# LANGUAGE MODEL ADAPTATION USING MIXTURES AND AN EXPONENTIALLY DECAYING CACHE

*P.R. Clarkson* *A.J. Robinson*

Cambridge University Engineering Department,
Trumpington Street, Cambridge, CB2 1PZ, UK.
{prc14,ajr}@eng.cam.ac.uk

## ABSTRACT

This paper presents two techniques for language model adaptation. The first is based on the use of mixtures of language models: the training text is partitioned according to topic, a language model is constructed for each component, and at recognition time appropriate weightings are assigned to each component to model the observed style of language. The second technique is based on augmenting the standard trigram model with a cache component in which words recurrence probabilities decay exponentially over time. Both techniques yield a significant reduction in perplexity over the baseline trigram language model when faced with multi-domain test text, the mixture-based model giving a 24% reduction and the cache-based model giving a 14% reduction. The two techniques attack the problem of adaptation at different scales, and as a result can be used in parallel to give a total perplexity reduction of 30%.

## 1. INTRODUCTION

In constructing a language model intended for general text, one is faced with the following problem. One can either generate a model which is trained on material from a specific domain, with the result that the model's performance will be good for text from the same domain, but poor for more general text, or one can train the model on text from many diverse sources, which will perform better on general text, but will not be especially well suited for any particular domain.

Clearly, the ideal would be a general language model whose parameters could be automatically tuned according to the style of text it is attempting to model.

Two methods of achieving this will be presented here. The first is a mixture-based approach. The training text is partitioned according to the style of text, and a trigram language model is constructed for each component[1]. Each component is assigned a weighting according to its performance at modelling the observed text, and a final language model is constructed as the weighted sum of each of the mixture components.

The second approach is based on a cache of recent words. Previous work has shown that words that have occurred recently have a higher probability of occurring in future than would be predicted by a static trigram language model [6, 7]. This work investigates the hypothesis that more recent words should be considered more significant within the cache by implementing a cache in which a word's contribution to its recurrence probability decays exponentially with its distance from the current word.

The corpus used for this work was the British National Corpus (BNC) [1]. This contains 4142 different "texts", each from a different source, and covering diverse topics and styles of language. The corpus comprises 100 million words of British English, 10% of which is transcribed speech, and the remainder consists of written language.

From this corpus, 90% (from both the spoken and written components) was used for the purposes of constructing the language models, the rest being reserved as test data. The held-out 10% was taken from random points throughout the corpus.

[1] In this paper a trigram language model refers to a back-off trigram language model with a vocabulary of 65,000 words, employing Turing-Good discounting [5], and with both bigram and trigram cut-offs set to 1.

## 2. MIXTURE-BASED LANGUAGE MODELS

### 2.1. Framework

For this model, general language model training data is partitioned according to topic, and a trigram language model is constructed for each component. Weightings are assigned to each language model according to its performance at modelling the previously seen text, and the word probabilities used by the final model are then simply a linear combination of those from the smaller language models, i.e.

$$P(w_i \mid w_{i-2}, w_{i-1}) = \sum_{j=1}^{k} \lambda_j P(w_i \mid w_{i-2}, w_{i-1}, M_j) \tag{1}$$

where $k$ is the number of mixtures, $\lambda_j$ is the weighting assigned to model $j$, $\sum_{j=1}^{k} \lambda_j = 1$, and $M_j$ represents the parameters of model $j$.

While splitting the training text into smaller sections increases the problems of data sparsity that plague all work in language modelling, the results presented here show that the benefits of having a language model which more closely matches the target domain outweigh this problem. Furthermore, by including the "full" language model (i.e. one trained on the full set of training data) as an additional mixture component, we can reduce the problems caused by data sparsity.

### 2.2. Partitioning the training data

The training data is partitioned such that each text within the training set is assigned to a cluster. A simple $k$-means style clustering algorithm is used. The "distance" between a text and a cluster is defined as being the perplexity of a language model constructed from the text within the cluster with respect to the text. Ideally this would be based on a trigram language model in order to maintain consistency with the language models being constructed. Unigram language models were used here, however, to keep the computational requirements to a practical level.

