Adaptive Statistical Class-based Language Modelling

Gareth Lewis Moore

Downing College

October 2001

This dissertation is submitted to the University of Cambridge for the degree of Doctor of Philosophy

Summary

For automatic speech recognition to be effective, a language model is required which can assign a probability to any postulated word sequence. Traditionally such models are based on word *n*-grams, and therefore rely on strictly local constraints. The work in this dissertation is focussed on adaptation techniques which improve the modelling capabilities of existing language models. In this work, the adaptation is applied to class-based *n*-gram models which are then combined with word-based models. Adaptation to both changing topic and local word usage is considered, with both methods allowing adaptation to be performed using only small amounts of additional parameters relative to a standard class model.

The first new technique aims to exploit a notion of current topic, which is first defined by examining various methods of automatic topic corpora construction. Articles are defined on the basis of existing news story boundaries and then clustered agglomeratively using an inverse document frequency similarity metric, obtaining coherent topics suitable for *n*-gram modelling. By changing the likelihood of words within classes on the basis of estimates derived from these topics, significant improvements in model performance are obtained with only minimal additional storage space per topic. By rescoring lattices built for the 1997 CU-HTK HUB-4 evaluation system, a combined word and class 4-gram model baseline word error rate of 17.1% is reduced to 16.9%.

In the second approach a dynamic class membership scheme is proposed whereby words are moved between classes on the basis of the most recently seen word of context. A word pair exchange clusterer is described which optimises the model's pair class map via iterative improvement from an existing standard map, and the effect of different iteration restart points is examined. Results show that significant gains are obtained in bigram experiments, with pair models able to obtain a perplexity of 184 against that of a larger word model's 194 on a broadcast news test set, whilst HUB-4 lattice rescoring reduces a combined word and class bigram model baseline of 21.1% to 20.5%. Gains are reduced significantly as *n*-gram order increases, however, and no HUB-4 system 4-gram baseline reduction was obtained.

Overall it is concluded that unsupervised class model adaptation on a large general corpus can lead to not only perplexity but also recognition gains. Preliminary results also show that the two models can be combined for further improvement.

Acknowledgements

Thanks are due to my supervisor, Steve Young, for his guidance and assistance, and also to my departmental colleagues Konrad Scheffler, Gavin Smith, Ed Whittaker and in particular Gunnar Evermann for their many useful comments, help and advice. Thanks are also due to Patrick Gosling and Phil Woodland for assistance with various aspects of the group's computer system and the existing tools which were employed as part of the work described in this thesis.

This research would not have been possible without funding from the Engineering and Physical Sciences Research Council (EPSRC) and the Newton Trust.

Declaration

This dissertation is the result of my own work, and includes nothing which is the outcome of collaborative work. The length of this dissertation, including all footnotes, appendices and the bibliography, is approximately 50,000 words. There are 62 figures.

Contents

1. Introduction

Which words can go together to make sentences or phrases that people are likely to say? If there is a complete answer to this question then it must be as wide-ranging and complex as a full description of language itself, yet *language modelling* aims to resolve this issue by constructing systems which are capable of finding solutions to exactly this conundrum.

This chapter provides a brief introduction to language modelling in the context of its use in speech recognition systems and then concludes with an overview of the rest of this thesis.

1.1 Speech Recognition

Every day the vast majority of us will read, write, speak and listen, and by and large we don't have too much difficulty working out which words we happen to be hearing or reading at any particular moment. Sometimes we can't quite make out a word or few, but usually we can resolve the ambiguity quickly by stopping to think for a second about what would make the most sense. Immediately, then, we have an advantage over any automated language recognition system – we can make use of an underlying, innate understanding of the *meaning* of words, which is a step above the simple consideration of which word sequences are most likely on a purely statistical level.

We can get by without knowing for certain that "black cat" is more frequently said than "black mat" because it will usually be obvious to us which of the two to infer given a particular situation. For a computer, however, none of this is obvious – every inference must be learnt as a rule or statistic. An entity which encapsulates knowledge about likely word sequences is called a *language model*, and by definition it attempts to model language by being able to predict likely word sequences and to attach probabilities to observed words. Since no one yet knows how to make computers think for themselves,

language models must either be based on a statistical analysis of language or they must be given specific rules to follow by human teachers.

It's not just language that computers need to be taught in order to be able to recognise speech – they also require the ability to convert audio waveforms into hypothesised word sequences. Computers 'hear' sound by sampling audio digitally, representing it as a series of discrete, regularly-spaced amplitude observations which together approximate the original sound waveform. These can be analysed mathematically to reveal the frequency of the sounds being observed. By studying the changing frequency patterns over time, *acoustic models* can be built to estimate the likelihood of different phoneme sequences – combined with a phoneme-to-word dictionary and a language model a full speech recognition system can then be constructed.

1.1.1 Language models

As described above, language models can be either statistical or rule-based. Rule-based models are typically based on *grammars* constructed to specify which sentences are permissible and which are not. With the exception of highly restricted applications where it is feasible to consider all possibilities, such rule-based models fail in situations which were not envisaged by the author of the rules. In addition they suffer from the problem that human speakers tend not to speak in purely grammatical sentences,[1] and manual updating can be required to include new words or structures – this might be necessary in order to cope with ever-evolving usage or to add additional coverage within a restricted application. Adaptation to new tasks is also difficult, with no guarantee that any existing rules can be reused. Statistical models, on the other hand, can alleviate these problems.

Statistical models infer properties of language from the analysis of suitably large quantities of text, basing their estimates on the relative frequencies of word occurrences and co-occurrences. They can be designed so as to always allow any possible sequence of words, even if that sequence is assigned a very low probability, and because they are based on automatic text analysis they can be rebuilt for a different domain, task or speaker simply by retraining with appropriate text.

[1] Possible errors include repetitions, hesitation words and sentence or word fragments.

1.1.2 Training language models

Training a statistical language model requires large quantities of text – typically many millions of words – from which to deduce reliable probability estimates. Many suitable sources of text exist and various standard corpora are available, covering areas such as broadcast news transcriptions – scripts from TV and radio news shows – and conversational speech. The performance of a language model is dependent upon how well matched its training text is with the domain to which it is applied, so for example to recognise speech from TV news shows language models are trained using broadcast news transcriptions – models trained on less appropriate text are likely to have inferior performance. Differences between various sources can be manifested in terms of style, sentence structure – including any common speaker errors – and vocabulary coverage. One possible solution to the problems resulting from these issues is to develop *adaptive* language models which contain features which modify the model on the basis of the current domain.

1.1.3 Other applications of language models

The work described in this thesis is focussed on the use of language models in speech recognition, but they can also be applied to other fields. Areas such as handwriting recognition, machine translation and even spelling and grammar checkers all benefit from the integration of a model which can assess the likelihood of given word sequences.

1.2 Statistical models

A speech recognition system takes as input a sequence of observed audio samples and aims to convert them into the most likely word sequence associated with those samples.

The digitised input waveform is first processed into a sequence of feature vectors, O. The recognition process can then be described as the search for the most likely word sequence, $\hat{W}$, given those observations:

$$\hat{W} = \underset{W}{\arg\max}\, P(W|O) \tag{1.1}$$

The conditional probability $P(W|O)$ can be rewritten using Bayes' law [Bayes 1763] as:

$$P(W|O) = \frac{P(O|W)\,P(W)}{P(O)} \tag{1.2}$$

Equation 1.2 can be substituted into equation 1.1, noting that $P(O)$ is independent of W and so need not be considered in the search:

$$\hat{W} = \underset{W}{\operatorname{argmax}}(P(O|W)\,P(W)) \tag{1.3}$$

Training for speech recognition can therefore be broken down into the dual tasks of finding estimates for both $P(O|W)$ and $P(W)$ – the former is encapsulated within an acoustic model and the latter within a language model.

1.2.1 Applying the language model

Practical speech recognition systems combine simple language models directly into the acoustic recognition process in order to produce a first-pass solution. Recognition systems can also export a *lattice*[2] of possible word sequences with each word-to-word transition edge marked with its acoustic likelihood. These lattices provide a reduced space representation of the recogniser state history by virtue of *pruning* the least-likely solutions, and allow more complex language models to be applied to rescore the various possible paths through the lattice and potentially produce a different second-pass solution.

The number of potential paths can remain large, even within a lattice,[3] so a speech recogniser can also produce a list of the N best solutions found – so-called *N-best lists* [Schwartz and Chow 1990]. These may or may not include different acoustic scores attached to each possible sequence of words, but by limiting the number of solutions to be scored to a fixed number, N, slow and complex language models, or ones which need to maintain an amount of context too large to be used with a lattice or directly in a recognition system, can be used. N-best lists also allow language models to consider future words as well as previously-seen words without having to maintain many long state histories.

[2] A connected graph of possible word paths.

[3] How large depends upon how heavily pruned the lattice is.

The use of lattices and *N*-best lists does have drawbacks, however. In the former case many 'correct' paths may have been pruned from the lattice, and it may well be the case that the 100%-correct solution does not exist within a given lattice. With *N*-best lists the number of possible hypotheses is typically many times lower again, so the possible improvements that can be obtained are often severely limited – if acoustic scores are omitted then even more information is lost. Overall, the more heavily pruned the number of considered hypotheses then the more likely it is that search errors will occur. Lattice rescoring was used for the word recognition experiments described in this thesis.

