displays similar prompt sensitivity in the audio domain. In the context of prompting Whisper, the 'prompt' refers to the template applied to the different classes, which is subsequently used to calculate corresponding ASR likelihoods. Tables 8.5 and 8.6 present the accuracies when using for various prompts, utilising Whisper large-v2 with prior-matching. The first prompt represents the default configuration used in the main experiments. Prompts 2-4 contain only the class label, which may resemble how sounds and actions would appear in video captions. While prompts 5-9 were generated by instructing ChatGPT to paraphrase prompt 1[4].

The results reveal that while zero-shot prompting can be effective across various prompts, there is considerable prompt sensitivity. Accuracy fluctuations are observed to be as high as 25%, 20%, and 15% for ESC50, RAVDESS, and Vocal datasets, respectively. An interesting observation is that although prompts 2-4 more closely resemble the type of captions that are likely to appear in the ASR decoding task, on average, the natural language prompts 5-9 demonstrate better performance for the SER datasets (RAVDESS, CREMA-D). This finding suggests that the zero-shot capability extends beyond mere ASR task transfer, indicating the emergence of more sophisticated task abilities acquired during pre-training.

Prompt #	Text	ESC50	US8K	TUT	Vocal
1	This is a sound of *class_label*.	65.4	60.4	26.0	84.9
2	*class_label*	48.6	54.8	15.7	60.1
3	*(class_label)*	68.0	65.5	21.3	86.3
4	*[class_label]*	64.3	64.2	16.1	85.9
5	Listen to the sound, it's called *class_label*.	50.3	56.5	16.0	81.7
6	The noise you hear is from the category *class_label*.	54.6	55.1	19.3	79.7
7	This is what we call *class_label* sound.	45.3	55.7	26.7	69.5
8	Identify this noise as *class_label*.	46.6	52.8	13.6	81.6
9	This sound belongs to the group *class_label*.	41.4	57.0	15.0	76.1
10	Ensemble of Prompts	67.1	67.6	25.2	87.3

Table 8.5 Prompt sensitivity for Sound Event Classification (SEC), Vocal Sound Classification (VSC) and Acoustic Scene Classification (ASC). Accuracies for Whisper large-v2 using prior-matching.

[4]using the instruction: *"Please paraphrase the given prompt five times using simple language:"*

Prompt #	Prompt	RAVDESS	CREMA-D
1	The speaker is feeling *class_label*.	41.7	28.8
2	*class_label*	1 20.7	18.1
3	*(class_label)*	33.1	35.3
4	*[class_label]*	32.6	26.6
5	The person talking feels *class_label*.	38.5	29.6
6	The speaker is experiencing *class_label* emotions.	20.8	20.5
7	The person speaking is in a *class_label* mood.	29.9	27.4
8	The speaker's emotion is *class_label*.	33.6	25.1
9	The person talking is filled with *class_label* feelings.	39.7	33.0
10	Ensemble of Prompts	44.0	33.1

Table 8.6 Prompt sensitivity for Speech Emotion Recognition (SER). Accuracies for Whisper large-v2 using prior-matching.

8.6 Chapter Summary

This chapter explored prompting to ASR foundation models for zero-shot audio classification. Though trained for ASR and translation tasks, Whisper exhibited impressive emergent classification performance that outperformed existing zero-shot approaches such as CLAP. Further, the zero-shot prompted systems were highly sensitive to the selected label word sequence, and a crucial element of achieving competitive performance was applying the reweighting debiasing methods applied in Chapter 6, and in particular prior-matching. This highlights the generalisability of our proposed weight debiasing method, which proves effective across different tasks, domains and modalities.

Chapter 9

Conclusion

The thesis investigated spurious correlations and bias in text-based foundation models, deriving methods of identifying and assessing bias, as well as methods to mitigate their influence. We further introduce LLM comparative assessment, a novel way of leveraging LLMs for zero-shot text assessment, as well as considerations of positional bias and extensions to make the approach more computationally practical. We also briefly consider the audio domain, and consider label-word bias for newly proposed zero-shot audio classifiers. This final chapter concludes the thesis and provides a review of the major contributions, as well as a brief discussion of possible limitations and continuations for future work. A breakdown of the contributions for each chapter is discussed next.

9.1 Review of Contributions

Chapter 5 investigates the spurious correlations that may arise when NLP foundation models are applied to tasks within the pre-train and fine-tuning paradigm. Existing work typically either focuses on artificially injecting spurious correlations and identifying their impact or highly restricts the feature space by only analysing simple word-level spurious correlations. We extend this analysis by considering spurious features derived from a richer feature space. By only extracting stopwords from the input text, which ensures that any derived feature is unrelated to the classification tasks of interest, we propose a novel spurious feature that may appear informative for the training data but is designed to be uninformative for the task. Our analysis demonstrates that models trained in a standard manner on various NLP tasks can achieve accuracies above random chance when given only the spurious shuffled stopword feature. This provides evidence that these NLP systems are highly capable of learning relationships that exhibit correlations in the training domain, including some that are clearly spurious. Although this form of spurious stopword correlations represents just one of many

possible spurious correlations, we show that models trained in standard settings learn the arbitrary stopword spurious correlation. We further provide insight into the impact of these correlations on real systems when applied out-of-domain and corroborate the finding that pre-training results in better robustness, with our experiments demonstrating that pre-trained systems are less susceptible to relying on spurious correlations.

In addition to extending the analysis of spurious correlations in NLP tasks, we also propose the concept of spurious questions: questions that test-takers can answer while bypassing the assessed attribute. We train 'shortcut' systems that do not have access to the passage when performing reading comprehension, and although reading comprehension should not be possible without the context, we demonstrate that these systems can achieve accuracies of over 50%. To the best of our knowledge, this is the first work to leverage such shortcut systems to identify potentially compromised questions that may not require comprehension, presenting a practical application of our spurious analysis. Through human evaluation, we further illustrate that our novel metrics of 'effective number of options' (ENO) and 'mutual information' can be useful for capturing aspects of question quality. Further, questions with the highest ENO could be answered correctly over 90% of the time by human test-takers who did not have access to the context.

Chapter 6 discusses the bias present in zero-shot LLMs and introduces possible mitigation schemes. This chapter makes the contribution of extending null-input calibration by proposing different weight optimization approaches based on data availabilities and formalising the reweighting debiasing approach. To the best of our knowledge, we are the first to propose using prior-matching for zero-shot classifiers and demonstrate that the method yields significant performance improvements and better robustness across many different NLP classification tasks. We show that prior-matching achieves performance similar to using the oracle optimal weights and can result in performance boosts of up to 30%, as seen on SNLI and MNLI. We further draw connections between existing data-free approaches [355] and prior-matching, demonstrating that the null-input approach can be interpreted as a data-free approximation of prior matching.

Furthermore, this chapter provides one of the first works to comprehensively analyse permutation bias across various multiple-choice tasks, and demonstrate the level of permutation bias across various models and datasets. To evaluate the level of permuation bias, we propose novel metrics, positional bias and permutation sensitivity, which better capture the bias than existing coarse metrics used by concurrent work. We also establish the effectiveness of permutation debiasing, which can result in performance improvements of up to 10% for models with significant position bias (such as Llama2).

Additionally, to overcome the computational costs associated with permutation debiasing, we introduce a framework to train efficient debiased students without the factorial ($K!$) costs associated with permutation debiasing. We show how both distillation and error-correction students can enable efficient, better performing and less biased predictions. Notably, we demonstrate that the error-correcting student, which uses and corrects only a single decision from the block box teacher, leads to a system with a better understanding of the teacher's systematic bias and provides less positionally biased predictions.

Chapter 7 introduces LLM comparative assessment, a novel method of using instruction-following LLMs for text quality assessment. We are the first work to investigate leveraging LLM judges to make pairwise decisions for NLG evaluation. We provide a comprehensive study considering a range of LLM judges, NLG tasks, and assessment attributes, demonstrating that LLM comparative assessment is more effective than prompt scoring when using moderately-sized LLMs (3B-13B parameters). Furthermore, LLM comparative assessment can achieve performance competitive with other state-of-the-art approaches; comparative assessment using a 3B LLM can achieve performance that is better than prompt-scoring when using LLMs with orders of magnitude more parameters. We also analyse the approach extensively, showing the impact of positional bias on the reliability of individual pairwise decisions. However, similar to multiple-choice question answering, we find that permutation debiasing, averaging the results across both permutations, is an effective debiasing approach that results in better agreement with human judgements.

Moreover, to address the computational concerns of LLM comparative assessment, we extend the approach to settings where only a subset of comparisons are used. We introduce the PoE framework for LLM comparative assessment, which provides a theoretical framework to effectively combine information from many different comparisons. We empirically show that the PoE framework of comparative assessment results in faster convergence to the system's best performance (when all comparisons are used) while scaling linearly with the number of candidate texts (rather than quadratically, like in standard comparative assessment). The contributions in these sections are essential for enabling comparative assessment to be a practical approach in real-world applications involving many texts to be assessed and ranked. Within the framework, we consider several expert variants and present the Gaussian expert (which has a closed-form solution), the soft-BT expert (which offers the best performance but requires an iterative optimisation algorithm), as well as experts that directly account for positional bias (enabling improved performance in regions with very few comparisons).

Finally, **Chapter 8** extends the zero-shot debiasing approaches to the speech modality. Our main contribution in this chapter is introducing a novel form of zero-shot audio prompting that generalises across a wide range of tasks. Unlike existing zero-shot methods—which rely on representation matching trained on similar audio tagging tasks and thus have limited performance for larger task shifts—our zero-shot audio prompting uses the associated ASR likelihood probabilities of relevant audio labels. We show that this approach achieves reasonable performance across diverse general audio classification tasks. However, we find that it is heavily impacted by label word bias, where the resulting class priors can be highly skewed towards particular classes depending on the chosen class words. By applying the same reweighting approaches described in Chapter 6, our zero-shot prompting method can surpass existing state-of-the-art audio classification performance across a wider range of tasks, highlighting the applicability of previous debiasing analyses.

9.2 Limitations and Future Work

- In our work, spurious correlations can only be identified for predefined candidate features (e.g., stopwords) or shortcut tasks (e.g., world knowledge). However, our approach cannot uncover unknown spurious correlations that a model might learn implicitly. For instance, models may exploit other syntactic patterns that have not been considered, and our work does not provide interpretability into the features a model may be using or whether they are spurious. Future work could investigate developing automated methods to detect emergent spurious correlations in trained models, such as using feature attribution techniques or adversarial probing.

- Although we demonstrate that proficient candidates can exploit the world knowledge shortcut, exams typically target a specific proficiency level where candidates can be validly assumed to have gaps in their world knowledge, making them unable to exploit the shortcut. Additionally, while metrics like 'effective number of options' (ENO) and 'mutual information' highlight questions answerable without context, they do not capture other critical aspects of question quality, such as ambiguity, cultural bias, or alignment with educational objectives. Future work could aim to build a more comprehensive automatic question quality analysis framework that incorporates evaluation of other question properties, such as discrimination, difficulty, and grammatical accuracy.

- In our work, we proposed the prior matching debiasing and the null-input debiasing approaches. A drawback of prior-matching is that it requires data (albeit without labels) and a larger batch of data to be available simultaneously. However, unlabelled

data may not always be accessible, and in cases where individual examples need to be classified, prior-matching is inapplicable. Additionally, prior-matching assumes that the class prior should be uniform, which may not apply to tasks with underlying class imbalances (e.g., fraud detection). Although null-input debiasing is unsupervised, it performs significantly worse than prior-matching. Future work could therefore explore alternative unsupervised optimization approaches that cater to different base assumptions or propose adaptive reweighting schemes that estimate task-specific priors from unlabelled data.