The BNC assigns a tag to each piece of text in the written component indicating its subject. There are nine tags, ranging from "Imaginative" to "Pure and Natural Science". By partitioning the written component of the corpus according to these tags, and by considering the spoken component as an additional component, we have a manually generated partition of ten components that can be compared with the automatically generated ones.

### 2.3. Assigning weights to mixtures

Given a (totally or partially correct) transcription of the *previously seen* section of the text to which we are aiming to tune our language model, we can compute the probability of each word in this section of the text according to the language model of each component, thus generating a probability stream for each language model. From these probability streams we can use an Expectation-Maximisation (EM) algorithm [4] to generate a set of weights (the $\lambda_j$s in (1)) which are optimal for the previously seen section of text. It is to be hoped that this set of weights will be similarly good for upcoming text.

Ideally, the interpolation weights should be updated after every word, but the process of calculating weightings is too computationally expensive for this to be practical, so we instead update the weights after every 10% of each text in the test set.

### 2.4. Results

The results given in Table 1 show the perplexities for mixture-based language models based on 5, 10, 20, 30, 40 and 50 clusters. Note that the automatically generated partition into 10 clusters performs better that the manually generated BNC clustering, and that the inclusion of the "full" language model gives a greater gain as the number of mixtures increases, and the individual models become more under-trained.

The best performance was achieved using 50 mixtures plus the full language model, when there was a decrease in perplexity of 24.0%.[2]

| Number of clusters | Perplexity | |
|---|---|---|
| | With full LM | W/out full LM |
| 0 (Baseline) | 165.52 | - |
| 5 | 141.17 | 143.23 |
| 10 | 134.30 | 138.24 |
| 10 (BNC) | 139.24 | 142.87 |
| 20 | 129.09 | 134.42 |
| 30 | 126.77 | 134.14 |
| 40 | 127.52 | 135.48 |
| 50 | 125.78 | 134.91 |

Table 1. The perplexities of mixture-based language models

## 3. CACHE BASED MODELS

### 3.1. Framework

It is a commonly observed phenomenon that words which have occurred recently in a piece of text have a higher probability of re-occurring [6]. This is the motivation behind cache-based language models. We store a cache of recently occurring words, and boost their probabilities within the language model:

$$P(w_i \mid w_1, w_2, \ldots w_{i-1}) = \mu P_{\text{cache}}(w_i \mid w_1, w_2, \ldots w_{i-1}) + (1-\mu) P_{\text{trigram}}(w_i \mid w_{i-2}, w_{i-1}) \quad (2)$$

Much previous work on cache-based language models has defined

$$P_{\text{cache}}(w_i \mid w_{i-K}, w_{i-K+1}, \ldots w_{i-1}) = \frac{\text{Number of occurrences of } w_i \text{ in cache}}{K} \quad (3)$$

where $K$ is the size of the cache.

This approach, however, has some clear drawbacks. It seems implausible, for example, that a word's importance within the cache is independent of its distance from the current word; one would expect that more recent words should contribute more to the cache probability.

A cache component was investigated in which the importance of each position in the cache decays exponentially with distance from the current word, i.e.

$$P_{\text{cache}}(w_i \mid w_1, w_2, \ldots, w_{i-1}) = \beta \sum_{j=1}^{i-1} I_{\{w_i = w_j\}} e^{-\alpha(i-j)} \quad (4)$$

where $I$ is an indicator function such that $I_A = 1$ if $A$ is true, and 0 otherwise, $\alpha$ is the decay rate and $\beta$ is a normalising constant[3].

Both "regular" (equation (3)) and "decaying" (equation (4)) cache-based language models were investigated. In both cases the interpolation weights (i.e. the values of $\mu$ in equation (2)) were calculated in exactly the same way as those for the mixture-based language models (Section 2.3).

[2] The amount of memory and CPU time required by these language models increases exponentially with the number of mixtures. It has so far proved impossible to build language models based on more than 50 mixtures.

[3] The size of the cache is no longer an issue; words which occurred many words ago make such a small contribution to the cache probabilities that they can essentially be ignored. We can thus consider the cache size as infinite.