1.2.2 Recognition system performance

Speech recognition systems are usually assessed on the basis of the accuracy of their output, and a standard and intuitive form of assessment is *word error rate* (WER). An associated measure, *word accuracy*, is defined as 100% minus the word error rate. Word accuracy is calculated by taking account of the number of words in the reference transcription, *C*, the number of word insertions, *I*, word deletions, *D*, and word substitutions, *S*, as follows:

$$\text{Word accuracy (\%)} = \frac{C - D - S - I}{C} \times 100\% \tag{1.4}$$

In order to evaluate *I*, *D* and *S* the recognised output must be aligned with the correct reference transcription – this is performed using an optimal string match via dynamic programming. This optimal matching is implemented by minimising the Levenshtein distance between the two strings with a scoring metric which assigns a score of 0 for matching words and positive scores for insertions, deletions and substitutions.[4]

1.3 Work reported in this thesis

To date the most successful single-component statistical language models centre around counting the number of occurrences of word tuples in text corpora and then associating probabilities with each of these on the basis of the frequency with which they are observed in the training text – such a model is called a word *n*-gram model and is

[4] Standard scores, as used by NIST in various system evaluations, are 3, 3 and 4 for insertions, deletions and substitutions respectively – see `http://www.nist.gov/speech/`.

described in detail in section 2.2. It is also possible to count tuples of entities other than words – one possible model is a class *n*-gram model, where the probabilities of different word class sequences are modelled, with each word class consisting of one or more words. Such a model has the potential to capture more general information about language, and requires less text to produce good estimates – there is the additional problem of finding suitable word classes, however. Class *n*-gram models are introduced more formally in section 2.2.3.

In this thesis two modifications to the basic operation of a class *n*-gram model are described. The first of these is a topic adaptation model, where the relative probability distributions of words within a class are modified on the basis of what is perceived to be the current topic. In order to motivate this work and to obtain topic sets which are as applicable as possible to the model, a detailed examination of automatic topic clustering and the quality of its results is first undertaken.

In the second new model described herein, the method most commonly used to assign words to classes is refined to allow words to change class on the basis of their immediate left context, whilst maintaining the existing deterministic class map constraint of allowing a word to be in only one class at any one given time.

1.4 Thesis structure

This chapter has provided a brief overview of speech recognition, whilst the following chapter proceeds to describe the field of language modelling in detail, with a particular focus on developments which are relevant to the new work described later in this thesis, including a detailed examination of current class models. Chapter 3 examines methods of unsupervised statistical topic clustering from text corpora, and describes and assesses various methods with respect to their applicability for use in *n*-gram language models. Chapter 4 uses the results of the preceding chapter to motivate and develop a new topic adaptation class-based *n*-gram language model, which is then assessed and compared with various other alternative formulations. Next, chapter 5 introduces a class *n*-gram model which uses a class map based on pairs rather than single words, and describes an algorithm for finding a suitable class map. The performance of the proposed model using classes found by this algorithm is then assessed in detail and compared with the standard class *n*-gram model. Finally, chapter 6 summarises the

results of the work reported throughout this thesis before arriving at some conclusions and considering possible future work.

2. Language Modelling

The principle aim of the work reported in this thesis is to describe methods of improving on the current performance of language models. In order for this work to be placed in context, however, it is necessary to assess the existing state of affairs in this field. This chapter, therefore, seeks to provide a comprehensive overview of statistical language modelling with particular focus on the models which currently perform best – *n*-gram word and *n*-gram class models. Methods of robust model estimation, combination and assessment are also considered.

2.1 Evaluating a Language Model

The best test of a language model designed for speech recognition is, unsurprisingly, to use it in a speech recognition system. Only in this way can the interaction between the model and the acoustic part of the system be properly evaluated – a model stands to give better gains by distinguishing between those word sequences which are poorly disambiguated by the acoustic model than those which are already acoustically very dissimilar. It is not always practical to evaluate language models in full recognition systems, however, due to the computational complexity of the overall task making it somewhat slow, especially for development purposes. This problem is exacerbated by the necessity of using large test sets in order to ensure that a wide-enough sample is used to obtain a statistically meaningful assessment, whilst some models cannot feasibly be used directly within the speech recognition process in any case. Lattice rescoring retains the same problems at a less extreme level, whilst *N*-best list rescoring typically provides a significant limit on the improvement that a language model can obtain.

It is fair to conclude, therefore, that some other method is needed in order to reduce the computational requirements of language model assessment, at least during development and testing, to a more feasible level.

2.1.1 Measuring performance

The word string output of a speech recognition system allows the distance of the recognised text from a correct reference transcription to be computed in terms of word or sentence error rate. Similarly, a language model can be assessed by finding the distance of the model from some given reference text, the distance being assessed in terms of the probability the model assigns to that text.

When assessing a language model a fair test is required to ensure the results are not misleading. This means that the test must use unseen text which the model was not trained on, allowing an assessment to be made of how well the model generalises – a trivial model of *seen* text would be to simply store it all in sequence and perform a context look-up to retrieve the expected word every time. In order to produce an overall assessment of the model, however, its average performance is needed rather than its specific performance on one particular piece of text, so a result normalised by the number of words assessed is required. Higher average probabilities indicate better language models. It also follows that the more test text is used then the more confidence can be attached to the result.

2.1.2 Entropy

A measure of language model performance based on average probability can be developed within the field of information theory [Shannon 1948]. A speaker emitting language can be considered to be a discrete information source which is generating a sequence of words $w_1, w_2, \ldots, w_m$ from a vocabulary set, $\mathbb{W}$. The probability of a symbol w_i is dependent upon the previous symbols $w_1, \ldots, w_{i-1}$. The information source's inherent per-word entropy H represents the amount of non-redundant information provided by each new word on average, defined in bits as:

$$H = -\lim_{m\to\infty} \frac{1}{m} \sum_{w_1, w_2, \ldots, w_m} \left(P(w_1, w_2, \ldots, w_m) \log_2 P(w_1, w_2, \ldots, w_m) \right) \qquad (2.1)$$

This summation is over all possible sequences of words, but if the source is ergodic – that is, it has the property that the probability of any state can be estimated from a large enough history independent of the starting conditions[1] – then the summation over all

[1] See section 5 of [Shannon 1948] for a more formal definition of ergodicity.

possible word sequences can be discarded and the equation becomes equivalent to:

$$H = -\lim_{m\to\infty} \frac{1}{m} \log_2 P(w_1, w_2, \ldots, w_m) \tag{2.2}$$

It is reasonable to assume ergodicity on the basis that we use language successfully without having been privy to all words that have ever been spoken or written, and we can disambiguate words on the basis of only the recent parts of a conversation or piece of text.

Having assumed this ergodic property, it follows that given a large enough value of m, H can be approximated with:

$$\hat{H} = -\frac{1}{m} \log_2 P(w_1, w_2, \ldots, w_m) \tag{2.3}$$

This last estimate is feasible to evaluate, thus providing the basis for a metric suitable for assessing the performance of a language model.

2.1.3 Perplexity

Considering a language model as an information source, it follows that a language model which took advantage of all possible features of language to predict words would also achieve a per-word entropy of H. It thus makes sense to use a measure related to entropy to assess the actual performance of a language model. Perplexity [Jelinek 1977, Bahl et al 1983], PP, is one such measure that is in standard use, defined such that:

$$PP = 2^{\hat{H}} \tag{2.4}$$

Substituting equation 2.3 into equation 2.4 and rearranging obtains:

$$PP = \hat{P}(w_1, w_2, \ldots, w_m)^{-\frac{1}{m}} \tag{2.5}$$

where $\hat{P}(w_1, w_2, \ldots, w_m)$ is the probability estimate assigned to the word sequence (w_1, w_2, $\ldots$, w_m) by a language model.

Perplexity can be considered to be a measure of on average how many different equally most probable words can follow any given word. Lower perplexities represent better language models, although this simply means that they 'model language better', rather than necessarily work better in speech recognition systems – perplexity is only loosely

correlated with performance in a speech recognition system since it has no ability to note the relevance of acoustically similar or dissimilar words.

In order to calculate perplexity both a language model and some test text are required, so a meaningful comparison between two language models on the basis of perplexity requires the same test text and word vocabulary set to have been used in both cases. The size of the vocabulary can easily be seen to be relevant because as its cardinality is reduced so the number of possible words given any history must monotonically decrease, therefore the probability estimates must on average increase and so the perplexity will decrease.

Alternatives to perplexity have been proposed, such as those described in [Chen (S.) et al 1998], [Clarkson and Robinson 1999, Clarkson and Robinson 2001] or [Printz and Olsen 2000], but are not in general usage, perhaps due to the additional complexity and only marginal improvements in word error rate correlation they provide. In [Bimbot et al 2001] an alternative method for estimating perplexity is described whereby language models hypothesise words to follow sentence fragments which can then be analysed independently, removing a possible source of error due to faulty model normalisation code and allowing fair comparisons across sites.

2.2 *n*-gram language models

Language models estimate the probability of a word sequence, $\hat{P}(w_1, w_2, \ldots, w_m)$ – that is, they evaluate $P(W)$ as defined in equation 1.3 on page 15.

The probability $\hat{P}(w_1, w_2, \ldots, w_m)$ can be decomposed as a product of conditional probabilities:

$$\hat{P}(w_1, w_2, \ldots, w_m) = \prod_{i=1}^{m} \hat{P}(w_i \mid w_1, \ldots, w_{i-1}) \tag{2.6}$$

via use of the chain rule.

2.2.1 Word *n*-gram models

Equation 2.6 presents an opportunity for approximating $\hat{P}(W)$ by limiting the context:

$$\hat{P}(w_1, w_2, \ldots, w_m) \simeq \prod_{i=1}^{m} \hat{P}(w_i \mid w_{i-n+1}, \ldots, w_{i-1}) \tag{2.7}$$

for some $n \geqslant 1$. If language is assumed to be ergodic then for sufficiently high n equation 2.7 is exact. Due to reasons of data sparsity, however, values of n in the range of 1 to 4 inclusive are typically used, and there are also practicalities of storage space for these estimates to consider. Models using contiguous but limited context in this way are usually referred to as n-gram language models,[2] and the conditional context component of the probability ("$w_{i-n+1}, \ldots, w_{i-1}$" in equation 2.7) is referred to as the *history*.