- Our experiments focused on zero-shot prompting and did not explore few-shot prompting, which would likely yield improved performance. This limitation was due to the constraints of the models used, which were state-of-the-art at the time of conducting the experiments but had insufficient input size to accommodate multiple examples effectively. With the advent of newer models capable of handling input lengths of up to 128k tokens, a more comprehensive investigation into few-shot learning could be conducted. This would include examining its impact on performance, positional biases, and new biases such as how the labels of the few-shot examples influence model behaviour. Future work could leverage these advancements to better understand the trade-offs between zero-shot and few-shot prompting, particularly in terms of performance, computational cost, and generalization. Additionally, exploring the role of example selection and prompt design in few-shot settings could provide further insights into optimizing model performance while mitigating biases.

- The Product of Experts framework for comparative assessment achieves its best performance when probabilities from the LLM are available. However, for some black-box APIs (e.g., GPT-4), these probabilities are inaccessible, leading to slower convergence and a higher number of required sampled comparisons. Future work could focus on developing methods to estimate uncertainty levels even without access to output probabilities. This could involve prompting the LLM to provide confidence estimates alongside its outputs or training lightweight surrogate models to approximate LLM confidence scores without requiring full access to logits. Additionally, while current comparative assessment decisions are binary, selecting which text is better, future work could explore ternary methods that incorporate an ‘uncertain’ or ‘tie’ option.

- While our work addresses positional bias through permutation-based debiasing in comparative assessment, it does not account for other potential biases that may influence decisions, such as preferences for verbose responses, writing styles, or length-based heuristics. The current approach assumes that positional bias is the primary confound-

ing factor in comparative assessment, potentially overlooking how these other biases may interact with or compound positional effects. Additionally, our analysis does not investigate how different model architectures or pre-training strategies might have different preferences that may result in unfair assessments. Future work could explore how these additional biases manifest across models and tasks, and develop more comprehensive debiasing techniques that address multiple sources of bias simultaneously. This could include designing multi-faceted evaluation frameworks that account for stylistic preferences, response length, and other latent factors, as well as studying the interplay between model architecture, training data, and assessment fairness.

9.3 Additional Publications

The following are additional publications, accepted at peer-reviewed conferences and done in collaboration with others during the course of my PhD, which were omitted from the thesis:

- [202] P. Manakul, **A. Liusie**, and M.J.F Gales. "MQAG: Multiple-choice Question Answering and Generation for Assessing Information Consistency in Summarization." Proceedings of the 13th International Joint Conference on Natural Language Processing and the 3rd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics (Volume 1: Long Papers). (**AACL 2023**)
- [203] P. Manakul, **A. Liusie**, and M.J.F Gales. "SelfCheckGPT: Zero-Resource Black-Box Hallucination Detection for Generative Large Language Models." Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing. (**EMNLP 2023**)
- [221] P. Molenda, **A. Liusie**, and M.J.F Gales. "WaterJudge: Quality-Detection Trade-off when Watermarking Large Language Models." Findings of the Association for Computational Linguistics: NAACL 2024. (**NAACL 2023 findings**)
- [69] Y Fathullah, P. Radmard, **A. Liusie**, and M.J.F Gales. "Who Needs Decoders? Efficient Estimation of Sequence-Level Attributes with Proxies." Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers). (**EACL 2024**)
- [269] V. Raina*, **A. Liusie***, and M.J.F Gales. "Is LLM-as-a-Judge Robust? Investigating Universal Adversarial Attacks on Zero-shot LLM Assessment", Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: (**EMNLP 2024**)

References

[1] Acerbi, A. and Stubbersfield, J. M. (2023). Large language models show human-like content biases in transmission chain experiments. *Proceedings of the National Academy of Sciences*, 120(44):e2313790120.

[2] Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., et al. (2023). Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

[3] Ainslie, J., Lee-Thorp, J., de Jong, M., Zemlyanskiy, Y., Lebron, F., and Sanghai, S. (2023). Gqa: Training generalized multi-query transformer models from multi-head checkpoints. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 4895–4901.

[4] Alderson, J. C. (2000). *Assessing reading*. Cambridge University Press.

[5] Alikaniotis, D., Yannakoudakis, H., and Rei, M. (2016). Automatic text scoring using neural networks. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 715–725.

[6] Anil, R., Borgeaud, S., Wu, Y., Alayrac, J.-B., Yu, J., Soricut, R., Schalkwyk, J., Dai, A. M., Hauth, A., et al. (2023). Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*.

[7] Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., and Ives, Z. (2007). Dbpedia: A nucleus for a web of open data. In *international semantic web conference*, pages 722–735. Springer.

[8] Ba, L. J., Kiros, J. R., and Hinton, G. E. (2016). Layer normalization. *CoRR*, abs/1607.06450.

[9] Bach, S., Sanh, V., Yong, Z. X., Webson, A., Raffel, C., Nayak, N. V., Sharma, A., Kim, T., Bari, M. S., Févry, T., et al. (2022). Promptsource: An integrated development environment and repository for natural language prompts. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 93–104.

[10] Baddeley, A. (2020). Working memory. In *Memory*, pages 71–111. Routledge.

[11] Baevski, A., Zhou, Y., Mohamed, A., and Auli, M. (2020). wav2vec 2.0: A framework for self-supervised learning of speech representations. *Advances in neural information processing systems*, 33:12449–12460.

[12] Baeza-Yates, R. (2018). Bias on the web. *Communications of the ACM*, 61(6):54–61.

[13] Bahdanau, D., Cho, K., and Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. In Bengio, Y. and LeCun, Y., editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

[14] Balaraman, V., Sheikhalishahi, S., and Magnini, B. (2021). Recent neural methods on dialogue state tracking for task-oriented dialogue systems: A survey. In *Proceedings of the 22nd annual meeting of the special interest group on discourse and dialogue*, pages 239–251.

[15] Banerjee, S. and Lavie, A. (2005). Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, pages 65–72.

[16] Barzilay, R. and Lapata, M. (2008). Modeling local coherence: An entity-based approach. *Computational Linguistics*, 34(1):1–34.

[17] Beaudoin, D. and Swartz, T. (2018). A computationally intensive ranking system for paired comparison data. *Operations Research Perspectives*, 5:105–112.

[18] Belz, A. and Reiter, E. (2006). Comparing automatic and human evaluation of NLG systems. In *11th Conference of the European Chapter of the Association for Computational Linguistics*, pages 313–320, Trento, Italy. Association for Computational Linguistics.

[19] Bengio, S., Vinyals, O., Jaitly, N., and Shazeer, N. (2015). Scheduled sampling for sequence prediction with recurrent neural networks. *Advances in neural information processing systems*, 28.

[20] Bengio, Y., Simard, P., and Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166.

[21] Bertrand, M. and Mullainathan, S. (2004). Are emily and greg more employable than lakisha and jamal? a field experiment on labor market discrimination. *American Economic Review*, 94(4):991–1013.

[22] Bird, S., Klein, E., and Loper, E. (2009). *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.".

[23] Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer. Covers the relationship between features and target variables, including correlation in machine learning.

[24] Bommasani, R., Hudson, D. A., Adeli, E., Altman, R., Arora, S., von Arx, S., Bernstein, M. S., Bohg, J., Bosselut, A., Brunskill, E., et al. (2021). On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*.

[25] Bottou, L. (2010). Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010: 19th International Conference on Computational StatisticsParis France, August 22-27, 2010 Keynote, Invited and Contributed Papers*, pages 177–186. Springer.

[26] Bowman, S. R., Angeli, G., Potts, C., and Manning, C. D. (2015). A large annotated corpus for learning natural language inference. In Màrquez, L., Callison-Burch, C., and Su, J., editors, *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642, Lisbon, Portugal. Association for Computational Linguistics.

[27] Bradley, R. A. and Terry, M. E. (1952). Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39(3/4):324–345.

[28] Bramley, T. (2005). A rank-ordering method for equating tests by expert judgment. *JOURNAL OF APPLIED MEASUREMENT*, 6(2):202–223.

[29] Bramley, T. (2015). Investigating the reliability of adaptive comparative judgment. *Cambridge Assessment, Cambridge*, 36.

[30] Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. (2020). Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

[31] Cao, H., Cooper, D. G., Keutmann, M. K., Gur, R. C., Nenkova, A., and Verma, R. (2014). CREMA-D: Crowd-sourced emotional multimodal actors dataset. *IEEE transactions on affective computing*, 5(4):377–390.

[32] Cao, Z., Qin, T., Liu, T.-Y., Tsai, M.-F., and Li, H. (2007). Learning to rank: from pairwise approach to listwise approach. In *Proceedings of the 24th international conference on Machine learning*, pages 129–136.

[33] Chatziagapi, A., Paraskevopoulos, G., Sgouropoulos, D., Pantazopoulos, G., Nikandrou, M., Giannakopoulos, T., Katsamanis, A., Potamianos, A., and Narayanan, S. (2019). Data augmentation using gans for speech emotion recognition. In *Interspeech*, pages 171–175.

[34] Chen, H. and Garner, P. N. (2024). Bayesian parameter-efficient fine-tuning for overcoming catastrophic forgetting. *arXiv preprint arXiv:2402.12220*.

[35] Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. (2020). A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR.

[36] Chhun, C., Colombo, P., Suchanek, F., and Clavel, C. (2022). Of human criteria and automatic metrics: A benchmark of the evaluation of story generation. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 5794–5836.

[37] Child, R., Gray, S., Radford, A., and Sutskever, I. (2019). Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*.

[38] Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

[39] Choi, H., Kim, J., Joe, S., and Gwon, Y. (2021). Evaluation of bert and albert sentence embedding performance on downstream nlp tasks. In *2020 25th International conference on pattern recognition (ICPR)*, pages 5482–5487. IEEE.

[40] Chomsky, N. (1957). Syntactic structures. *The Hague: Mouton*, pages 12–30.

[41] Chowdhery, A., Narang, S., Devlin, J., Bosma, M., Mishra, G., Roberts, A., Barham, P., Chung, H. W., Sutton, C., Gehrmann, S., et al. (2023). Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113.

[42] Chung, H. W., Hou, L., Longpre, S., Zoph, B., Tay, Y., Fedus, W., Li, E., Wang, X., Dehghani, M., Brahma, S., et al. (2022). Scaling instruction-finetuned language models. *arXiv preprint arXiv:2210.11416*.

[43] Ciregan, D., Meier, U., and Schmidhuber, J. (2012). Multi-column deep neural networks for image classification. In *2012 IEEE conference on computer vision and pattern recognition*, pages 3642–3649. IEEE.

[44] Clark, C., Lee, K., Chang, M.-W., Kwiatkowski, T., Collins, M., and Toutanova, K. (2019). Boolq: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2924–2936.

[45] Clark, K., Luong, M.-T., Le, Q. V., and Manning, C. D. (2020). Electra: Pre-training text encoders as discriminators rather than generators. In *International Conference on Learning Representations*.

[46] Clark, P., Cowhey, I., Etzioni, O., Khot, T., Sabharwal, A., Schoenick, C., and Tafjord, O. (2018). Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.

[47] Collobert, R. and Weston, J. (2008). A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167.