That is, the EM algorithm was used to calculate the optimal choice for $\mu$ based on the *previously seen* section of text, and the weights were updated after every 10% of each text.

### 3.2. Results

Table 2 shows the perplexities of regular cache-based language models with various cache sizes, and Figure 1 shows the perplexities of decaying cache-based language models with various decay rates. It can be seen that the regular cache performs best with a cache size of 500, and that the decaying cache performs better than the standard cache, with the best performance being given by a decay rate of 0.005, when the perplexity is 141.75. This represents a 14.4% reduction from the baseline.

| Size of Cache | Perplexity |
|---|---|
| 50 | 152.59 |
| 100 | 148.33 |
| 200 | 145.83 |
| 500 | 144.73 |
| 1000 | 144.98 |
| 2000 | 145.56 |

Table 2. The perplexities of regular cache-based models

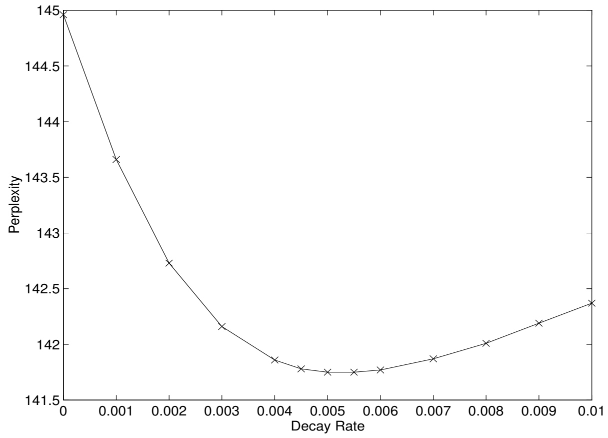


Figure 1. The effect of decay rate on the perplexity of decaying cache-based language models

## 4. COMBINING TECHNIQUES

The cache-based model attempts to model the short-term fluctuations in word probabilities by boosting the probabilities of a small set of words, whereas the mixture-based approach seeks to tune the entire language model toward the general topic of discourse. Because these two techniques tackle the problem of adaptation at different scales, they may be usefully combined to give an even greater reduction in perplexity.

A language model was constructed based on a combination of the two techniques presented here. A mixture-based language model based on 30 mixtures (plus the "full" language model) was combined with a decaying cache component with a decay rate of 0.005. That is

$$P(w_i \mid w_1, w_2, \ldots, w_{i-1}) = \nu P_{\text{mixture}}(w_i \mid w_{i-2}, w_{i-1}) + (1-\nu) P_{\text{cache}}(w_i \mid w_1, w_2, \ldots, w_{i-1}) \quad (5)$$

The interpolation weight $\nu$ was, as before, calculated by EM to minimise the perplexity of the previously seen text, and updated after every 10% of each text.

The resulting language model had a perplexity of 115.86, which represents a decrease of 30.0% over the baseline language model.

## 5. RESULTS BASED ON IMPERFECT TRANSCRIPTION

The results given in Section 2.4 are based on mixture weightings which are calculated using a perfect transcription of the observed text, and those in Section 3.2 are based on the assumption that every word in the cache is correct. Clearly, if these language models were used in an automatic speech recognition (ASR) system, then a perfect transcription would not be available. An experiment was conducted in which the text used to compute the mixture weightings and the cache-based probabilities had been "corrupted" by substituting a proportion of words with words chosen at random from the vocabulary.

Figure 2 shows the effect of the degree of corruption (i.e. the proportion of words which have been replaced with a random selection from the vocabulary) on the performance of both mixture- and cache-based language models. Two mixture-based language models were used, both based on 30 mixture components, one included the "full" model as an additional component, the other did not. The cache-based model was based on a decaying cache with a decay rate of 0.005.