Estimates of probabilities in n-gram models are commonly based on maximum likelihood estimates – that is, by counting events in context on some given training text:

$$\hat{P}(w_i | w_{i-n+1}, \ldots, w_{i-1}) = \frac{C(w_{i-n+1}, \ldots, w_i)}{C(w_{i-n+1}, \ldots, w_{i-1})} \tag{2.8}$$

where $C(.)$ is the count of a given word sequence in the training text. Refinements to this maximum likelihood estimate are described later in this chapter.

The choice of n has a significant effect on the number of parameters that the model has, which is maximally bounded by $|\mathbb{W}|^n$. A 4-gram model with a typically-sized 65,000 word vocabulary can therefore potentially have $65,000^4 \simeq 1.8 \times 10^{19}$ parameters. In practice, however, only a small subset of the possible parameter combinations represent likely word sequences, so the storage requirement is far less than this theoretical maximum – of the order of 10^{11} times less in fact.[3] Even given this significant reduction in coverage and a very large training text[4] there are still many plausible word sequences which will not be encountered in the training text, or will not be found a statistically significant number of times. It would not be sensible to assign all unseen sequences zero probability, so methods of coping with low and zero occurrence word tuples have been developed. This is discussed later in section 2.4

It is not only the storage space that must be considered, however – it is also necessary to be able to attach a reasonable degree of confidence to the derived estimates. Suitably large quantities of example training text are also therefore required to ensure statistical significance. Increasing the amount of training text not only gives greater confidence in model estimates, however, but also demands more storage space and longer analysis periods when estimating model parameters, which may place feasibility limits on how much data can be used in constructing the final model or how thoroughly it can

[2] As first used in speech recognition by [Jelinek 1976].

[3] Based on the analysis of 170 million words of newspaper and broadcast news text.

[4] A couple of hundred million words, for example.

be analysed. At the other end of the scale, for restricted domain models there may be only a limited quantity of suitable in-domain text available, so local estimates may need smoothing with global priors. In addition, if language models are to be used for speech recognition then it is good to train them on *precise* acoustic transcriptions where possible – that is, text which features the hesitations, repetitions, word fragments, mistakes and all the other sources of deviation from purely grammatical language that characterise everyday speech. However, such acoustically accurate transcriptions are in limited supply since they must be specifically prepared; real-world transcripts as available for various other purposes almost ubiquitously correct any disfluencies or mistakes made by speakers.

2.2.2 Equivalence classes

The word *n*-gram model described in equation 2.7 uses an equivalence mapping on the word history which assumes that all contexts which have the same most recent $n-1$ words all have the same probability. This concept can be expressed more generally by defining an equivalence class function that acts on word histories, $E(.)$, such that if $E(x) = E(y)$ then $\forall w : P(w|x) = P(w|y)$:

$$P(w_i \mid w_1, w_2, \ldots, w_{i-1}) = P(w_i \mid E(w_1, w_2, \ldots, w_{i-1})) \tag{2.9}$$

A definition of E that describes a word *n*-gram is thus:

$$E_{\text{word-}n\text{-gram}}(w_1, \ldots, w_i) = E(w_{i-n+1}, \ldots, w_i) \tag{2.10}$$

In a good language model the choice of E should be such that it provides a reliable predictor of the next word, resulting in classes which occur frequently enough in the training text that they can be well modelled, and does not result in so many distinct history equivalence classes that it is infeasible to store or analyse all the resultant separate probabilities.

2.2.3 Class *n*-gram models

One method of reducing the number of word history equivalence classes to be modelled in the *n*-gram case is to consider some words as equivalent. This can be implemented by mapping a set of words to a word class $g \in \mathbb{G}$ by using a classification

function $G(w) = g$. If any class contains more than one word then this mapping will result in less distinct word classes than there are words, $|\mathbb{G}| < |\mathbb{W}|$, thus reducing the number of separate contexts that must be considered. The equivalence classes can then be described as a sequence of these classes:

$$E_{\text{class-}n\text{-gram}}(w_1, \ldots, w_i) = E(G(w_{i-n+1}), \ldots, G(w_i)) \tag{2.11}$$

A deterministic word-to-class mapping like this has some advantages over a word *n*-gram model since the reduction in the number of distinct histories reduces the storage space and training data requirements whilst improving the robustness of the probability estimates for a given quantity of training data. Because multiple words can be mapped to the same class, the model has the ability to make more confident assumptions about infrequent words in a class based on other more frequent words in the same class[5] than is possible in the word *n*-gram case – and furthermore for the same reason it is able to make generalising assumptions about words used in contexts which are not explicitly encountered in the training text. These gains, however, clearly correspond with a loss in the ability to distinguish between different histories, although this might be offset by the ability to choose a higher value of *n*.

The most commonly used form of class *n*-gram model uses a single classification function, $G(.)$, as in equation 2.11, which is applied to each word in the *n*-gram, including the word which is being predicted.[6] Considering for clarity the bigram[7] case, then given $G(.)$ the language model has the terms w_i, w_{i-1}, $G(w_i)$ and $G(w_{i-1})$ available to it. The probability estimate can be decomposed as follows:

$$\begin{aligned} P_{\text{class'}}(w_i \mid w_{i-1}) \quad &= \quad P(w_i \mid G(w_i), G(w_{i-1}), w_{i-1}) \\ &\times \quad P(G(w_i) \mid G(w_{i-1}), w_{i-1}) \end{aligned} \tag{2.12}$$

It is assumed that $P(w_i \mid G(w_i), G(w_{i-1}), w_{i-1})$ is independent of $G(w_{i-1})$ and w_{i-1} and

[5] Since it is assumed that words are placed in the same class because they share certain properties.

[6] This is the model that is generally implied when referring simply to *a class-based n-gram model* and is what is implied by this phrase throughout this thesis. This is also referred to as the *two-sided* variant in [Whittaker 2000] but this same term is used to refer to the type of class model defined in equation 2.15 in [Ney et al 1994] so I will avoid this terminology and when disambiguation is required I will simply refer to it as the *standard* form of a class model.

[7] By convention *unigram* refers to a 1-gram, *bigram* indicates a 2-gram and *trigram* is a 3-gram. There is no standard term for a 4-gram.

that $P(G(w_i) \mid G(w_{i-1}), w_{i-1})$ is independent of w_{i-1}, resulting in the model:

$$P_{\text{class}}(w_i \mid w_{i-1}) = P(w_i \mid G(w_i)) \times P(G(w_i) \mid G(w_{i-1})) \tag{2.13}$$

Many variants of this model are possible. For example, the independence assumptions can be modified or different classifications can be applied to specific positions in the word history – in equation 2.11 the same class map function $G(.)$ was applied to each word irrespective of its position in the history, but this need not be the case:

$$E_{\text{class-}n\text{-gram'}}(w_1, \ldots, w_i) = E(G_n(w_{i-n+1}), \ldots, G_1(w_i)) \tag{2.14}$$

given a set of class map functions, $G_1(.), \ldots, G_n(.)$.

Alternatively, entire word histories can be clustered into classes:

$$E_{\text{class-}n\text{-gram''}}(w_1, \ldots, w_i) = E(G(w_{i-n+1}, \ldots, w_i)) \tag{2.15}$$

These two variants are described in, for example, [Ueberla 1995] – see section 2.3.1.

One-sided model

Using equation 2.11 to define history equivalence classes but not also applying $G(.)$ to the word being predicted produces the one-sided class language model described in [Whittaker and Woodland 2001]:

$$P_{\text{class-1s}}(w_i \mid w_{i-n+1}, \ldots, w_{i-1}) = P(w_i \mid G(w_{i-n+1}), \ldots, G(w_{i-1})) \tag{2.16}$$

Although the following observation is not made in the reported work, this model can be justified by noting that if the word-given-class probability is expanded to be dependent on not only the current class but also the preceding class then this is equivalent to the one-sided class model. Taking a bigram example:

$$\begin{aligned}
P_{\text{class}-2}(w_i \mid w_{i-1}) &= P(w_i \mid G(w_i), G(w_{i-1})) \times P(G(w_i) \mid G(w_{i-1})) \\
&= \frac{P(w_i, G(w_i), G(w_{i-1}))}{P(G(w_i), G(w_{i-1}))} \times \frac{P(G(w_i), G(w_{i-1}))}{P(G(w_{i-1}))} \\
&= \frac{P(w_i, G(w_i), G(w_{i-1}))}{P(G(w_{i-1}))} \\
&= P(w_i, G(w_i) \mid G(w_{i-1})) \\
&= P(w_i \mid G(w_{i-1})) \\
&= P_{\text{class}-1\text{s}}(w_i \mid w_{i-1})
\end{aligned} \tag{2.17}$$

Although true in terms of maximum likelihood estimates, discounting, smoothing or other approximation methods would typically be applied to the initial decomposed estimates,[8] so the models would not necessarily be equivalent. This derivation also relies on a deterministic class mapping for the penultimate step: $P(G(w) \mid w) = 1$.

The advantage of the one-sided model is that when using a bigram clustering algorithm[9] a class map can be constructed much more quickly than in the standard case, due to a linear as opposed to quadratic computational scaling with the number of classes – $O(|\mathbb{G}|.|\mathbb{W}|)$ versus $O(|\mathbb{G}|^2.|\mathbb{W}|)$. Both methods scale linearly with vocabulary size, $|\mathbb{W}|$.[10] It is due to the reduced order of the number of classes term, $|\mathbb{G}|$, that it is possible to cluster into a given number of classes with far less computation, although the resulting model will generally have more parameters. Experimentally the one-sided model performs better than the standard model when used alone, as might be expected due to the greater specificity of its predictions, but when interpolated with a word *n*-gram the perplexity is worse than an interpolated standard class model with the same number of classes [Whittaker 2000]. The recognition word error rate performance is similar for both class model variants, however [Whittaker and Woodland 2001].