[48] Csató, L. (2013). Ranking by pairwise comparisons for swiss-system tournaments. *Central European Journal of Operations Research*, 21:783–803.

[49] de Vries, W., Van Cranenburgh, A., and Nissim, M. (2020). What's so special about bert's layers? a closer look at the nlp pipeline in monolingual and multilingual models. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4339–4350.

[50] DeGroot, M. H. and Fienberg, S. E. (1983). The comparison and evaluation of forecasters. In *The Statistician*, volume 32, pages 12–22. JSTOR.

[51] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

[52] Dietterich, T. (1995). Overfitting and undercomputing in machine learning. *ACM computing surveys (CSUR)*, 27(3):326–327.

[53] Doersch, C., Gupta, A., and Efros, A. A. (2015). Unsupervised visual representation learning by context prediction. In *Proceedings of the IEEE international conference on computer vision*, pages 1422–1430.

[54] Dogra, V., Verma, S., Kavita, Chatterjee, P., Shafi, J., Choi, J., and Ijaz, M. F. (2022). A complete process of text classification system using state-of-the-art nlp models. *Computational Intelligence and Neuroscience*, 2022(1):1883698.

[55] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. (2021). An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.

[56] Dranker, Y., He, H., and Belinkov, Y. (2021). Irm—when it works and when it doesn't: A test case of natural language inference. *Advances in Neural Information Processing Systems*, 34:18212–18224.

[57] Du, M., Manjunatha, V., Jain, R., Deshpande, R., Dernoncourt, F., Gu, J., Sun, T., and Hu, X. (2021). Towards interpreting and mitigating shortcut learning behavior of nlu models. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 915–929.

[58] Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Yang, A., Fan, A., et al. (2024). The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

[59] Duchi, J., Hazan, E., and Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7).

[60] Durmus, E., He, H., and Diab, M. (2020). Feqa: A question answering evaluation framework for faithfulness assessment in abstractive summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5055–5070.

[61] Dykstra, O. (1956). A note on the rank analysis of incomplete block designs–applications beyond the scope of existing tables. *Biometrics*, 12(3):301–306.

[62] Elizalde, B., Deshmukh, S., Al Ismail, M., and Wang, H. (2023). Clap learning audio concepts from natural language supervision. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE.

[63] Elman, J. L. (1990). Finding structure in time. *Cognitive science*, 14(2):179–211.

[64] Elo, A. E. and Sloan, S. (1978). The rating of chessplayers: Past and present. *(No Title)*.

[65] Etzioni, O., Banko, M., Soderland, S., and Weld, D. S. (2008). Open information extraction from the web. *Communications of the ACM*, 51(12):68–74.

[66] Eyal, M., Baumel, T., and Elhadad, M. (2019). Question answering as an automatic evaluation metric for news article summarization. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3938–3948.

[67] Fabbri, A. R., Kryściński, W., McCann, B., Xiong, C., Socher, R., and Radev, D. (2021). Summeval: Re-evaluating summarization evaluation. *Transactions of the Association for Computational Linguistics*, 9:391–409.

[68] Fan, A., Lewis, M., and Dauphin, Y. (2018). Hierarchical neural story generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 889–898.

[69] Fathullah, Y., Radmard, P., Liusie, A., and Gales, M. (2024). Who needs decoders? efficient estimation of sequence-level attributes with proxies. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1478–1496.

[70] Feder, A., Keith, K. A., Manzoor, E., Pryzant, R., Sridhar, D., Wood-Doughty, Z., Eisenstein, J., Grimmer, J., Reichart, R., Roberts, M. E., et al. (2022). Causal inference in natural language processing: Estimation, prediction, interpretation and beyond. *Transactions of the Association for Computational Linguistics*, 10:1138–1158.

[71] Firth, D. and Turner, H. (2012). Bradley-terry models in r: the bradleyterry2 package. *Journal of Statistical Software*, 48(9).

[72] Fiske, S. T. (2002). What we know now about bias and intergroup conflict, the problem of the century. *Current Directions in Psychological Science*, 11(4):123–128.

[73] Fox, J. D. and Delworth, N. (2022). Improving contextual recognition of rare words with an alternate spelling prediction model. *arXiv preprint arXiv:2209.01250*.

[74] French, R. M. (1999). Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3(4):128–135.

[75] Friedman, J., Hastie, T., and Tibshirani, R. (2001). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer. Discusses correlation and feature-target relationships in supervised learning models.

[76] Fu, J., Ng, S.-K., Jiang, Z., and Liu, P. (2023). Gptscore: Evaluate as you desire. *arXiv preprint arXiv:2302.04166*.

[77] Gal, Y. and Ghahramani, Z. (2016). Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059. PMLR.

[78] Gandomi, A. and Haider, M. (2015). Beyond the hype: Big data concepts, methods, and analytics. *International journal of information management*, 35(2):137–144.

[79] Ganguli, D., Askell, A., Schiefer, N., Liao, T. I., Lukošiūtė, K., Chen, A., Goldie, A., Mirhoseini, A., Olsson, C., Hernandez, D., et al. (2023). The capacity for moral self-correction in large language models. *arXiv preprint arXiv:2302.07459*.

[80] Gao, T., Fisch, A., and Chen, D. (2021). Making pre-trained language models better few-shot learners. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3816–3830.

[81] Gardent, C., Shimorina, A., Narayan, S., and Perez-Beltrachini, L. (2017a). Creating training corpora for NLG micro-planners. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 179–188, Vancouver, Canada. Association for Computational Linguistics.

[82] Gardent, C., Shimorina, A., Narayan, S., and Perez-Beltrachini, L. (2017b). The webnlg challenge: Generating text from rdf data. In *10th International Conference on Natural Language Generation*, pages 124–133. ACL Anthology.

[83] Gardner, M., Artzi, Y., Basmov, V., Berant, J., Bogin, B., Chen, S., Dasigi, P., Dua, D., Elazar, Y., Gottumukkala, A., et al. (2020). Evaluating models' local decision boundaries via contrast sets. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1307–1323.

[84] Gardner, M., Merrill, W., Dodge, J., Peters, M. E., Ross, A., Singh, S., and Smith, N. A. (2021). Competency problems: On finding and removing artifacts in language data. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 1801–1813.

[85] Geirhos, R., Jacobsen, J.-H., Michaelis, C., Zemel, R., Brendel, W., Bethge, M., and Wichmann, F. A. (2020). Shortcut learning in deep neural networks. *Nature Machine Intelligence*, 2(11):665–673.

[86] Gekhman, Z., Herzig, J., Aharoni, R., Elkind, C., and Szpektor, I. (2023). Trueteacher: Learning factual consistency evaluation with large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 2053–2070.

[87] Gemmeke, J. F., Ellis, D. P., Freedman, D., Jansen, A., Lawrence, W., Moore, R. C., Plakal, M., and Ritter, M. (2017). Audio set: An ontology and human-labeled dataset for audio events. In *2017 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 776–780. IEEE.

[88] Gigerenzer, G. and Gaissmaier, W. (2011). Heuristic decision making. *Annual Review of Psychology*, 62:451–482.

[89] Gill, T., Bramley, T., and Black, B. (2007). An investigation of standard maintaining in gcse english using a rank-ordering method. In *British Educational Research Association Annual Conference*.

[90] Glorot, X., Bordes, A., and Bengio, Y. (2011). Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 315–323. JMLR Workshop and Conference Proceedings.

[91] Golub, G. H. and Van Loan, C. F. (2013). *Matrix computations*. JHU press.

[92] Gong, Y., Khurana, S., Karlinsky, L., and Glass, J. (2023). Whisper-at: Noise-robust automatic speech recognizers are also strong general audio event taggers. In *Annual Conference of the International Speech Communication Association*.

[93] Gong, Y., Yu, J., and Glass, J. (2022). Vocalsound: A dataset for improving human vocal sounds recognition. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 151–155. IEEE.

[94] Goodrich, B., Rao, V., Liu, P. J., and Saleh, M. (2019). Assessing the factual accuracy of generated text. In *proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 166–175.

[95] Goswami, K., Park, Y., and Song, C. (2017). Impact of reviewer social interaction on online consumer review fraud detection. *Journal of big data*, 4:1–19.

[96] Graves, A., Fernández, S., Gomez, F. J., and Schmidhuber, J. (2006). Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd international conference on Machine learning*, pages 369–376.

[97] Graves, A. and Schmidhuber, J. (2005). Framewise phoneme classification with bidirectional lstm and other neural network architectures. In *Neural Networks*, volume 18, pages 602–610. Elsevier.

[98] Gui, J., Chen, T., Zhang, J., Cao, Q., Sun, Z., Luo, H., and Tao, D. (2024). A survey on self-supervised learning: Algorithms, applications, and future trends. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

[99] Guo, C., Pleiss, G., Sun, Y., and Weinberger, K. Q. (2017). On calibration of modern neural networks. In *International conference on machine learning*, pages 1321–1330. PMLR.

[100] Gupta, R., Halevy, A., Wang, X., Whang, S. E., and Wu, F. (2014). Biperpedia: An ontology for search applications. *Proceedings of the VLDB Endowment*, 7(7):505–516.

[101] Gupta, U., Dhamala, J., Kumar, V., Verma, A., Pruksachatkun, Y., Krishna, S., Gupta, R., Chang, K.-W., Ver Steeg, G., and Galstyan, A. (2022). Mitigating gender bias in distilled language models via counterfactual role reversal. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 658–678.

[102] Gururangan, S., Swayamdipta, S., Levy, O., Schwartz, R., Bowman, S., and Smith, N. A. (2018). Annotation artifacts in natural language inference data. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 107–112.

[103] Guzhov, A., Raue, F., Hees, J., and Dengel, A. (2022). AudioCLIP: Extending clip to image, text and audio. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 976–980. IEEE.

[104] Hadsell, R., Chopra, S., and LeCun, Y. (2006). Dimensionality reduction by learning an invariant mapping. In *2006 IEEE computer society conference on computer vision and pattern recognition (CVPR'06)*, volume 2, pages 1735–1742. IEEE.

[105] Han, J., Yoo, H., Myung, J., Kim, M., Lim, H., Kim, Y., Lee, T. Y., Hong, H., Kim, J., Ahn, S.-Y., et al. (2023). Fabric: Automated scoring and feedback generation for essays. *arXiv preprint arXiv:2310.05191*.

[106] Handley, J. C. (2001). Comparative analysis of bradley-terry and thurstone-mosteller paired comparison models for image quality assessment. In *PICS*, volume 1, pages 108–112.

[107] Hanley, J. A. and McNeil, B. J. (1982). The meaning and use of the area under a receiver operating characteristic (roc) curve. *Radiology*, 143(1):29–36.

[108] He, K., Chen, X., Xie, S., Li, Y., Dollár, P., and Girshick, R. (2022). Masked autoencoders are scalable vision learners. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16000–16009.

[109] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

[110] He, P., Liu, X., Gao, J., and Chen, W. (2021). Deberta: Decoding-enhanced bert with disentangled attention. In *International Conference on Learning Representations*.

[111] Hendrycks, D., Burns, C., Basart, S., Zou, A., Mazeika, M., Song, D., and Steinhardt, J. (2021). Measuring massive multitask language understanding. In *International Conference on Learning Representations*.

[112] Hendrycks, D. and Gimpel, K. (2016). Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*.

[113] Hendrycks, D., Liu, X., Wallace, E., Dziedzic, A., Krishnan, R., and Song, D. (2020). Pretrained transformers improve out-of-distribution robustness. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2744–2751.