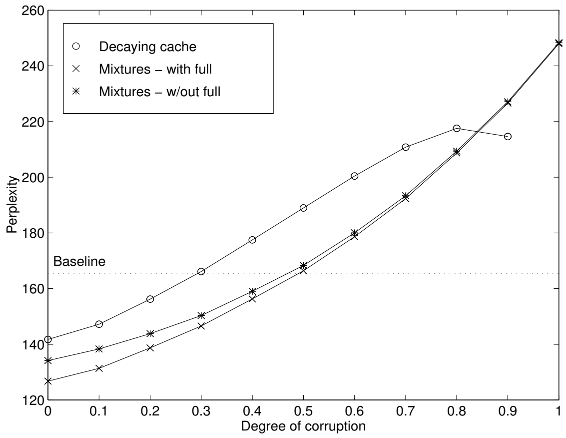


Figure 2. The effect of transcription error rate on perplexity

It can be seen that the cache-based model performs better than the baseline trigram model providing that no more than 30% of the words in the hypothesis are incorrect, and that the mixture-based models yield an improvement providing that the error rate is lower than 50%. The addition of the "full" component to the mixture-based language model becomes less useful as the hypothesis becomes more corrupted because the "full" language model is no better (in fact worse) at modelling random data than the smaller language models, and so its weighting becomes very close to zero.

## 6. CONCLUSIONS AND FUTURE WORK

This paper has presented two techniques for language model adaptation, both of which result in a significant reduction in perplexity over the standard trigram model. In addition, we have shown that the techniques can continue to be useful when the hypothesised transcription which forms the basis for adaptation has been somewhat corrupted. The hope is that if these techniques were used in a speech recognition system, they would result in a useful reduction in word error rate. Future work should therefore be focussed on incorporating these adaptive language models in an ASR system.

It is unlikely to be practical, however, to require a speech decoder to access many individual language models, because of the large amount of memory required to store each of them. Therefore a one-pass approach to recognition using a mixture-based language model is out of the question and we must instead evaluate these language models using lattice (or $N$-best) re-scoring. This has advantages and disadvantages. In many ways these adaptation techniques are ideally suited to the sort of two-pass approach used in lattice re-scoring. In the case of mixture-based models, the mixture weights can be based on a complete transcription right from the start, as opposed to the incomplete transcription used here in the perplexity calculations. Similarly, the cache need not be based solely on words in the past, but can use information from future sentences too. The disadvantage, however, is that lattice re-scoring must take place on a short-scale (essentially sentence-by-sentence) basis if the lattices are to be kept to a reasonable size. The mixture weights and cache-based probabilities for each sentence must be based on the transcriptions of the other sentences, and it is necessary to assume at each stage that these are correct. If this assumption is particularly inaccurate then these adaptation techniques will be of little use. It is therefore difficult to assess the usefulness of a language model which uses long-term context from its perplexity alone.

## ACKNOWLEDGEMENTS

P.R. Clarkson is supported by an EPSRC advanced studentship. This work was supported in part by ESPRIT project 20077 - SPRACH.

## REFERENCES

[1] L. Burnard (editor). *Users Reference Guide for the British National Corpus.* Oxford University Computing Services, May 1995.

[2] D. Carter. Improving Language Models by Clustering Training Sentences. Technical report, SRI International, 1994.

[3] R. Iyer, M. Ostendorf, and J.R. Rohlicek. Language Modeling with Sentence-Level Mixtures. In *Proceedings of the ARPA Workshop on Human Language Technology*, 1994.

[4] F. Jelinek. Self-Organized Language Models for Speech Recognition. In A. Waibel and K.-F Lee, editors, *Readings in Speech Recognition*, pages 450–506. Morgan Kaufman Publishers, 1990.

[5] S.M. Katz. Estimation of Probabilities from Sparse Data for the Language Model Component of a Speech Recognizer. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 35(3):400–401, 1987.

[6] R. Kuhn and R. De Mori. A Cache-Based Natural Language Model for Speech Reproduction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(6):570–583, 1990.

[7] R. Kuhn and R. De Mori. Corrections to 'A Cache-Based Natural Language Model for Speech Reproduction'. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14:691–692, 1992.

[8] R. Rosenfeld. The CMU Statistical Language Modeling Toolkit, and its use in the 1994 ARPA CSR Evaluation. In *ARPA Spoken Language Technology Workshop, Austin, TX*, January 1995.