Almost all reported class *n*-gram work using statistically-found classes is based on clustering algorithms which optimise $G(.)$ on the basis of bigram training set likelihood, even if the class map is to be used with longer-context models. For example, the one-sided work discussed above is no exception, despite assessing overall performance through use of trigram class models. It is interesting to note that this approximation appears to works well, however, suggesting that the class maps found are in some respects "general" and capture some features of natural language which apply irrespective of the context length used when finding these features. The fact that this observation holds true with the one-sided model is perhaps even more surprising, given that the bigrams used for clustering consist of one word and only one class.

Work in [Martin, Liermann and Ney 1998] compares the class maps produced by a class trigram clusterer with those from a bigram clusterer when used in a trigram model, and only on the largest sets is there found to be an improvement in the language model performance when a trigram optimisation measure is used for clustering. [Whittaker

[8] See section 2.4.1 on page 42.

[9] Such as those described in section 2.3.1 on page 30.

[10] In practice the order of the two-sided algorithm is less than quadratic, but it is also substantially more than linear – it depends on the coverage of class bigrams by the training text being used.

2000], however, reports that interchanging class maps between the one-sided and standard variants produces poorly-performing models, so perhaps the class maps are not quite so general after all.

[Whittaker 2000] also examines the use of a trigram position-dependent class map function – as in equation 2.14 – with the one-sided model, obtaining an average 1% improvement in perplexity over the position-independent version. In this case the position-dependent functions $G_1(.)$ and $G_2(.)$ are found with two successive runs of the clustering algorithm rather than optimised simultaneously – the second run uses distance-2 bigrams; that is, trigrams with the middle word omitted.[11] It is acknowledged that this result might have been improved if a single run of the clusterer had been used to optimise both classification functions simultaneously, but this would have lost much of the speed advantage of the one-sided algorithm.

2.2.4 Multiple class membership

Classification schemes group similar words together, but it is clear that when using hard classifications[12] that sometimes a word will be placed into a sub-optimal class – homographs, differing parts of speech and varying usage are all examples of why a fixed classification scheme must be suboptimal. For example, `BOW` can mean the head of a boat, a mark of respect, a weapon, a decorative ribbon, a dog starting to bark or around 30 other things.[13] It is unlikely that there is another word which has exactly this same set of meanings, so if `BOW` is not in a singleton class then at least one member of its class cannot be accurately modelled in all contexts. Nearly all words in English have multiple meanings or usages, so it is clear that a hard classification scheme has inherent limits to its modelling ability.

Some class models aim to avoid many of the generalisations inherent with a hard classification scheme by allowing non-deterministic class membership functions which distribute a word across an arbitrary number of classes, usually based on some probabilistic distribution. To evaluate a model of this form, all possible mappings of the word history must be summed, weighted by the probability of each mapping.

[11] See section 2.6.2 on page 50 for a discussion of distance *n*-grams.

[12] *Hard* or *deterministic* classifications are those in which there is never any ambiguity about which class a word is in.

[13] According to the Oxford English Dictionary.

Consider a set $M(w_i)$ of classes that word w_i can be found in. Given this ambiguity, let $O(w_i)$ be the set of all possible class histories of w_i. The probability of word w_i can then be defined as:

$$P_{\text{class-m}}(w_i|w_1,\ldots,w_{i-1}) = \sum_{m\in M(w_i)} P(w_i|m) \times \sum_{o\in O(w_i)} P(m|o) \times P(o|w_1,\ldots,w_{i-1}) \quad (2.18)$$

A multiple class membership language model like this is used, for example, in [Niesler and Woodland 1996/ICASSP] – see section 2.3.2. A model with ambiguous classes is potentially slower in use and more expensive in its memory consumption, however, through the necessity to maintain and evaluate each prediction on the basis of multiple possible histories.

Two additional considerations with multiple class membership are how to decide which classes a word can be in – the search space is much larger than that for deterministic classes, which is already very large – and how to decide which class a word should be in when tagging text for the purpose of training the model. The ubiquitous solution to this is to base classes on grammatical parts of speech, as will be illustrated in the following section.

2.3 Finding Class Maps

An obvious question that arises is how to compute or otherwise obtain a class map for use in a language model. This section discusses various strategies which have been proposed, including statistical clustering and linguistically-derived classes. Grammar-based classes are also examined, as well as a novel acoustic algorithm.

2.3.1 Statistically-derived maps

Methods of statistical class map construction seek to maximise the likelihood of the training text given the class model by making iterative controlled changes to an initial class map – in order to make this problem more computationally feasible they typically use a deterministic map.

Word exchange algorithm

[Kneser and Ney 1993] describe an algorithm to build a class map by starting from some initial guess at a solution and then iteratively searching for changes to improve the existing class map. This is repeated until some minimum change threshold has been reached or a chosen number of iterations have been performed. The initial guess at a class map is typically chosen by a simple method such as randomly distributing words amongst classes or placing all words in the first class except for the most frequent words which are put into singleton classes. Potential moves are then evaluated and those which increase the likelihood of the training text most are applied to the class map. The algorithm is described in detail below.

Let W be the training text list of words $(w_1, w_2, w_3, \dots)$ and let $\mathbb{W}$ be the set of all words in W. From equation 2.6 on page 22 it follows that:

$$P_{\text{class}}(W) = \prod_{x,y \in \mathbb{W}} P_{\text{class}}(x \mid y)^{C(x,y)} \tag{2.19}$$

where (x, y) is some word pair 'x' preceded by 'y' and $C(x, y)$ is the number of times that the word pair '$y\ x$' occurs in the list W.

In general evaluating equation 2.19 will lead to problematically small values, so logarithms can be used:

$$\log P_{\text{class}}(W) = \sum_{x,y \in \mathbb{W}} C(x, y). \log P_{\text{class}}(x \mid y) \tag{2.20}$$

Given the definition of a class n-gram model in equation 2.13, the maximum likelihood bigram probability estimate of a word is:

$$P_{\text{class}}(w_i \mid w_{i-1}) = \frac{C(w_i)}{C(G(w_i))} \times \frac{C(G(w_i), G(w_{i-1}))}{C(G(w_{i-1}))} \tag{2.21}$$

where $C(w)$ is the number of times that the word 'w' occurs in the list W and $C(G(w))$ is the number of times that the class $G(w)$ occurs in the list resulting from applying $G(.)$ to each entry of W;[14] similarly $C(G(w_x), G(w_y))$ is the count of the class pair '$G(w_y)\ G(w_x)$' in that resultant list.

[14] That is, $C(G(w)) = \sum_{x:G(x)=G(w)} C(x)$.

Substituting equation 2.21 into equation 2.20 and then rearranging gives:

$$\begin{aligned}
\log P_{\text{class}}(W) &= \sum_{x,y\in\mathbb{W}} C(x,y).\log\left(\frac{C(x)}{C(G(x))}\times\frac{C(G(x),G(y))}{C(G(y))}\right)\\
&= \sum_{x,y\in\mathbb{W}} C(x,y).\log\left(\frac{C(x)}{C(G(x))}\right) + \sum_{x,y\in\mathbb{W}} C(x,y).\log\left(\frac{C(G(x),G(y))}{C(G(y))}\right)\\
&= \sum_{x\in\mathbb{W}} C(x).\log\left(\frac{C(x)}{C(G(x))}\right) + \sum_{g,h\in\mathbb{G}} C(g,h).\log\left(\frac{C(g,h)}{C(h)}\right)\\
&= \sum_{x\in\mathbb{W}} C(x).\log C(x) - \sum_{x\in\mathbb{W}} C(x).\log C(G(x))\\
&\quad + \sum_{g,h\in\mathbb{G}} C(g,h).\log C(g,h) - \sum_{g\in\mathbb{G}} C(g).\log C(g)\\
&= \sum_{x\in\mathbb{W}} C(x).\log C(x) + \sum_{g,h\in\mathbb{G}} C(g,h).\log C(g,h)\\
&\quad - 2\sum_{g\in\mathbb{G}} C(g).\log C(g) \qquad (2.22)
\end{aligned}$$

where (g,h) is some class sequence 'h g'.

Note that the first of these three terms in the final stage of equation 2.22, "$\sum_{x\in\mathbb{W}} C(x)$. $\log(C(x))$", is independent of the class map function $G(.)$, therefore it is not necessary to consider it when optimising $G(.)$. The value a class map must seek to maximise, F_{M_C}, can now be defined:

$$F_{\text{M}_\text{C}} = \sum_{g,h\in\mathbb{G}} C(g,h).\log C(g,h) - 2\sum_{g\in\mathbb{G}} C(g).\log C(g) \qquad (2.23)$$

A fixed number of classes must be decided before running the algorithm, which can now be formally defined:

1. **Initialise**: $\forall w \in \mathbb{W} : \; G(w) = 1$
   Set up the class map so that all words are in the first class and all other classes are empty (*or* initialise using some other scheme)

2. **Iterate**: $\forall i \in \{1 \ldots n\} \; \wedge \; \neg s$
   For a given number of iterations $1 \ldots n$ or until some stop criterion s is fulfilled

   (a) **Iterate**: $\forall w \in \mathbb{W}$
   For each word w in the vocabulary

   i. **Iterate**: $\forall c \in \mathbb{G}$
   For each class c

   A. **Move** word w to class c, remembering its previous class

   B. **Calculate** the change in $F_{\mathrm{M_C}}$ for this move

   C. **Move** word w back to its previous class

   ii. **Move** word w to the class which increased $F_{\mathrm{M_C}}$ by the most, or do not move it if no move increased $F_{\mathrm{M_C}}$

The initialisation scheme given here in step 1 represents a word unigram language model, making no assumptions about which words should belong in which class.[15] The algorithm is greedy and so can get stuck in a local maximum and is therefore not guaranteed to find the optimal class map for the training text. The algorithm is rarely run until total convergence, however, and it is found in practice that an extra iteration can compensate for even a deliberately poor choice of initialisation.