[114] Herbrich, R., Minka, T., and Graepel, T. (2006). Trueskill™: a bayesian skill rating system. *Advances in neural information processing systems*, 19.

[115] Hershey, S., Chaudhuri, S., Ellis, D. P., Gemmeke, J. F., Jansen, A., Moore, R. C., Plakal, M., Platt, D., Saurous, R. A., Seybold, B., et al. (2017). Cnn architectures for large-scale audio classification. In *2017 ieee international conference on acoustics, speech and signal processing (icassp)*, pages 131–135. IEEE.

[116] Hinton, G., Srivastava, N., and Swersky, K. (2012a). Neural networks for machine learning lecture 6a overview of mini-batch gradient descent. *Cited on*, 14(8):2.

[117] Hinton, G., Vinyals, O., and Dean, J. (2015). Distilling the knowledge in a neural network. In *NIPS Deep Learning and Representation Learning Workshop*.

[118] Hinton, G. E. (1999). Products of experts. In *Artificial Neural Networks, 1999. ICANN 99. Ninth International Conference on (Conf. Publ. No. 470)*, volume 1, pages 1–6. IET.

[119] Hinton, G. E., Osindero, S., and Teh, Y.-W. (2006). A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554.

[120] Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. R. (2012b). Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*.

[121] Hochreiter, S. (1998). The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(02):107–116.

[122] Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

[123] Hoffmann, J., Borgeaud, S., Mensch, A., Buchatskaya, E., Cai, T., Rutherford, E., Casas, D. d. L., Hendricks, L. A., Welbl, J., Clark, A., et al. (2022). Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*.

[124] Holtzman, A., Buys, J., Du, L., Forbes, M., and Choi, Y. (2020). The curious case of neural text degeneration. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

[125] Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, 79(8):2554–2558.

[126] Hou, Y., Li, J., He, Z., Yan, A., Chen, X., and McAuley, J. (2024). Bridging language and items for retrieval and recommendation. *arXiv preprint arXiv:2403.03952*.

[127] Howard, J. and Ruder, S. (2018). Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 328–339.

[128] Hsu, W.-N., Bolte, B., Tsai, Y.-H. H., Lakhotia, K., Salakhutdinov, R., and Mohamed, A. (2021). Hubert: Self-supervised speech representation learning by masked prediction of hidden units. *IEEE/ACM transactions on audio, speech, and language processing*, 29:3451–3460.

[129] Hu, E. J., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., Chen, W., et al. (2021). Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

[130] Huang, L., Le Bras, R., Bhagavatula, C., and Choi, Y. (2019). Cosmos QA: Machine reading comprehension with contextual commonsense reasoning. In Inui, K., Jiang, J., Ng, V., and Wan, X., editors, *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2391–2401, Hong Kong, China. Association for Computational Linguistics.

[131] Hunter, D. R. (2004). Mm algorithms for generalized bradley-terry models. *The annals of statistics*, 32(1):384–406.

[132] Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. pmlr.

[133] Itzhak, I., Stanovsky, G., Rosenfeld, N., and Belinkov, Y. (2024). Instructed to bias: Instruction-tuned language models exhibit emergent cognitive bias. *Transactions of the Association for Computational Linguistics*, 12:771–785.

[134] Jacowitz, K. E. and Kahneman, D. (1995). Measures of anchoring in estimation tasks. *Personality and Social Psychology Bulletin*, 21(11):1161–1166.

[135] Jahangir, R., Teh, Y. W., Hanif, F., and Mujtaba, G. (2021). Deep learning approaches for speech emotion recognition: State of the art and research challenges. *Multimedia Tools and Applications*, 80(16):23745–23812.

[136] Jamieson, K. and Hyland, P. (2006). Good intuition or fear and uncertainty: The effects of bias on information systems selection decisions. *Informing Science*, 9:49.

[137] Jang, E., Gu, S., and Poole, B. (2016). Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*.

[138] Jegou, H., Douze, M., and Schmid, C. (2010). Product quantization for nearest neighbor search. *IEEE transactions on pattern analysis and machine intelligence*, 33(1):117–128.

[139] Jiang, A. Q., Sablayrolles, A., Mensch, A., Bamford, C., Chaplot, D. S., de las Casas, D., Bressand, F., Lengyel, G., Lample, G., Saulnier, L., Lavaud, L. R., Lachaux, M.-A., Stock, P., Scao, T. L., Lavril, T., Wang, T., Lacroix, T., and Sayed, W. E. (2023). Mistral 7b.

[140] Jiang, H., He, P., Chen, W., Liu, X., Gao, J., and Zhao, T. (2020a). Smart: Robust and efficient fine-tuning for pre-trained natural language models through principled regularized optimization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2177–2190.

[141] Jiang, Z., Xu, F. F., Araki, J., and Neubig, G. (2020b). How can we know what language models know? *Transactions of the Association for Computational Linguistics*, 8:423–438.

[142] Jones, I. and Inglis, M. (2015). The problem of assessing problem solving: Can comparative judgement help? *Educational Studies in Mathematics*, 89:337–355.

[143] Joshi, N., Pan, X., and He, H. (2022). Are all spurious features in natural language alike? an analysis through a causal lens. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 9804–9817.

[144] Jurafsky, D. and Martin, J. H. (2009). *Speech and Language Processing*. Prentice Hall.

[145] Kahn, J., Riviere, M., Zheng, W., Kharitonov, E., Xu, Q., Mazaré, P.-E., Karadayi, J., Liptchinsky, V., Collobert, R., Fuegen, C., and Likhomanenko, T. (2020). Libri-light: A benchmark for asr with limited or no supervision. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7669–7673. IEEE.

[146] Kahneman, D., Frederick, S., et al. (2002). Representativeness revisited: Attribute substitution in intuitive judgment. *Heuristics and biases: The psychology of intuitive judgment*, 49(49-81):74.

[147] Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., and Amodei, D. (2020). Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*.

[148] Kaushik, D., Hovy, E., and Lipton, Z. (2020). Learning the difference that makes a difference with counterfactually-augmented data. In *International Conference on Learning Representations*.

[149] Keith, K., Jensen, D., and O'Connor, B. (2020). Text and causal inference: A review of using text to remove confounding from causal estimates. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5332–5344.

[150] Kilgarriff, A. (1997). I don't believe in word senses. *Computers and the Humanities*, 31:91–113.

[151] Kim, S., Shin, J., Cho, Y., Jang, J., Longpre, S., Lee, H., Yun, S., Shin, S., Kim, S., Thorne, J., et al. (2023). Prometheus: Inducing fine-grained evaluation capability in language models. In *The Twelfth International Conference on Learning Representations*.

[152] Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. In Bengio, Y. and LeCun, Y., editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

[153] Kong, Q., Cao, Y., Iqbal, T., Wang, Y., Wang, W., and Plumbley, M. D. (2020). Panns: Large-scale pretrained audio neural networks for audio pattern recognition. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 28:2880–2894.

[154] Koroteev, M. V. (2021). Bert: a review of applications in natural language processing and understanding. *arXiv preprint arXiv:2103.11943*.

[155] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25.

[156] Kudo, T. and Richardson, J. (2018). Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Association for Computational Linguistics.

[157] Kullback, S. and Leibler, R. A. (1951). On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86.

[158] Lai, A. and Tetreault, J. (2018a). Discourse coherence in the wild: A dataset, evaluation and methods. In *Proceedings of the 19th Annual SIGdial Meeting on Discourse and Dialogue*, pages 214–223, Melbourne, Australia. Association for Computational Linguistics.

[159] Lai, A. and Tetreault, J. (2018b). Discourse coherence in the wild: A dataset, evaluation and methods. In *Proceedings of the 19th Annual SIGdial Meeting on Discourse and Dialogue*, pages 214–223.

[160] Lai, G., Xie, Q., Liu, H., Yang, Y., and Hovy, E. (2017). Race: Large-scale reading comprehension dataset from examinations. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 785–794.

[161] Laming, D. (2004). *Human Judgment The eye of the beholder*.

[162] Lauscher, A., Glavaš, G., Ponzetto, S. P., and Vulić, I. (2020). A general framework for implicit and explicit debiasing of distributional word vector spaces. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8131–8138.

[163] Lauscher, A., Lueken, T., and Glavaš, G. (2021). Sustainable modular debiasing of language models. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 4782–4797.

[164] LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *nature*, 521(7553):436–444.

[165] LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

[166] LeCun, Y., Huang, F. J., and Bottou, L. (2006). A tutorial on energy-based learning. In *Predicting Structured Data*.

[167] Leshno, M., Lin, V. Y., Pinkus, A., and Schocken, S. (1993). Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural networks*, 6(6):861–867.

[168] Lester, B., Al-Rfou, R., and Constant, N. (2021). The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

[169] Levin, D. A. and Peres, Y. (2017). *Markov chains and mixing times*, volume 107. American Mathematical Soc.

[170] Levin, I. P., Schneider, S. L., and Gaeth, G. J. (1998). All frames are not created equal: A typology and critical analysis of framing effects. *Organizational behavior and human decision processes*, 76(2):149–188.

[171] Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., and Zettlemoyer, L. (2020). Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, page 7871. Association for Computational Linguistics.

[172] Li, J., Chen, J., Sheng, B., Li, P., Yang, P., Feng, D. D., and Qi, J. (2022). Automatic detection and classification system of domestic waste via multimodel cascaded convolutional neural network. *IEEE Transactions on Industrial Informatics*, 18(1):163–173.

[173] Li, J., Tang, Z., Liu, X., Spirtes, P., Zhang, K., Leqi, L., and Liu, Y. (2024a). Steering llms towards unbiased responses: A causality-guided debiasing framework. In *ICLR 2024 Workshop on Reliable and Responsible Foundation Models*.

[174] Li, Z., Xu, X., Shen, T., Xu, C., Gu, J.-C., and Tao, C. (2024b). Leveraging large language models for nlg evaluation: A survey. *arXiv preprint arXiv:2401.07103*.

[175] Liang, Y., Li, J., and Yin, J. (2019). A new multi-choice reading comprehension dataset for curriculum learning. In *Proceedings of The Eleventh Asian Conference on Machine Learning*, pages 742–757.

[176] Lin, C.-Y. (2004). Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

[177] Lin, C.-Y. and Och, F. J. (2004). Automatic evaluation of machine translation quality using longest common subsequence and skip-bigram statistics. In *Proceedings of the 42nd annual meeting of the association for computational linguistics (ACL-04)*, pages 605–612.

[178] Lin, R. and Ng, H. T. (2023). Mind the biases: Quantifying cognitive biases in language model prompting. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 5269–5281.

[179] Lita, L., Rogati, M., and Lavie, A. (2005). BLANC: Learning evaluation metrics for MT. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 740–747, Vancouver, British Columbia, Canada. Association for Computational Linguistics.

[180] Liu, F., Lin, K., Li, L., Wang, J., Yacoob, Y., and Wang, L. (2023a). Mitigating hallucination in large multi-modal models via robust instruction tuning. In *The Twelfth International Conference on Learning Representations*.

[181] Liu, T.-Y. et al. (2009). Learning to rank for information retrieval. *Foundations and Trends® in Information Retrieval*, 3(3):225–331.

[182] Liu, Y., Iter, D., Xu, Y., Wang, S., Xu, R., and Zhu, C. (2023b). G-eval: NLG evaluation using gpt-4 with better human alignment. In Bouamor, H., Pino, J., and Bali, K., editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 2511–2522, Singapore. Association for Computational Linguistics.

[183] Liu, Y. and Lapata, M. (2019). Text summarization with pretrained encoders. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3730–3740.

[184] Liu, Y., Moosavi, N. S., and Lin, C. (2023c). Llms as narcissistic evaluators: When ego inflates evaluation scores. *arXiv preprint arXiv:2311.09766*.

[185] Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. (2019). Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

[186] Liusie, A., Fathullah, Y., and Gales, M. J. (2024a). Teacher-student training for debiasing: General permutation debiasing for large language models. *arXiv preprint arXiv:2403.13590*.

[187] Liusie, A., Manakul, P., and Gales, M. (2023a). Mitigating word bias in zero-shot prompt-based classifiers. In *Findings of the Association for Computational Linguistics: IJCNLP-AACL 2023 (Findings)*, pages 327–335.

[188] Liusie, A., Manakul, P., and Gales, M. (2024b). Llm comparative assessment: Zero-shot nlg evaluation through pairwise comparisons using large language models. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 139–151.

[189] Liusie, A., Manakul, P., and Gales, M. (2024c). LLM comparative assessment: Zero-shot NLG evaluation through pairwise comparisons using large language models. In Graham, Y. and Purver, M., editors, *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 139–151, St. Julian's, Malta. Association for Computational Linguistics.

[190] Liusie, A., Raina, V., Fathullah, Y., and Gales, M. (2024d). Efficient llm comparative assessment: a product of experts framework for pairwise comparisons. *arXiv preprint arXiv:2405.05894*.

[191] Liusie, A., Raina, V., and Gales, M. (2023b). "world knowledge" in multiple choice reading comprehension. In *Proceedings of the Sixth Fact Extraction and VERification Workshop (FEVER)*, pages 49–57, Dubrovnik, Croatia. Association for Computational Linguistics.

[192] Liusie, A., Raina, V., Raina, V., and Gales, M. (2022). Analyzing biases to spurious correlations in text classification tasks. In *Proceedings of the 2nd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 12th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 78–84.

[193] Longpre, S., Hou, L., Vu, T., Webson, A., Chung, H. W., Tay, Y., Zhou, D., Le, Q. V., Zoph, B., Wei, J., et al. (2023). The flan collection: Designing data and methods for effective instruction tuning. In *International Conference on Machine Learning*, pages 22631–22648. PMLR.

[194] Loshchilov, I. and Hutter, F. (2017). Decoupled weight decay regularization. In *International Conference on Learning Representations*.

[195] Louviere, J. J., Hensher, D. A., and Swait, J. D. (2000). *Stated choice methods: analysis and applications*. Cambridge university press.

[196] Lu, Z., Pu, H., Wang, F., Hu, Z., and Wang, L. (2017). The expressive power of neural networks: A view from the width. *Advances in neural information processing systems*, 30.

[197] Luce, R. D. (1959). *Individual choice behavior*, volume 4. Wiley New York.

[198] Luna-Jiménez, C., Kleinlein, R., Griol, D., Callejas, Z., Montero, J. M., and Fernández-Martínez, F. (2021). A proposal for multimodal emotion recognition using aural transformers and action units on RAVDESS dataset. *Applied Sciences*, 12(1):327.

[199] Ma, R., Liusie, A., Gales, M., and Knill, K. (2024). Investigating the emergent audio classification ability of asr foundation models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 4746–4760.

[200] Maas, A. L., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y., and Potts, C. (2011). Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA. Association for Computational Linguistics.

[201] Manakul, P. and Gales, M. J. (2022). Podcast summary assessment: A resource for evaluating summary assessment methods. *arXiv preprint arXiv:2208.13265*.

[202] Manakul, P., Liusie, A., and Gales, M. (2023a). Selfcheckgpt: Zero-resource black-box hallucination detection for generative large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 9004–9017.

[203] Manakul, P., Liusie, A., and Gales, M. J. (2023b). Mqag: Multiple-choice question answering and generation for assessing information consistency in summarization. *arXiv preprint arXiv:2301.12307*.

[204] Manning, C. D. and Schütze, H. (1999). *Foundations of Statistical Natural Language Processing*. MIT Press.

[205] Manning, C. D., Surdeanu, M., Bauer, J., Finkel, J. R., Bethard, S., and McClosky, D. (2014). The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*, pages 55–60.

[206] Manski, C. F. (1977). The structure of random utility models. *Theory and decision*, 8(3):229.

[207] matthias bastian (2023). Gpt-4 has more than a trillion parameters - report. https://the-decoder.com/gpt-4-has-a-trillion-parameters/.

[208] Maudslay, R. H., Gonen, H., Cotterell, R., and Teufel, S. (2019). It's all in the name: Mitigating gender bias with name-based counterfactual data substitution. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5267–5275.

[209] Maurits, L. (2012). *Representation, information theory and basic word order.* PhD thesis, The University of Adelaide.

[210] McCoy, T., Pavlick, E., and Linzen, T. (2019). Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3428–3448.

[211] McCulloch, W. S. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5:115–133.

[212] McKnight, S. W., Civelekoglu, A., Gales, M. J., Bannò, S., Liusie, A., and Knill, K. M. (2023). Automatic assessment of conversational speaking tests. In *Proceedings of 9th Workshop on Speech and Language Technology in Education (SLaTE)*, pages 99–103.

[213] McNamara, D. (2007). *Reading comprehension strategies: Theories, interventions, and technologies*. Lawrence Erlbaum Associates.

[214] Mehrabi, N., Morstatter, F., Saxena, N., Lerman, K., and Galstyan, A. (2021). A survey on bias and fairness in machine learning. *ACM computing surveys (CSUR)*, 54(6):1–35.

[215] Mehri, S. and Eskenazi, M. (2020). USR: An unsupervised and reference free evaluation metric for dialog generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 681–707, Online. Association for Computational Linguistics.

[216] Mesaros, A., Heittola, T., and Virtanen, T. (2016). TUT database for acoustic scene classification and sound event detection. In *2016 24th European Signal Processing Conference (EUSIPCO)*, pages 1128–1132. IEEE.

[217] Mihaylova, T. and Martins, A. F. (2019). Scheduled sampling for transformers. *ACL 2019*, page 351.

[218] Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

[219] Minka, T., Cleven, R., and Zaykov, Y. (2018). Trueskill 2: An improved bayesian skill rating system. *Technical Report*.

[220] Mohamed, A.-r., Dahl, G. E., and Hinton, G. (2011). Acoustic modeling using deep belief networks. *IEEE transactions on audio, speech, and language processing*, 20(1):14–22.

[221] Molenda, P., Liusie, A., and Gales, M. (2024). Waterjudge: Quality-detection trade-off when watermarking large language models. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 3515–3525.

[222] Moskal, B. M. and Leydens, J. A. (2000). Scoring rubrics: What, when and how? *Practical Assessment, Research, and Evaluation*, 7(1):3.

[223] Mullooly, A., Andersen, Ø., Benedetto, L., Buttery, P., Caines, A., Gales, M. J., Karatay, Y., Knill, K., Liusie, A., Raina, V., et al. (2023). The cambridge multiple-choice questions reading dataset.

[224] Murdock Jr, B. B. (1962). The serial position effect of free recall. *Journal of experimental psychology*, 64(5):482.

[225] Nair, V. and Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814.

[226] Nassif, A. B., Shahin, I., Attili, I., Azzeh, M., and Shaalan, K. (2019). Speech recognition using deep neural networks: A systematic review. *IEEE access*, 7:19143–19165.

[227] Naveed, H., Khan, A. U., Qiu, S., Saqib, M., Anwar, S., Usman, M., Akhtar, N., Barnes, N., and Mian, A. (2023). A comprehensive overview of large language models. *arXiv preprint arXiv:2307.06435*.

[228] Newman, M. (2023). Efficient computation of rankings from pairwise comparisons. *Journal of Machine Learning Research*, 24(238):1–25.

[229] Ng, A. Y. (2004). Feature selection, l 1 vs. l 2 regularization, and rotational invariance. In *Proceedings of the twenty-first international conference on Machine learning*, page 78.

[230] Ngai, E. W., Hu, Y., Wong, Y. H., Chen, Y., and Sun, X. (2011). The application of data mining techniques in financial fraud detection: A classification framework and an academic review of literature. *Decision support systems*, 50(3):559–569.

[231] Oba, D., Kaneko, M., and Bollegala, D. (2023). In-contextual bias suppression for large language models. *arXiv preprint arXiv:2309.07251*.

[232] Oehlert, G. W. (1992). A note on the delta method. *The American Statistician*, 46(1):27–29.

[233] Otter, D. W., Medina, J. R., and Kalita, J. K. (2020). A survey of the usages of deep learning for natural language processing. *IEEE transactions on neural networks and learning systems*, 32(2):604–624.

[234] Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., et al. (2022). Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744.

[235] Pan, S. J. and Yang, Q. (2009). A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359.

[236] Panayotov, V., Chen, G., Povey, D., and Khudanpur, S. (2015). Librispeech: an asr corpus based on public domain audio books. In *2015 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 5206–5210. IEEE.

[237] Pang, B. and Lee, L. (2005). Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 115–124, Ann Arbor, Michigan. Association for Computational Linguistics.

[238] Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318.

[239] Park, C., Choi, M., Lee, D., and Choo, J. (2024). Paireval: Open-domain dialogue evaluation with pairwise comparison. *arXiv preprint arXiv:2404.01015*.

[240] Pascanu, R., Mikolov, T., and Bengio, Y. (2013). On the difficulty of training recurrent neural networks. In *International conference on machine learning*, pages 1310–1318. Pmlr.

[241] Patwardhan, N., Marrone, S., and Sansone, C. (2023). Transformers in the real world: A survey on nlp applications. *Information*, 14(4):242.

[242] Pearl, J. (2000). *Causality: Models, Reasoning, and Inference*. Cambridge University Press, 1st edition.

[243] Pearl, J. (2009). *Causality*. Cambridge university press.

[244] Pearson, K. (1895). Note on regression and inheritance in the case of two parents. *Proceedings of the Royal Society of London*, 58:240–242.

[245] Pennington, J., Socher, R., and Manning, C. D. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

[246] Peters, M. E., Ruder, S., and Smith, N. A. (2019). To tune or not to tune? adapting pretrained representations to diverse tasks. *ACL 2019*, page 7.

[247] Pezeshkpour, P. and Hruschka, E. (2024). Large language models sensitivity to the order of options in multiple-choice questions. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 2006–2017.

[248] Phan, H., Koch, P., Katzberg, F., Maass, M., Mazur, R., and Mertins, A. (2017). Audio scene classification with deep recurrent neural networks. In *Proceedings of Interspeech*, pages 3043–3047. International Speech Communication Association.

[249] Piczak, K. J. (2015). Esc: Dataset for environmental sound classification. In *Proceedings of the 23rd ACM international conference on Multimedia*, pages 1015–1018.

[250] Platt, J. et al. (1999). Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in large margin classifiers*, 10(3):61–74.

[251] Pollitt, A. (2012). The method of adaptive comparative judgement. *Assessment in Education: principles, policy & practice*, 19(3):281–300.

[252] Pollitt, A. and Crisp, V. (2004). Could comparative judgements of script quality replace traditional marking and improve the validity of exam questions. In *BERA annual conference, UMIST Manchester, England*.

[253] Pollitt, A. and Murray, N. L. (1996). What raters really pay attention to. *Studies in language testing*, 3:74–91.