The above algorithm requires the number of classes to be fixed before running. It should be noted that as the number of classes utilised increases so the overall likelihood of the training text will tend tend towards that of the word model.[16] This is why the algorithm does not itself modify the number of classes, otherwise it would naïvely

[15] Given this initialisation, the first $(|\mathbb{G}| - 1)$ moves will be to place each word into an empty class, however, since the class map which maximises $F_{\mathrm{M_C}}$ is the one which places each word into a singleton class.

[16] Which will be higher, given maximum likelihood estimates.

converge on $|\mathbb{W}|$ classes. A *leaving one out* method, a specific form of cross-validation described in [Ney and Essen 1993], was proposed for use in the exchange algorithm in [Kneser and Ney 1993], thus preventing the algorithm from tending towards an unchanging global optimum and so allowing it to also optimise the number of classes. The method of leaving-one-out as applied to the word exchange algorithm dictates that, when clustering, before each move is considered one count is omitted from the corpus, thus preventing the class map from over-modelling the training corpus – in this way it is ensured that there are different events not seen in the training data before each move. The withheld count is rotated throughout the corpus.

Brown's class-joining algorithm

If a clusterer uses maximum likelihood estimates of the text on which it is being optimised then increasing the number of classes is guaranteed to increase the log likelihood of the training text given that class model. [Brown et al 1992] uses this fact to describe a clusterer which begins with each word in a singleton class, with as many classes as there are words, and then successively merges classes so as to decrease the log likelihood as little as possible. This is repeated until a desired number of classes is reached. The algorithm then proceeds as per the exchange algorithm above, swapping words between classes so as to maximise the log likelihood whilst keeping the number of classes fixed.

In the reported work a modification to the algorithm is applied when more than 5000 words are in the vocabulary, whereby an additional initialisation stage is added before the algorithm starts merging classes. 5000 singleton classes are created for the 5000 most frequent words and then each of the $(|\mathbb{W}| - 5000)$ remaining words are in turn added to the class which will decrease the log likelihood the least.

In practice the class exchange algorithm is not particularly sensitive to its initialisation, especially if multiple iterations are run, and since the only relevant difference in Brown's algorithm is the initialisation of the classes its performance is very similar to the basic exchange clustering algorithm described in the previous section.

Jardino and Adda's simulated annealing

In [Jardino and Adda 1993/ICASSP] a simulated annealing word clustering algorithm is described and used to build a bigram class model. From an initial distribution of

each of the most frequent words in a singleton class – and the remaining words in one additional class – the algorithm proceeds by tentatively moving words to classes as in the word exchange algorithm, but the words and classes are chosen randomly at each stage rather than iterated over. All likelihood-increasing moves are accepted but so are some other moves with a probability that decreases as the algorithm proceeds. The acceptance of likelihood-decreasing moves is controlled via a parameter which initially accepts more than 90% of moves but is then logarithmically reduced to tend towards but never reach zero – this is the titular *annealed* element. This means that an optimal class map will result, so long as as an infinite number of moves can be made [van Laarhoven and Aarts 1987]! This implies the weaker conclusion that if run for long enough with suitable parameters that this algorithm will find a very good class map.

Further work is reported in [Jardino and Adda 1993/Eurospeech], but unfortunately neither this nor [Jardino and Adda 1993/ICASSP] compares the model performance with any other model, such as a word *n*-gram. In [Jardino 1994] the algorithm is extended to allow each word to be in one of two classes, where each word is split into two subsets – one consisting of the word and its most frequent immediate left context, and the other containing the word in all its other contexts. These are then clustered as separate entities. In their analysis on a French corpus it is concluded that by using this two-way context disambiguation significant gains in perplexity can be obtained, although for many words the model does not benefit from them being split into two classes – however it is unclear whether the number of classes is held constant in this comparison; if not then this gain could be as a result of the additional classes. In fact the number of classes used is not described anywhere, unfortunately.

In [Jardino 1996] it is claimed that the simulated annealing algorithm performs better than the exchange or Brown's algorithm described above, but this is based on theoretical claims derived from its ability to make likelihood-decreasing moves rather than experimental evidence – the rationale is that it cannot get stuck in local minima, but since in any practical timespan on any useful data set the exchange/Brown algorithms will not be run to convergence this advantage would appear to remain a theoretical one only.

Additional refinements

[Ueberla 1995] discusses extensions of class *n*-gram language models as per equations 2.14 and 2.15 on page 26. Observing that the latter of these is a superset of the former,

and that the former is itself a superset of the standard class model variant, equivalence class mappings of the entire history are investigated:

$$P_{\text{class-}n\text{-gram}}(w_i \mid w_{i-n+1}, \ldots, w_{i-1}) = P(G_0(w_i) \mid G_1(w_{i-n+1}, \ldots, w_{i-1})) \times P(w_i \mid G_0(w_i)) \quad (2.24)$$

The clustering algorithm used is based on the word exchange method above, but before each move a list of candidate target classes is computed – this risks omitting the best-possible class but allows a significant decrease from the quadratic computational complexity of the standard algorithm. This list is computed by finding the most common preceding classes for the class being considered as well as the most commonly following classes and keeping only those which are in both lists. It is reported that the new bigram model performs better than a bigram back off word model on smaller data sets, although not on the largest tests tried.[17] Extending to trigrams gave additional gains, and it is noted that these trigram models have a size of the same order of magnitude as the bigram models, which is not normally the case for standard *n*-gram models. The model does not back off because it represents varying context lengths explicitly in its history equivalence classes. Similar history mappings are also examined in [Ney et al 1994]. A model which can make arbitrary classifications on the recent history has at least as much power as a word *n*-gram, were it to be allowed enough classes.

A refinement to the algorithm is reported in [Ueberla and Gransden 1996], motivated on the basis that parts of the model based on contexts which have not been observed many times are poorly modelled. They propose, therefore, a clusterer that can move groups of contexts between classes rather than just individual contexts. To implement this they begin with all contexts grouped and then only separate them at a later stage in the algorithm. Specifically, using a trigram-based clusterer, they begin by tying together all trigram contexts (w_{i-1}, w_{i-2}) that have the same w_{i-1} and moving them between classes as a group until a given number of iterations have been performed – based on empirical observations they find that two iterations are usually sufficient.

In [Miller and Alleva 1996] it is reported that a hierarchical version of the word exchange algorithm gives improved results on their bigram evaluation system. Building a 256 class language model, they begin by clustering into 32 separate classes and then splitting each of these into a further 8 distinct classes to form the full 256. In a second iteration, words are constrained to only move between classes in the set of 8 formed

[17] Up to 1.7 million words of training data gave relative gains; 8.5 and 40 million word sets did not.

from their original class, before in a final third iteration words are freed to move to any class. They claim this works better because it encourages classes with all low probability words to form and removes the bias which otherwise exists towards ending up with all classes having at least one high frequency member. This can help reduce the level of noise in the class map induced by the dominance of the most frequently-occurring words.

[Mori et al 1998] claim that a better class map can be built by splitting the training corpus into multiple smaller corpora and trying to optimise the map simultaneously across all combinations of those corpora. They also claim that the deleted interpolation method of leaving-one-out proposed in [Kneser and Ney 1993] actually damages the performance of the resulting model. However, they report such huge improvements in perplexity of their smaller class bigram model over their larger baseline word bigram model[18] that it would appear that there were additional factors contributing to this reduction.

[Siu and Ostendorf 1997] store a full set of *n*-gram contexts by use of a branching tree structure, which they then proceed to use for various purposes such as not only building a skip *n*-gram model – see section 2.5.3 on page 47 – but also constructing some context-based classes for inclusion in the otherwise word-based language model. Using their tree structure they can trim contexts or generate skips by comparing child nodes with their parents or with the siblings of their parents respectively,[19] but they can also, however, compare child nodes with their own siblings – that is, those that share the same parent. By doing this they can look for similar distributions across subtrees, and if they find any which are close enough then they merge the sub-trees of these child nodes, thus tying together certain pairs of words when found in the context described by the relevant point in the tree – the degree of similarity is assessed on the basis of maximising the log likelihood of their model. Having added a limited selection of context-based classes to their word *n*-gram model, they report a small perplexity reduction from 77.4 to 77.2 and a word error rate drop from 35.29% to 35.10%.

2.3.2 Linguistically-derived maps

Grammatical parts of speech provide a predefined class mapping, albeit one which suggests strongly that words should be allowed into multiple classes since many have

[18] 203.5 reduced to 136.0 for word to class bigrams respectively.

[19] See section 2.5.3 later in this chapter.

multiple commonly-encountered parts of speech. Also unlike statistical class maps, models which use part of speech (POS) maps are constrained to modelling only on syntactic grounds, since no semantic information is encoded within parts of speech. There are various collections of manually-tagged text suitable for language model training,[20] as well as various automated part of speech tagging tools.[21]

Research using *n*-gram models constructed using POS class maps has not been encouraging, however, compared to the performance of statistically-found maps. For example, in [Ueberla and Gransden 1996] it is found that hard classes found with the word exchange algorithm give better performance than classes based on parts of speech with an additional advantage being, they note, that the number of classes can be determined at will in the statistical case. [Maltese et al 2001] provides a large amount of evidence across five different languages[22] for the same conclusion. Similarly, work in [Wakita et al 1996] concludes that use of raw POS tags performs notably worse than a classification metric which attempts to combine these with semantic categories when finding classes.

[Niesler and Woodland 1996/ICASSP] describe a class *n*-gram model which uses categories based on parts of speech. Each word is permitted to be in multiple classes, so a probabilistic mapping exists from each word history to each possible history equivalence class as per equation 2.18 on page 29. Training is performed via use of the tagged LOB corpus [Johansson et al 1986], although once trained the model can also be used to statistically tag extra text. Models are constructed which contain only a sixth of the number of parameters of a word bigram, but at the expense of a performance degradation measured in terms of perplexity.[23] In [Niesler 1997] word error rate gains are reported when interpolating this model with a trigram word model, whilst [Niesler et al 1998] confirms that the statistical class model outperforms the POS class model.