[254] Polyak, B. T. (1964). Some methods of speeding up the convergence of iteration methods. *Ussr computational mathematics and mathematical physics*, 4(5):1–17.

[255] Pratap, V., Tjandra, A., Shi, B., Tomasello, P., Babu, A., Kundu, S., Elkahky, A., Ni, Z., Vyas, A., Fazel-Zarandi, M., et al. (2024). Scaling speech technology to 1,000+ languages. *Journal of Machine Learning Research*, 25(97):1–52.

[256] Press, O. and Wolf, L. (2017). Using the output embedding to improve language models. *EACL 2017*, page 157.

[257] Pundak, G., Sainath, T. N., Prabhavalkar, R., Kannan, A., and Zhao, D. (2018). Deep context: end-to-end contextual speech recognition. In *2018 IEEE spoken language technology workshop (SLT)*, pages 418–425. IEEE.

[258] Qian, N. (1999). On the momentum term in gradient descent learning algorithms. *Neural networks*, 12(1):145–151.

[259] Qin, Z., Jagerman, R., Hui, K., Zhuang, H., Wu, J., Shen, J., Liu, T., Liu, J., Metzler, D., Wang, X., et al. (2023). Large language models are effective text rankers with pairwise ranking prompting. *arXiv preprint arXiv:2306.17563*.

[260] Quora, I. (2017). Quora question pairs dataset. https://www.quora.com/q/quoradata/First-Quora-Dataset-Release-Question-Pairs.

[261] Rabiner, L. R. (1989). A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286.

[262] Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al. (2021). Learning transferable visual models from natural language supervision. In *Proc. of the 38th International Conference on Machine Learning*, pages 8748–8763. PMLR.

[263] Radford, A., Kim, J. W., Xu, T., Brockman, G., McLeavey, C., and Sutskever, I. (2023). Robust speech recognition via large-scale weak supervision. In *International conference on machine learning*, pages 28492–28518. PMLR.

[264] Radford, A., Narasimhan, K., Salimans, T., Sutskever, I., et al. (2018). Improving language understanding by generative pre-training.

[265] Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. (2019). Language models are unsupervised multitask learners. Available at: https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf.

[266] Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551.

[267] Raikes, N., Scorey, S., and Shiell, H. (2009). Grading examinations using expert judgements from a diverse pool of judges.

[268] Raina, V. and Gales, M. (2022). Answer uncertainty and unanswerability in multiple-choice machine reading comprehension. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 1020–1034, Dublin, Ireland. Association for Computational Linguistics.

[269] Raina, V., Liusie, A., and Gales, M. (2024). Is llm-as-a-judge robust? investigating universal adversarial attacks on zero-shot llm assessment. *arXiv preprint arXiv:2402.14016*.

[270] Ranzato, M., Chopra, S., Auli, M., and Zaremba, W. (2016). Sequence level training with recurrent neural networks. In *4th International Conference on Learning Representations, ICLR 2016*.

[271] Razeghi, Y., Logan IV, R. L., Gardner, M., and Singh, S. (2022). Impact of pretraining term frequencies on few-shot numerical reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 840–854.

[272] Rei, R., Stewart, C., Farinha, A. C., and Lavie, A. (2020). Comet: A neural framework for mt evaluation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2685–2702.

[273] Robbins, H. and Monro, S. (1951). A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407.

[274] Robinson, J., Rytting, C. M., and Wingate, D. (2023). Leveraging large language models for multiple choice question answering. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

[275] Rogers, A., Gardner, M., and Augenstein, I. (2023). Qa dataset explosion: A taxonomy of nlp resources for question answering and reading comprehension. *ACM Computing Surveys*, 55(10):1–45.

[276] Rogers, A., Kovaleva, O., and Rumshisky, A. (2021). A primer in bertology: What we know about how bert works. *Transactions of the Association for Computational Linguistics*, 8:842–866.

[277] Roller, S., Dinan, E., Goyal, N., Ju, D., Williamson, M., Liu, Y., Xu, J., Ott, M., Smith, E. M., Boureau, Y.-L., and Weston, J. (2021). Recipes for building an open-domain chatbot. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics*, pages 300–325.

[278] Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386.

[279] Ruder, S. (2016). An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*.

[280] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986a). Learning representations by back-propagating errors. *nature*, 323(6088):533–536.

[281] Rumelhart, D. E., McClelland, J. L., Group, P. R., et al. (1986b). *Parallel distributed processing, volume 1: Explorations in the microstructure of cognition: Foundations*. The MIT press.

[282] Salamon, J., Jacoby, C., and Bello, J. P. (2014). A dataset and taxonomy for urban sound research. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 1041–1044.

[283] Sanchez, P., Voisey, J. P., Xia, T., Watson, H. I., O'Neil, A. Q., and Tsaftaris, S. A. (2022). Causal machine learning for healthcare and precision medicine. *Royal Society Open Science*, 9(8):220638.

[284] Sánchez-Rada, J. F. and Iglesias, C. A. (2019). Social context in sentiment analysis: Formal definition, overview of current trends and framework for comparison. *Information Fusion*, 52:344–356.

[285] Saravia, E., Liu, H.-C. T., Huang, Y.-H., Wu, J., and Chen, Y.-S. (2018). Carer: Contextualized affect representations for emotion recognition. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, pages 3687–3697.

[286] Schick, T. and Schütze, H. (2021). Exploiting cloze-questions for few-shot text classification and natural language inference. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 255–269.

[287] Schneider, S., Baevski, A., Collobert, R., and Auli, M. (2019). wav2vec: Unsupervised pre-training for speech recognition. *Interspeech 2019*.

[288] Schramowski, P., Turan, C., Andersen, N., Rothkopf, C. A., and Kersting, K. (2022). Large pre-trained language models contain human-like biases of what is right and wrong to do. *Nature Machine Intelligence*, 4(3):258–268.

[289] Schuller, B., Steidl, S., and Batliner, A. (2009). The interspeech 2009 emotion challenge. In *Proceedings of Interspeech*, pages 312–315.

[290] Schuster, M. and Paliwal, K. K. (1997). Bidirectional recurrent neural networks. *IEEE transactions on Signal Processing*, 45(11):2673–2681.

[291] Schütze, H., Manning, C. D., and Raghavan, P. (2008). *Introduction to information retrieval*, volume 39. Cambridge University Press Cambridge.

[292] Schwarz, N., Bless, H., Strack, F., Klumpp, G., Rittenauer-Schatka, H., and Simons, A. (1991). Ease of retrieval as information: Another look at the availability heuristic. *Journal of Personality and Social Psychology*, 61(2):195.

[293] Sellam, T., Das, D., and Parikh, A. (2020). Bleurt: Learning robust metrics for text generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7881–7892.

[294] Sennrich, R., Haddow, B., and Birch, A. (2016). Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725.

[295] Shannon, C. E. (1948). A mathematical theory of communication. *The Bell System Technical Journal*, 27(3):379–423.

[296] Shi, J. and Norgeot, B. (2022). Learning causal effects from observational data in healthcare: a review and summary. *Frontiers in Medicine*, 9:864882.

[297] Shin, T., Razeghi, Y., Logan IV, R. L., Wallace, E., and Singh, S. (2020). Autoprompt: Eliciting knowledge from language models with automatically generated prompts. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4222–4235.

[298] Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C. D., Ng, A., and Potts, C. (2013). Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.

[299] Spearman, C. (1904). The proof and measurement of association between two things. *The American Journal of Psychology*, 15(1):72–101.

[300] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958.

[301] Steedle, J. T. and Ferrara, S. (2016). Evaluating comparative judgment as an approach to essay scoring. *Applied Measurement in Education*, 29(3):211–223.

[302] Stevens, S. S., Volkmann, J., and Newman, E. B. (1937). A scale for the measurement of the psychological magnitude pitch. *The journal of the acoustical society of america*, 8(3):185–190.

[303] Stöter, F.-R., Chakrabarty, S., Habets, E., and Edler, B. (2018). LibriCount, a dataset for speaker count estimation.

[304] Sturm, B. L. (2013). The GTZAN dataset: Its contents, its faults, their effects on evaluation, and its future use. *arXiv preprint arXiv:1306.1461*.

[305] Sutskever, I., Martens, J., Dahl, G., and Hinton, G. (2013). On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pages 1139–1147. PMLR.

[306] Tang, R., Kong, D., Huang, L., and Xue, H. (2023). Large language models can be lazy learners: Analyze shortcuts in in-context learning. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 4645–4657.

[307] Taori, R., Gulrajani, I., Zhang, T., Dubois, Y., Li, X., Guestrin, C., Liang, P., and Hashimoto, T. B. (2023). Stanford alpaca: A strong, replicable instruction-following model. https://crfm.stanford.edu/2023/03/13/alpaca.html. Accessed: 2024-08-11.

[308] Team, G., Mesnard, T., Hardin, C., Dadashi, R., Bhupatiraju, S., Pathak, S., Sifre, L., Rivière, M., Kale, M. S., Love, J., et al. (2024). Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*.

[309] Thrun, S. and Pratt, L. (1998). Learning to learn: Introduction and overview. In *Learning to learn*, pages 3–17. Springer.

[310] Thurstone, L. L. (1927). Three psychophysical laws. *Psychological Review*, 34(6):424.

[311] Tikhonov, A. N. et al. (1943). On the stability of inverse problems. In *Dokl. akad. nauk sssr*, volume 39, pages 195–198.

[312] Tishby, N., Pereira, F. C., and Bialek, W. (2000). The information bottleneck method. *arXiv preprint physics/0004057*.

[313] Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., et al. (2023a). Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

[314] Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., et al. (2023b). Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

[315] Tversky, A. and Kahneman, D. (1974). Judgment under uncertainty: Heuristics and biases: Biases in judgments reveal some heuristics of thinking under uncertainty. *science*, 185(4157):1124–1131.

[316] Tversky, A. and Kahneman, D. (1981). The framing of decisions and the psychology of choice. *science*, 211(4481):453–458.

[317] Vajjala, S., Majumder, B., Gupta, A., and Surana, H. (2020). *Practical natural language processing: a comprehensive guide to building real-world NLP systems*. O'Reilly Media.

[318] Van Rijsbergen, C. J. (1974). Foundation of evaluation. *Journal of documentation*, 30(4):365–373.

[319] Vapnik, V. (1991). Principles of risk minimization for learning theory. *Advances in neural information processing systems*, 4.

[320] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

[321] Vincent, P., Larochelle, H., Bengio, Y., and Manzagol, P.-A. (2008). Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th International Conference on Machine Learning (ICML)*, pages 1096–1103. ACM.

[322] Viterbi, A. (1967). Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE transactions on Information Theory*, 13(2):260–269.

[323] Voulodimos, A., Doulamis, N., Doulamis, A., and Protopapadakis, E. (2018). Deep learning for computer vision: A brief review. *Computational intelligence and neuroscience*, 2018(1):7068349.

[324] Wang, A., Cho, K., and Lewis, M. (2020). Asking and answering questions to evaluate the factual consistency of summaries. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5008–5020.

[325] Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., and Bowman, S. (2018). Glue: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355.

[326] Wang, J., Liang, Y., Meng, F., Shi, H., Li, Z., Xu, J., Qu, J., and Zhou, J. (2023). Is chatgpt a good nlg evaluator? a preliminary study. *arXiv preprint arXiv:2303.04048*.

[327] Wang, T., Sridhar, R., Yang, D., and Wang, X. (2022a). Identifying and mitigating spurious correlations for improving robustness in NLP models. In Carpuat, M., de Marneffe, M.-C., and Meza Ruiz, I. V., editors, *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 1719–1729, Seattle, United States. Association for Computational Linguistics.

[328] Wang, Y., Mishra, S., Alipoormolabashi, P., Kordi, Y., Mirzaei, A., Arunkumar, A., Ashok, A., Dhanasekaran, A. S., Naik, A., Stap, D., et al. (2022b). Super-naturalinstructions: Generalization via declarative instructions on 1600+ nlp tasks. *arXiv preprint arXiv:2204.07705*.

[329] Wang, Z. and Culotta, A. (2020). Identifying spurious correlations for robust text classification. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 3431–3440.

[330] Wei, J., Bosma, M., Zhao, V., Guu, K., Yu, A. W., Lester, B., Du, N., Dai, A. M., and Le, Q. V. (2021). Finetuned language models are zero-shot learners. In *International Conference on Learning Representations*.

[331] Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., Le, Q. V., Zhou, D., et al. (2022). Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

[332] Weizenbaum, J. (1966). Eliza—a computer program for the study of natural language communication between man and machine. *Communications of the ACM*, 9(1):36–45.

[333] Welling, M. (2007). Product of experts. *Scholarpedia*, 2(10):3879. revision #137078.

[334] Weng, L. (2018). Attention? attention. *Lil'Log, June*, 24.

[335] Werbos, P. J. (1990). Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560.

[336] Williams, A., Nangia, N., and Bowman, S. R. (2018). A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of NAACL-HLT*, pages 1112–1122.

[337] Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8:229–256.

[338] Williams, R. J. and Zipser, D. (1989). A learning algorithm for continually running fully recurrent neural networks. In *Neural Computation*, volume 1, pages 270–280. MIT Press.

[339] Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., et al. (2016). Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

[340] Wu, Z., Jain, P., Wright, M., Mirhoseini, A., Gonzalez, J. E., and Stoica, I. (2021). Representing long-range context for graph neural networks with global attention. *Advances in Neural Information Processing Systems*, 34:13266–13279.

[341] Yan, R. (2018). " chitty-chitty-chat bot": Deep learning for conversational ai. In *IJCAI*, volume 18, pages 5520–5526.

[342] Ye, Q., Zhang, Z., and Law, R. (2009). Sentiment classification of online reviews to travel destinations by supervised machine learning approaches. *Expert systems with applications*, 36(3):6527–6535.

[343] Yu, W., Jiang, Z., Dong, Y., and Feng, J. (2020). Reclor: A reading comprehension dataset requiring logical reasoning. In *International Conference on Learning Representations*.

[344] Yuan, W., Neubig, G., and Liu, P. (2021). Bartscore: Evaluating generated text as text generation. *Advances in Neural Information Processing Systems*, 34:27263–27277.

[345] Zen, H., Gales, M. J., Nankaku, Y., and Tokuda, K. (2011). Product of experts for statistical parametric speech synthesis. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(3):794–805.

[346] Zermelo, E. (1929). Die berechnung der turnier-ergebnisse als ein maximumproblem der wahrscheinlichkeitsrechnung. *Mathematische Zeitschrift*, 29(1):436–460.

[347] Zhang, J., Zhao, Y., Saleh, M., and Liu, P. (2020a). Pegasus: Pre-training with extracted gap-sentences for abstractive summarization. In *International conference on machine learning*, pages 11328–11339. PMLR.

[348] Zhang, S., Dong, L., Li, X., Zhang, S., Sun, X., Wang, S., Li, J., Hu, R., Zhang, T., Wu, F., et al. (2023). Instruction tuning for large language models: A survey. *arXiv preprint arXiv:2308.10792*.

[349] Zhang, T., Kishore, V., Wu, F., Weinberger, K. Q., and Artzi, Y. (2019a). Bertscore: Evaluating text generation with bert. *arXiv preprint arXiv:1904.09675*.

[350] Zhang, X., Zhao, J., and LeCun, Y. (2015). Character-level convolutional networks for text classification. *Advances in neural information processing systems*, 28.

[351] Zhang, Y., Baldridge, J., and He, L. (2019b). PAWS: Paraphrase Adversaries from Word Scrambling. In *Proc. of NAACL*.

[352] Zhang, Y., Sun, S., Galley, M., Chen, Y.-C., Brockett, C., Gao, X., Gao, J., Liu, J., and Dolan, B. (2020b). Dialogpt: Large-scale generative pre-training for conversational response generation. In *ACL, system demonstration*.

[353] Zhao, D., Sainath, T. N., Rybach, D., Rondon, P., Bhatia, D., Li, B., and Pang, R. (2019a). Shallow-fusion end-to-end contextual biasing. In *Interspeech*, pages 1418–1422.

[354] Zhao, W., Peyrard, M., Liu, F., Gao, Y., Meyer, C. M., and Eger, S. (2019b). Moverscore: Text generation evaluating with contextualized embeddings and earth mover distance. *arXiv preprint arXiv:1909.02622*.

[355] Zhao, Z., Wallace, E., Feng, S., Klein, D., and Singh, S. (2021). Calibrate before use: Improving few-shot performance of language models. In *International Conference on Machine Learning*, pages 12697–12706. PMLR.

[356] Zheng, C., Zhou, H., Meng, F., Zhou, J., and Huang, M. (2023). Large language models are not robust multiple choice selectors. *arXiv e-prints*, pages arXiv–2309.

[357] Zheng, L., Chiang, W.-L., Sheng, Y., Zhuang, S., Wu, Z., Zhuang, Y., Lin, Z., Li, Z., Li, D., Xing, E., et al. (2024). Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems*, 36.

[358] Zhong, M., Liu, Y., Yin, D., Mao, Y., Jiao, Y., Liu, P., Zhu, C., Ji, H., and Han, J. (2022). Towards a unified multi-dimensional evaluator for text generation. In Goldberg, Y., Kozareva, Z., and Zhang, Y., editors, *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 2023–2038, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

[359] Zhou, C., Sun, C., Liu, Z., and Lau, F. (2015). A c-lstm neural network for text classification. *arXiv preprint arXiv:1511.08630*.

[360] Zhou, Y., Muresanu, A. I., Han, Z., Paster, K., Pitis, S., Chan, H., and Ba, J. (2022). Large language models are human-level prompt engineers. In *NeurIPS 2022 Foundation Models for Decision Making Workshop*.

[361] Zhou, Y., Xu, P., Liu, X., An, B., Ai, W., and Huang, F. (2024). Explore spurious correlations at the concept level in language models for text classification. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 478–492, Bangkok, Thailand. Association for Computational Linguistics.

[362] Zhu, Y., Wan, J., Zhou, Z., Chen, L., Qiu, L., Zhang, W., Jiang, X., and Yu, Y. (2019). Triple-to-text: Converting rdf triples into high-quality natural languages via optimizing an inverse kl divergence. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 455–464.

Appendix A

A.1 Product of Experts

A.1.1 Form of the Gaussiam PoE Score Distribution

To determine $\mathtt{p}(\mathbf{s}|\mathcal{C}_{1:K})$ given $\mathtt{P}(\tilde{\mathbf{W}}\mathbf{s}|\mathcal{C}_{1:K}) = \mathcal{N}\left(\tilde{\mathbf{W}}\mathbf{s};\tilde{\boldsymbol{\mu}},\tilde{\boldsymbol{\Sigma}}\right)$, one can expand the expression and isolate all terms that have an $\mathbf{s}$, yielding,

$$\mathtt{p}(\tilde{\mathbf{W}}\mathbf{s}|\mathcal{C}_{1:K}) = \mathcal{N}\left(\tilde{\mathbf{W}}\mathbf{s};\tilde{\boldsymbol{\mu}},\tilde{\boldsymbol{\Sigma}}\right) \tag{A.1}$$

$$\propto \exp\left(-\frac{1}{2}\left(\tilde{\mathbf{W}}\mathbf{s}-\tilde{\boldsymbol{\mu}}\right)^{\mathsf{T}}\tilde{\boldsymbol{\Sigma}}^{-1}\left(\tilde{\mathbf{W}}\mathbf{s}-\tilde{\boldsymbol{\mu}}\right)\right) \tag{A.2}$$

$$\propto \exp\left(-\frac{1}{2}\left(\mathbf{s}^{\mathsf{T}}\tilde{\mathbf{W}}^{\mathsf{T}}\tilde{\boldsymbol{\Sigma}}^{-1}\tilde{\mathbf{W}}\mathbf{s}+2\mathbf{s}^{\mathsf{T}}\tilde{\mathbf{W}}^{\mathsf{T}}\tilde{\boldsymbol{\Sigma}}^{-1}\tilde{\boldsymbol{\mu}}\right)\right) \tag{A.3}$$

As the distribution over scores will be Gaussian, $\mathtt{p}(\mathbf{s}|C_{1:K}) = \mathcal{N}(\mathbf{s};\boldsymbol{\mu}^*,\boldsymbol{\Sigma}^*)$, one can equate coefficients to derive the form used in the paper,

$$\mathtt{p}(\mathbf{s}|\mathcal{C}_{1:K}) = \mathcal{N}\left(\mathbf{s};(\tilde{\mathbf{W}}^{\mathsf{T}}\tilde{\boldsymbol{\Sigma}}^{-1}\tilde{\mathbf{W}})^{-1}\tilde{\mathbf{W}}^{\mathsf{T}}\tilde{\boldsymbol{\Sigma}}^{-1}\tilde{\boldsymbol{\mu}}\ ,\ (\tilde{\mathbf{W}}^{\mathsf{T}}\tilde{\boldsymbol{\Sigma}}^{-1}\tilde{\mathbf{W}})^{-1}\right) \tag{A.4}$$

Therefore the maximum probability scores will occur at $\hat{\mathbf{s}} = (\tilde{\mathbf{W}}^{\mathsf{T}}\tilde{\boldsymbol{\Sigma}}^{-1}\tilde{\mathbf{W}})^{-1}\tilde{\mathbf{W}}^{\mathsf{T}}\tilde{\boldsymbol{\Sigma}}^{-1}\tilde{\boldsymbol{\mu}}$, with a probability of,

$$\mathtt{p}(\hat{\mathbf{s}}|\mathcal{C}_{1:K}) = \frac{1}{(2\pi)^{N/2}\mathtt{det}\left((\tilde{\mathbf{W}}^{\mathsf{T}}\tilde{\boldsymbol{\Sigma}}^{-1}\tilde{\mathbf{W}})^{-1}\right)^{1/2}} \tag{A.5}$$

$$= \frac{\sqrt{\mathtt{det}(\tilde{\mathbf{W}}^{\mathsf{T}}\tilde{\boldsymbol{\Sigma}}^{-1}\tilde{\mathbf{W}})}}{(2\pi)^{N/2}} \tag{A.6}$$

For the linear Gaussian, where it is assumed that $\tilde{\mathbf{\Sigma}} = \sigma^2\mathbf{I}$, this can be reduced to,

$$\mathtt{p}(\hat{\mathbf{s}}|\mathcal{C}_{1:K}) = \frac{\sqrt{\mathtt{det}(\tilde{\mathbf{W}}^\mathsf{T}\tilde{\mathbf{W}})}}{(2\pi\sigma^2)^{N/2}} \tag{A.7}$$