An alternative method of employing parts of speech whilst retaining implied ambiguity is described in [Samuelsson and Reichl 1999]. They use a deterministic class map with multiple POS assignments handled by using suitable joint ambiguity classes, refined by including a 'likelihood order' in the class identity in such ambiguity cases

[20] Such as the LOB corpus [Johansson et al 1986] or the Penn treebank [Marcus et al 1995].

[21] For example see those described in [Samuelsson and Voutilainen 1997].

[22] English, French, German, Italian and Spanish.

[23] Word bigram perplexity of 108.57 with 305,605 parameters versus POS-model perplexity of 138.53 with 54,547 parameters.

– long POS lists are truncated, however,[24] as are classes which occur infrequently or contain less than five members. They also reserve singleton classes for the 50 words which occur most frequently in their training text. Using a 20,000 word vocabulary and POS statistics from the Penn Treebank corpus [Marcus et al 1995] augmented with those from a linguistic tagger [Samuelsson and Voutilainen 1997], they build 305 classes from 28 POS tags in addition to the 50 singleton classes. With this method they obtain a small but worthwhile word error rate improvement over their baseline trigram word model when interpolating with it, but they do not compare with a class model found via statistical methods and do not report the performance of their model on its own.

Linguistically-derived categories can also be based on semantic groupings, such as numbers, colours, birds, names of politicians and so on – classifications such as this, however, are very subjective and require a great deal of human interaction to construct, so are limited by the amount of effort that can be invested in them. In [Ando et al 1998] a statistical word exchange clustering algorithm is employed but with movement between classes restricted to those classes which contain words which are linked according to thesaurus entries. Words found in the thesaurus are moved first followed by the remaining words. Unfortunately the reported work only compares with classes obtained directly from the thesaurus,[25] reporting a gain relative to this baseline. The experimental work was performed using Japanese text.

[Uebler and Niemann 1998] place words in classes based on a morphological decomposition of their prefixes and suffixes, working with German and Italian corpora. The rationale behind this is to allow recognition of out-of-vocabulary (OOV) words by matching them to classes with similar morphological properties. Experiments use deterministic classes chosen on the basis of identical prefixes, suffixes or both, with remaining words placed in either one large class or split amongst further classes in what is basically an *ad hoc* manner. They find that prefix-based classes work well for German, whilst combined prefixes and suffixes gives the best results on their Italian corpus, with reported worthwhile word error rate reductions obtained relative to semantic classes or word *n*-grams in the latter case. Use of a class model to help cope with OOV words is also considered in [Gallwitz et al 1996], in which OOV words are manually assigned to one of various special classes reserved for OOV words, obtaining a small word error rate reduction.

[24] "After 90% probability mass, or after four tags, whichever occurred first."
[25] Two metrics are described for resolving class membership ambiguity.

2.3.3 Grammar-based classes

Class-based language models have been proposed that allow arbitrary sequences of words to be modelled using a single class entry, with the contents of each class described by a context free grammar (CFG). In general, models of this form use non-deterministic classes and the grammar itself is frequently stochastic with probabilities associated with each parse. Both the grammars and any associated probabilities may be automatically or manually derived, but most work to date uses such models with manually defined grammars and automatically-derived probabilities. Grammars are typically based on specific restricted domain problems such as flight booking systems.

In [Gillett and Ward 1998] a class trigram model is defined where each class may be specified by a stochastic context free grammar. Their experiments, however, define all class grammars as a set of singleton words, as per the previously considered class models, with the exception of a few explicit exceptions such as:

[date]	⇒	{month} {day}
[month]	⇒	“january” | “february” | . . . | “december”
[day]	⇒	{digit} | {digit} {digit}
[digit]	⇒	“0” | “1” | . . . | “9”

In the experiments reported, manually-defined grammar classes built for a small medical transcription domain gave no significant word error rate reduction relative to a word trigram model, but on an even smaller travel agent task a significant decrease was reported – in both cases worthwhile perplexity reductions were obtained, however. A similar model was also proposed in [Meteer and Rholicek 1993], with a small word error rate improvement reported.[26]

[Hacioglu and Ward 2001] presents an extended form of the model described in [Gillett and Ward 1998] with the dialogue context taken into account when deciding on the class mapping to use. In a restricted domain flight, hotel and rental car reservation system, the system-generated questions are taken as the contexts for the user responses. Context-dependent models are built in two ways – firstly by including the context as a unique token instead of the sentence-start symbol in the *n*-grams; and secondly by building a separate model for each context. The reported experiments reveal that both

[26] No perplexity results were given.

methods give worthwhile perplexity reductions, but the first method performs better. The authors hypothesise that this is due to the small average sentence length in their corpus. Word error rate results are only reported for the second method, however, and show an improvement.

Work on automatically finding grammars for use with *n*-gram models is described in [Siu and Ostendorf 2000], whereby a context-dependent phrase grammar is developed. The grammar is implemented by use of a variable-length *n*-gram language model and the set of phrases is found by attempting to minimise perplexity on the training corpus, but only a small word error rate improvement is reported.

Grammars for adaptation

Using grammar classes may have some potential in situations where limited domain-specific data is available. This is on the basis that it might be faster to construct CFGs which contain the expected sentence structures than it would be to obtain a representative set of example sentences necessary in order to train a traditional *n*-gram model for the given restricted domain. The use of small CFGs within an *n*-gram provides the ability to generalise to unseen or unexpected constructs that a purely CFG-based model would not be able to do. Restrictions would still apply, however – consider in the above boxed example encountering a date which is not recognised by the grammar's '[date]' class and the poor estimation an associated CFG-based *n*-gram might well then be expected to make. Work reported in [Wang et al 2000] uses domain-dependent CFGs to augment a domain-independent *n*-gram language model in various ways. In-domain training data is generated using two methods from their CFGs – firstly by using domain-specific CFGs to generate text to build a trigram model, and secondly in a less-direct method by using that generated text with a similarity measure to select domain-related text from a general training corpus. The former of these two techniques worked much better in terms of perplexity with their experiments (reported pps of 207 and 271 respectively), but interpolating the two worked much better still (pp=112). These models were then used to score expansions of the CFG in order to estimate weights,[27] and a further improvement was obtained (pp=90).

[27] As opposed to having a uniform probability distribution amongst the various expansions of each CFG start symbol.

2.3.4 Acoustic classes

A class model is constructed in [Fischer and Kunzmann 2000] which is designed to maximise the acoustic dissimilarity between classes rather than the training set likelihood. Their clustering algorithm measures the similarity between two words in terms of the Levenshtein distance[28] between the dictionary phoneme sequence representations of each of the words. In their experiments, however, no account is taken of the acoustic distance between any two different phonemes, with the assumption made that any two distinct phonemes are all equally distant – perhaps an experimentally-found phoneme confusion matrix could have been included in some way to improve on this issue. This system also does not allow for alternative pronunciations, although the most likely pronunciation could perhaps be taken in each case. None the less, in one out of two tests they report a worthwhile decrease in word error rate when they interpolate their *n*-gram class model with their baseline German-language model. The baseline system includes a standard statistical class model, so the comparison is made by adding the acoustic class model as a third linearly interpolated component.[29]

2.4 Robust model estimation

Given a suitably large amount of training data, an extremely long *n*-gram could be trained to give a very good model of language, as per equation 2.6 – in practice, however, any actual extant model must be an approximation. Because it is an approximation, it will be detrimental to include within the model information which in fact was just noise introduced by the limits of the bounded sample set used to train the model – this information may not accurately represent text not contained within the training corpus. In the same way, word sequences which were not observed in the training text cannot be assumed to represent impossible sequences, so some probability mass must be reserved for these. The issue of how to redistribute the probability mass, as assigned by the maximum likelihood estimates derived from the raw statistics of a specific corpus, into a sensible estimate of the real world is addressed by various standard methods, all of which aim to create more robust language models.

[28] Implemented as the minimum number of substitutions, insertions and deletions necessary to transform one string into another.

[29] By omitting the standard class model as a component, the acoustic model plus word model obtains a significant improvement over the word model alone in both their tests.

2.4.1 Estimating probabilities

Language models seek to estimate the probability of each possible word sequence event occurring. In order to calculate maximum likelihood estimates this set of events must be finite so that the language model can ensure that the sum of the probabilities of all events is 1 given some context. In an n-gram model the combination of the finite vocabulary and fixed length history limits the number of unique events to $|\mathbb{W}|^n$. For any $n > 1$ it is highly unlikely that all word sequence events will be encountered in the training corpora, and many that do occur may only appear one or two times. A language model should not give any unseen event zero probability,[30] but without an infinite quantity of training text it is almost certain that there will be events it does not encounter during training, so various mechanisms have been developed to redistribute probability within the model such that these unseen events are given some non-zero probability.

As in equation 2.8, the maximum likelihood estimate of the probability of an event A occurring is defined by the number of times that event is observed, a, and the total number of samples in the training set of all observations, A, where $P(A) = \frac{a}{A}$. With this definition, events that do not occur in the training data are assigned zero probability since it will be the case that $a = 0$. [Katz 1987] suggests multiplying each observed count by a discount coefficient factor, d_a, which is dependent upon the number of times the event is observed, a, such that $a' = d_a \,.\, a$. Using this discounted occurrence count, the probability of an event that occurs a times now becomes $P_{\text{discount}}(A) = \frac{a'}{A}$. Different discounting schemes have been proposed that define the discount coefficient, d_a, in specific ways. The same discount coefficient is used for all events that occur the same number of times on the basis of the symmetry requirement that two events that occur with equal frequency, a, must have the same probability, p_a.

Defining c_a as the number of events that occur exactly a times such that $A = \sum_{a \geq 1} a \,.\, c_a$ it follows that the total amount of reserved mass, left over for distribution amongst the unseen events, is $\frac{1}{c_0}\left(1 - \frac{1}{A}\sum_{a \geq 1} d_a \,.\, c_a \,.\, a\right)$.