A.1.2 Equivalence of Solution with Average Probability

The matrix $(\tilde{\mathbf{W}}^\mathsf{T}\tilde{\mathbf{W}})$ has an interesting structure that is tightly related with the comparisons. Let $\mathbf{A} = \tilde{\mathbf{W}}^\mathsf{T}\tilde{\mathbf{W}}$; for any set of selected comparisons, $a_{ij} = \tilde{\mathbf{W}}_i \cdot \tilde{\mathbf{W}}_j$. By noting the sparse structure of each row $\tilde{\mathbf{W}}_i$, the diagonal elements of $\mathbf{A}$ can be shown to represent the number of comparisons the element has been involved in, while the off-diagonal elements represent the number of times each comparison was made,

$$a_{kk} = \mathbb{1}(k=1) + \sum_{i,j\in\mathcal{C}} \left[\mathbb{1}(i=k) + \mathbb{1}(j=k)\right] \tag{A.8}$$

$$a_{km} = -1 \cdot \sum_{i,j\in\mathcal{C}} \left[\mathbb{1}(i=k)\mathbb{1}(j=m) + \mathbb{1}(i=m)\mathbb{1}(j=k)\right] \tag{A.9}$$

when considering the full comparison matrix, irrespective of N, the matrix $\tilde{\mathbf{W}}^\mathsf{T}\tilde{\mathbf{W}}$ will have the form,

$$\tilde{\mathbf{W}}^\mathsf{T}\tilde{\mathbf{W}} = \begin{bmatrix} N & -1 & -1 & \dots & -1 \\ -1 & N-1 & -1 & \dots & -1 \\ -1 & -1 & N-1 & \dots & -1 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -1 & -1 & -1 & \dots & N-1 \end{bmatrix} \tag{A.10}$$

and it can be shown that the inverse therefore has form,

$$\left(\tilde{\mathbf{W}}^\mathsf{T}\tilde{\mathbf{W}}\right)^{-1} = \begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & 1+\frac{2}{N} & 1+\frac{1}{N} & \dots & 1+\frac{1}{N} \\ 1 & 1+\frac{1}{N} & 1+\frac{2}{N} & \dots & 1+\frac{1}{N} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & 1+\frac{1}{N} & 1+\frac{1}{N} & \dots & 1+\frac{2}{N} \end{bmatrix} \tag{A.11}$$

$$= \frac{N+1}{N}\begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & 1 & 1 & \dots & 1 \\ 1 & 1 & 1 & \dots & 1 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & 1 & \dots & 1 \end{bmatrix} + \frac{1}{2N}\begin{bmatrix} -1 & -1 & -1 & \dots & -1 \\ -1 & 1 & 0 & \dots & 0 \\ -1 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -1 & 0 & 0 & \dots & 1 \end{bmatrix} \tag{A.12}$$

We previously demonstrated that linear Gaussian experts yield a solution of the form,

$$\hat{\mathbf{s}} = \alpha \cdot (\tilde{\mathbf{W}}^{\mathsf{T}}\tilde{\mathbf{W}})^{-1}\tilde{\mathbf{W}}^{\mathsf{T}}\tilde{\boldsymbol{\mu}} \tag{A.13}$$

Here, $\tilde{\boldsymbol{\mu}}$ represents the shifted LLM probabilities for each comparative decision. As a result, one can observe that $\tilde{\mathbf{W}}^{\mathsf{T}}\tilde{\boldsymbol{\mu}}$ corresponds to the sum of probabilities for all comparisons involving each element (plus a constant). Hence, when the full set of comparisons is used, the maximum probability solution becomes equivalent to the average probability solution. This equivalence seems reasonable as when all possible comparisons are made, each text is compared against every other text, resulting in the same set of opponents for each item. However when a subset of comparisons are used, the strength of opponents may vary and the PoE solution (which takes opponent strength into account) may provide a solution that diverges from the average probability.

A.1.3 Efficient Greedy Comparison Selection

It was shown that given $\tilde{\mathbf{W}}^{(k)}$ and $\mathbf{A}^{(k)} = (\tilde{\mathbf{W}}^{(k)*\mathsf{T}}\tilde{\mathbf{W}}^{(k)})^{-1}$, the next optimal comparison $(\hat{i},\hat{j})$ can be calculated as,

$$\hat{i},\hat{j} = \arg\max_{i,j} \mathbf{A}^{(k)}_{ii} + \mathbf{A}^{(k)}_{jj} - 2\cdot\mathbf{A}^{(k)}_{ij} \tag{A.14}$$

Updating $\tilde{\mathbf{W}}^{(k)}$ is trivial, since considering an additional comparison (i,j) is equivalent to adding an extra row $\mathbf{r} \in R^N$ to $\tilde{\mathbf{W}}^{(k)}$, where $\mathbf{r}_i = 1$, $\mathbf{r}_j = -1$ and $\mathbf{r}_l = 0 \;\; \forall l \neq i,j$. Therefore

$$\tilde{\mathbf{W}}^{(k+1)} = [\tilde{\mathbf{W}}^{(k)};\mathbf{r}] \tag{A.15}$$

However, one can also efficiently update the inverse using the Sherman-Morrison inversion lemma,

$$\mathbf{A}^{(k+1)} = \left([\tilde{\mathbf{W}}^{(k)};\mathbf{r}]^{\mathsf{T}}[\tilde{\mathbf{W}}^{(k)};\mathbf{r}]\right)^{-1} \tag{A.16}$$

$$= \left(\tilde{\mathbf{W}}^{(k)\mathsf{T}}\tilde{\mathbf{W}}^{(k)} + \mathbf{r}\mathbf{r}^{\mathsf{T}}\right)^{-1} \tag{A.17}$$

$$= \mathbf{A}^{(k)} - \frac{\mathbf{A}^{(k)}\mathbf{r}\mathbf{r}^{\mathsf{T}}\mathbf{A}^{(k)}}{1+\mathbf{r}^{\mathsf{T}}\mathbf{A}^{(k)}\mathbf{r}} \tag{A.18}$$

Note that to initialize $\tilde{\mathbf{W}}$, the simplest option would be to use $N-1$ comparisons and follow a stripped diagonal matrix, e.g.

$$\tilde{\mathbf{W}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 1 & -1 & 0 \\ 0 & 0 & 1 & -1 \end{bmatrix} \tag{A.19}$$

A.1.4 Derivation of Bias Parameter

A simple approach is to introduce a bias parameter γ that shifts the experts such that, $\mathtt{p}_\gamma(s_i - s_j|p_{ij}) = \mathtt{p}(s_i - s_j - \gamma|p_{ij})$. The value of γ can be determined by noting that the expected score difference between two randomly sampled texts is zero, $\mathbb{E}[s_i - s_j] = 0$,

$$\mathbb{E}[s_i - s_j] = \int_{-\infty}^{\infty} (s_i - s_j)\mathtt{p}(s_i - s_j)d(s_i - s_j) \tag{A.20}$$

$$= \int_0^1 \int_{-\infty}^{\infty} (s_i - s_j)\mathtt{p}_\gamma(s_i - s_j|p_{ij})\mathtt{p}(p_{ij})d(s_i - s_j)dp_{ij} \tag{A.21}$$

$$= \int_0^1 \mathtt{p}(p_{ij}) \left(\int_{-\infty}^{\infty} (s_i - s_j)\mathtt{p}_\gamma(s_i - s_j|p_{ij})d(s_i - s_j) \right) dp_{ij} \tag{A.22}$$

$$= \int_0^1 \mathtt{p}(p_{ij}) \cdot \mathbb{E}[s_i - s_j|p_{ij}, \gamma\,]\, dp_{ij} \tag{A.23}$$

Where $\mathtt{p}(p_{ij})$ denotes the distribution of the LLM comparative probability values from the LLM judge. For the linear Gaussian, $\mathcal{N}\big(s_i - s_j; \alpha\cdot(p_{ij} - \beta), \sigma^2\big)$, setting γ is equivalent to setting the β parameter. The mean of the expert is $f_\mu(p_{ij}) = \alpha\cdot(p_{ij} - \beta)$, and therefore,

$$\mathbb{E}[s_i - s_j] = \int_0^1 \mathtt{p}(p_{ij}) \cdot \mathbb{E}[s_i - s_j|p_{ij}, \gamma\,]\, dp_{ij} \tag{A.24}$$

$$= \int_0^1 \mathtt{p}(p_{ij}) \cdot \alpha\cdot(p_{ij} - \beta)\, dp_{ij} \tag{A.25}$$

$$= \alpha \left(\left(\int_0^1 p_{ij}\, \mathtt{p}(p_{ij})\, dp_{ij} \right) - \beta \right) \tag{A.26}$$

Which setting to zero yields $\beta = \mathbb{E}[p_{ij}] \approx \frac{1}{K}\sum_{k=1}^{K} p_{ij}^{(k)}$, i.e. β should be set to the average LLM comparative probability. For the Bradley-Terry model, it can be shown that $f_\mu(p_{ij}) = -\pi\cdot\cot(\pi p_{ij})$. This value tends to infinity when p_{ij} approaches either 0 or 1, which will yield unstable estimates for γ. Therefore, for experts that are unstable, or for which the expectation is analytically intractable, one can instead ensure the mode of the skill difference likelihood

is set to 0 when the skill difference is 0. Differentiating the expected score difference yields,

$$\frac{\partial}{\partial \gamma}\mathbb{E}[\log \mathrm{p}_\gamma(s_i - s_j)] \tag{A.27}$$

$$= \frac{\partial}{\partial \gamma}\int_0^1 \log \mathrm{p}_\gamma(s_i - s_j|p_{ij})\mathrm{p}(p_{ij})dp_{ij} \tag{A.28}$$

$$= \int_0^1 \mathrm{p}(p_{ij})\frac{\partial}{\partial \gamma}\Big(\log \mathrm{p}_\gamma(s_i - s_j|p_{ij})\Big)dp_{ij} \tag{A.29}$$

The probabilistic Bradley-Terry accounting for bias has form,

$$\mathrm{p}_\gamma(s_i - s_j|p_{ij}) = \frac{1}{Z_{ij}} \cdot \frac{e^{p_{ij}\cdot(s_i - s_j - \gamma)}}{1 + e^{(s_i - s_j - \gamma)}} \tag{A.30}$$

which when differentiated yields,

$$\frac{\partial}{\partial \gamma}\log \mathrm{p}(s_i - s_j|p_{ij}) \tag{A.31}$$

$$= \frac{\partial}{\partial \gamma}\big(p_{ij}\cdot(s_i - s_j - \gamma) - \log(1 + e^{s_i - s_j - \gamma})\big) \tag{A.32}$$

$$= -p_{ij} + \frac{e^{s_i - s_j - \gamma}}{1 + e^{s_i - s_j - \gamma}} \tag{A.33}$$

Evaluating the integral at $s_i - s_j = 0$,

$$\frac{\partial}{\partial \gamma}\mathbb{E}[\log \mathrm{p}_\gamma(s_i - s_j)]\Big|_{s_i - s_j = 0} \tag{A.34}$$

$$= \int_0^1 \mathrm{p}_{\mathrm{LM}}(p_{ij})\left(-p_{ij} + \frac{e^{-\gamma}}{1 + e^{-\gamma}}\right)dp_{ij} \tag{A.35}$$

setting to zero yields, $\gamma = -1\cdot\log\left(\frac{\mathbb{E}[p_{ij}]}{1+\mathbb{E}[p_{ij}]}\right) = -\mathrm{logit}(\mathbb{E}[p_{ij}]) \approx \mathrm{logit}\left(\frac{1}{K}\sum_{k=1}^K p_{ij}^{(k)}\right)$.