[30] If it did then from equation 2.6 it follows that the probability of any piece of text containing that event would also be zero, and would have infinite perplexity.

Discounting

In [Good 1953] a method of discounting maximum likelihood estimates was proposed[31] whereby the count of an event occuring a times is discounted with $d_a = (a+1)\frac{c_{a+1}}{a \,.\, c_a}$. A problem with this scheme, referred to as *Good-Turing* discounting, is that due to the count in the denominator it will fail if there is a case where $c_a = 0$ if there is any count $c_b > 0$ for $b > a$. Inevitably as a increases the count c_a will tend towards zero and for high a there are likely to be many such zero counts. A solution to this problem was proposed in [Katz 1987], which defines a cut-off value k at which counts a for $a > k$ are not discounted[32] – this is justified by considering these more frequently observed counts as reliable and therefore not needing to be discounted. Katz then describes a revised discount equation which preserves the same amount of mass for the unseen events:

$$d_a = \begin{cases} \frac{(a+1)\frac{c_{a+1}}{a \,.\, c_a} - (k+1)\frac{c_{k+1}}{c_1}}{1-(k+1)\frac{c_{k+1}}{c_1}} & : \quad 1 \leq a \leq k \\ 1 & : \quad a > k \end{cases} \tag{2.25}$$

This method is itself unstable, however – for example if $k.c_k > c_1$ then d_a will be negative for $1 \leq a \leq k$.

An alternative discounting method is *absolute* discounting, in which a constant value m is substracted from each count. The effect of this is that the events with the lowest counts are discounted relatively more than those with higher counts. The discount coefficient is defined as $d_a = \frac{a-m}{a}$

In *linear* discounting, event counts are discounted in proportion to their magnitude, thus d_a is constant over all values of a. In order to discount the same quantity of probability mass as the Good-Turing discounting scheme, d_a must be defined as $d_a = 1 - \frac{c_1}{A}$.

2.4.2 Smoothing probabilities

The above discounting schemes present various methods of redistributing probability mass from observed events to unseen events. Additionally, if events are infrequently observed then they can be smoothed with less precise but more frequently observed events.

[31] Two alternative methods of deriving the same estimate are given in [Nadas 1985].

[32] It is suggested that "$k = 5$ or so is a good choice" – [Katz 1987].

In [Katz 1987] a *back off* scheme is proposed and used alongside Good-Turing discounting. In this method probabilities are redistributed via the recursive utilisation of lower level conditional distributions. Given the *n*-gram case, if the *n*-tuple is not observed frequently enough in the training text then a probability based on the occurrence count of a shorter-context $(n-1)$-tuple is used instead – using the shorter context estimate is referred to as *backing off*. Katz defines a function $\hat{\beta}(w_{i-n+1}, \ldots w_{i-1})$ which represents the total probability of all the unseen events in a particular context. The probability mass $\hat{\beta}$ is then distributed amongst all the unseen w_i and the language model probability estimate becomes:

$$\hat{P}(w_i \mid w_{i-n+1}, \ldots, w_{i-1}) = \begin{cases} \alpha(w_{i-n+1}, \ldots, w_{i-1}) \cdot \hat{P}(w_i | w_{i-n+2}, \ldots, w_{i-1}) & : \quad c(w_{i-n+1}, \ldots, w_i) = 0 \\ d_{c(w_{i-n+1}, \ldots, w_i)} \cdot \frac{c(w_{i-n+1}, \ldots, w_i)}{c(w_{i-n+1}, \ldots, w_{i-1})} & : \quad 1 \leq c(w_{i-n+1}, \ldots, w_i) \leq k \\ \frac{c(w_{i-n+1}, \ldots, w_i)}{c(w_{i-n+1}, \ldots, w_{i-1})} & : \quad c(w_{i-n+1}, \ldots, w_i) > k \end{cases} \tag{2.26}$$

where $c(.)$ is the count of an event and:

$$\alpha(w_{i-n+1}, \ldots, w_{i-1}) = \frac{\hat{\beta}(w_{i-n+1}, \ldots, w_{i-1})}{\sum_{w_i : c(w_{i-n+1}, \ldots, w_i) = 0} \hat{P}(w_i | w_{i-n+2}, \ldots, w_{i-1})} \tag{2.27}$$

A back off scheme such as this can be implemented efficiently because all the back off weights α can be computed once and then stored as part of the language model, and through its recursive nature it is straightforward to incorporate within a language model.

With a back off scheme low count events can be discarded – *cut-off* – from the model and more frequently observed shorter-context estimates can be used instead. An additional advantage of discarding low occurrence events is that the model size can be substantially reduced, since in general as a decreases so the number of events c_a increases – in fact the Good-Turing discounting scheme depends upon this relationship. *Cut-offs* are used for all the language model experiments reported in this thesis, for example, with cut-offs of 1, 3 and 3 used for bigrams, trigrams and 4-grams respectively.

2.5 *n*-gram variants

2.5.1 Phrase-based models

The combination of word sequences into single vocabulary token entries in an *n*-gram model was proposed in [Jelinek 1990]. Advantages of such an approach include the

ability to gain additional context disambiguation of frequent phrases without changing the structure of the underlying *n*-gram model itself – that is, without having to change either the theory or the code used for the *n*-gram model – and the ability for phrases to be easily given specialist dictionary pronunciations. As an example, the phrase "going_to" could be given a second, alternative pronunciation closer to "gonna" than one based on the separate words "going" and "to".

In [Kuo and Reichl 1999], single token vocabulary entries are made by combining pairs of words together using an iterative greedy algorithm which attempts to maximise the training text unigram log likelihood, performing multiple groupings per iteration for speed. On a small test set they obtain bigram perplexity improvements but their trigram model worsens, whilst on a larger test set both models improve but only by small amounts. Using bigram and trigram likelihood measures to select the pairs actually decreased performance, rather than improving it. Based on a word error rate test they conclude that their method is only useful on small, restricted domains. In [Martin et al 1999], however, word phrases are found to provide both perplexity and word error rate reductions.

A class phrase model is built in [Ries et al 1996]. Words are first clustered into classes using the word exchange algorithm described in [Kneser and Ney 1993] – see section 2.3.1 – and then pairs of classes are joined together into single units in order to minimise the bigram perplexity. Only small perplexity improvements are reported and no word error rate gain. [Zitouni et al 1999] also convert phrases to single dictionary elements but use these with both word and class models – phrases are based on mutual pair information and are repeatedly merged until no improvement in perplexity is obtained, although unlike the other work described in this section they also allow pairs to be broken if they have become poorer in the light of pairings made since the pair was originally created. They get improvements which are particularly worthwhile with their class model, and interestingly they also compare with a variable-length model (see the following section) and conclude that their phrase model can also outperform that.

2.5.2 Variable-length *n*-grams

Although by definition *n*-gram language models store only *n*-tuples of words, models have been built which do not limit word sequences to a specific choice of *n* but instead

store different sequences with different lengths, the idea being to provide as much disambiguation as is possible given the statistical distribution of sequences in the training data – if some phrases are particularly well described by the training text then why throw away this information in the model by curtailing to an arbitrary *n*? The opposite argument can also be applied at the other extreme, thus methods such as cut-offs.

Variable length *n*-gram-like models have been given various names but are most commonly referred to as *varigrams*. In essence they can be considered to be *n*-gram models with suitably large *n* and a strong pruning policy which keeps only a small subset of all 'long' sequences encountered in the training text. In [Niesler and Woodland 1996/ICASSP] the part of speech class-based model, as described in section 2.3.2, is built using a variable-length *n*-gram model, which makes particular sense when modelling manually-assigned part of speech categories given the relatively small number of classes and the length of context required to provide good coverage of sentence structures. Contexts are extended on the basis of how much they improve the performance of the model. Variable-length *n*-grams with grammatically-associated categories are also used in [Siu and Ostendorf 2000] – see section 2.3.3.

[Kneser 1996] describes an alternative method of constructing a variable-length word *n*-gram model. Instead of extending existing *n*-grams, work begins by building a complete set of all 'long' *n*-grams found in the training corpus and then discarding those which contribute least to the performance of the model, assessed on the basis of pruning those contexts which give the most minimal increase in training set entropy. An approximation is used in order to make the computation more feasible, and it is assumed that when pruning shorter contexts that longer contexts which encompass that shorter context can also be removed. Pruning continues until a given number of *n*-grams results. Experiments were performed using a 64,000 word vocabulary and 240 million words of training text, with the maximum length of *n*-gram attempted set to a 5-gram and cut-offs of 1 applied to 3-, 4- and 5-grams before any of the entropy-assessed pruning took place. Results are not reported for 5-gram models, however, because it was found that the 5-gram models were not significantly different to the 4-gram models. Pruned 4-grams performed better than trigrams with cut-offs applied for the same number of parameters in terms of both perplexity and word error rate. Despite the monicker of 'variable context length', therefore, this work is similar to the entropy-based pruning described in [Stolcke 1998] or [Seymore and Rosenfeld 1996].

Other methods of extending the length of context modelled within an *n*-gram-like

framework have been proposed. In [Bonafonte and Mariño 1996, Bonafonte and Mariño 1998] *X-grams* are described, which are fundamentally *n*-grams with context-tied states constructed via the use of an explicit finite state automaton – it is the state-tying which allows longer sequences to be stored and to be provided with enough training data via this sharing, much like a class model. An alternative proposed method is named *multi-grams* in [Bimbot et al 1994, Deligne and Bimbot 1995], in which a probability estimate is based on the concatenation of independent variable-length sequences of words, and a word estimate is then calculated by summing the likelihood of each possible segmentation – by limiting the length of these individual segments whilst combining them together in a parent model longer contexts can be modelled.

2.5.3 Skipping models

[Siu and Ostendorf 1997] seek to extend the variable *n*-gram paradigm by providing support for discontinuities and repetition in spoken language by allowing some words to be skipped. They note that experiments have shown that skipping some words, such as disfleuncy markers like "uh", can help improve language model performance but that if this is blindly applied then it can actually damage performance.[33] They therefore propose a method based on adding context-specific skips which maximise the log likelihood on the training data set, using leaving-one-out deleted interpolation. In order to find these 'good' skips they store their *n*-gram counts within a tree structure with a separate sub-tree for each word under the master parent node, with each child of a node representing a possible preceding word of that node. This tree is constructed up to a depth of 5 words before various processes are applied. They begin by pruning those parts of the context which decrease the training set log likelihood performance of the model the least to achieve a similar effect to that in [Kneser 1996], as described in the previous section.

To search for skips, each node is compared with the sibling of its parent node which corresponds to the same word. If similarities are found then the two sub-trees headed by these nodes can be combined, which represents skipping the word of context represented by the parent. For example, when considering the part of the tree representing `THAT WAS UH YOU`, with `YOU` at the highest level, they hypothesise a link from the node for `WAS UH YOU` to the node for `WAS YOU`.[34] Once two nodes have been merged,

[33] See [Siu and Ostendorf 1996] or [Stolcke and Shriberg 1996].

[34] For comparison, when searching for contexts to prune they compare the node for `WAS UH YOU` with

the algorithm iterates until some threshold is reached. Unfortunately, in terms of both perplexity and word error rate no worthwhile gain is obtained by adding skips – the variable *n*-gram and variable *n*-gram with skip results reported are perplexities of 77.5 and 77.4 and word error rates of 35.30% and 35.29% respectively.

2.6 Further statistical language models

2.6.1 Structured language models

[Ries et al 1995] report perplexity reductions of 30% using a structured phrase model. Their algorithm works by placing words which are found with the same context for the preceding three words and following single word into equivalence classes, whilst all remaining words are binned into a single class. Having decided the class of each word, sequences of classes are then selected from the training corpus via a ranking system which makes decisions on the basis of the mutual information between a prefix and the rest of the sequence. Via an iterative process, class sequences can be subsumed in their entirety into other sequences, and then the occurrence of each sequence is modelled with a unigram probability. Using this method worthwhile word error rate reductions relative to a trigram word model are obtained, although this is on a very restricted test. Later results showed that these gains could not be reproduced on more general domains.[35]

In [Jang and Hauptmann 1998] a language model is proposed which has the ability to learn hierarchical statistical rules without supervision. They replace the most frequent word sequences with single token entries that represent either the ordered sequence or the constituents in any order, then iterate to build a hierarchical structure that can replace any sentence with a single token. Alternatively the construction of the model can be relaxed to allow a sentence to consist of a sequence of tokens. The overall effect, however, is to construct a parse tree of each sentence. By restricting the rules to combining only two words or tokens they find that a more 'traditional' parse results. When using their model to rescore *N*-best lists they report a significant reduction in word error rate over a trigram baseline.

that for UH YOU and merge them or not on the basis of the log likelihood test; to implement the skipping case, exactly the same comparison is made but it is with the sibling of the parent as well as the parent itself.

[35] See for example the first paragraph of [Ries et al 1996], later work from two of the same authors.

Work reported in [Chelba and Jelinek 1998, Chelba and Jelinek 2000] describes a structured language model which uses partial parses of sentences to allow use in a left-to-right manner based on a complete set of all possible parses up to the word being predicted. The model assigns a probability for each possible parse and part of speech class assignment for a given sentence or partial sentence, with classes and binary parse rules learnt from a manually-tagged corpus such as the Penn treebank [Marcus et al 1995]. After initial training model parameters are reestimated using the EM algorithm [Dempster et al 1977] in order to minimise the training text perplexity.

In this structured model, prediction of a word is based both on its part of speech and the two most recently exposed headwords, which includes special start and end of sentence tokens. With each new word the binary parse tree structure is grown in all possible ways with headwords being propagated up the tree as defined by each rule,[36] with this performed in a left-to-right manner with partial parses as described above in order to allow the model to be used in a recognition system without having to be restricted to rescoring complete sentences. [Chelba and Jelinek 1999] reports a small improvement in word error rate when interpolating this structured model with a word trigram model. In [Goodman 2000, Goodman 2001/MS], however, it is suggested that other existing *n*-gram-based models are capable of capturing similar information and can perform just as well, so assessing the structured language model solely relative to a trigram baseline is potentially misleading.

2.6.2 Longer-length predictions

The choice of *n* in *n*-gram models is limited due to reasons of data sparsity even when using a variable-length model. Despite the limited history length of each individual context equivalence class, however, the accuracy with which *n*-gram models can estimate the likelihood of word sequences longer than their length is surprisingly good – for example they have been found to model sentence length well [Rosenfeld 1999] despite possible intuition otherwise [Zhu et al 1999]. None the less they are clearly limited in some respects in the longer-distance information they can represent so various methods of modelling such correlations have been proposed, including *distance n-grams* and *trigger* models.

[36] In practice, however, pruning is applied because the state space of the model will otherwise become too large – this pruning retains the most likely parses to date.

Distance *n*-grams

Possible methods of avoiding data sparsity issues for longer context lengths include *n*-grams with skips, as previously discussed,[37] and distance *n*-grams, in which an explicit gap is encoded directly into the *n*-gram.

[Huang et al 1993] examine the performance of a language model which uses distance bigrams, where a distance-*d* bigram is defined as a bigram which predicts word w_i based on the preceding word w_{i-d}, so given distance bigrams up to distance k the probability of a word w_i can is modelled as:

$$P(w_i \mid w_{i-k}, \ldots, w_{i-1}) = \sum_{d=1}^{k} \lambda_d . P(w_i \mid w_{i-d}) \tag{2.28}$$

where each λ is a weight. The weights can be optimised on some withheld data using the EM algorithm.

Models were built for values of $d = 1, \ldots, 10$ and an additional $d = 1000$ model which was used as a control, based on the assumption that no mutual information would exist at a word separation of that distance. The training set perplexity of each separate distance bigram model was calculated individually so as to obtain a measure of the average mutual information between word w_i and word w_{i-d}. The perplexity was found to be low for $d = 1$ but to increase significantly for higher values of d. Once d reached 6 and above, however, the perplexity remained more or less constant, although they note that it was consistently lower than that for $d = 1000$, suggesting that some mutual information does still exist at such separations. Their conclusion, however, was that to make use of this small amount of mutual information at distances too long to be captured by conventional *n*-grams would require a more sensitive model. Their results suggest that even with huge amounts of training data a sensible bound on the choice of *n* in *n*-grams will always be 6 or 7.

In [Martin et al 1999] distance trigrams of the form $P(w_i \mid w_{i-1}, w_{i-3})$ and $P(w_i \mid w_{i-2}, w_{i-3})$ are trained and then used with a standard trigram by interpolation:

$$\begin{aligned} P(w_i \mid w_{i-1}, w_{i-2}, w_{i-3}) \quad = \quad & \lambda_{t1} . P_{t1}(w_i \mid w_{i-1}, w_{i-3}) \; + \; \lambda_{t2} . P_{t2}(w_i \mid w_{i-2}, w_{i-3}) \\ & + (1 - \lambda_{t1} - \lambda_{t2}) . P_{t3}(w_i \mid w_{i-1}, w_{i-2}) \end{aligned} \tag{2.29}$$

[37] Although no work which examines the use of skips with high values of *n*, such as $n > 5$, has been reported.

for some model weights, λ. It is found that incorporating these distance trigrams into the language model gives significant word error rate reductions over a word trigram baseline, and furthermore that on a small training corpus they also perform better than a word 4-gram model. No 4-gram comparison is given on a larger corpus, however, and whilst it seems reasonable to suppose that what are effectively 4-grams with a word missing will not outperform well-trained 4-grams, they have an intrinsic generalising ability that 4-grams do not have and require less training data.

Distance bigrams are linearly and log-linearly interpolated with a trigram word model in [Klakow 1998], where when used at distances of 1, 2 and 3 they are found to provide perplexity gains over a trigram system – but none of the models out-perform a word 4-gram baseline.

Trigger models

In [Rosenfeld 1994] it is argued that "long-distance *n*-grams are seriously deficient", since although distance bigrams capture correlations between words that are separated by distance *d* they have no ability to generalise to other distances. Given the limits of training data and to a much lesser extent the practicalities of storage space this is a major drawback of such models. It is proposed, therefore, that word pairs be modelled using a system which does not directly specify the distance between two words. Such a method is then proposed through the definition of word *trigger* constraints within the Maximum Entropy framework[38] whereby the occurrence of a *trigger* word in the recent history increases the likelihood of another word, commonly referred to as the *primed* or *target* word. Class triggers are also tested, whereby the presence of one class increases the likelihood of another class later on in the word sequence. The experimental performance of word and class triggers, however, is disappointing. Assessed on the basis of perplexity in [Rosenfeld 1994], they are found to add little information that is not already included in distance-2 bigrams.[39] Class self-triggers are also found not to perform as well as might be hoped, which it is suggested is due to the lack of purity in the classes.[40]

[Niesler and Woodland 1997] describe a trigger model built into a part of speech variable-length *n*-gram model where word and class pairs are used as triggers to modify the

[38] See section 2.7.3.

[39] And therefore, it is reasonable to assume, in well-trained 3-grams.

[40] The classes often contain one likely word and various far less likely words.

likelihood of words within classes. This is motivated by the observation that the probabilities of words within classes should be varied dependent upon the preceding text. In the proposed model the effect of each observed trigger exponentially decays with distance towards a constant probability, where this constant probability is estimated from the most distant pair separations observed in the training text. It is noted, however, that once the separation of two words increases past a certain point that their influence damages performance, so a cut-off point is empirically chosen.

In general it would be extremely slow to examine all possible trigger-prime pairings in detail since there are $|\mathbb{W}|^2$ different possibilities, so a two-pass selection strategy is described in [Niesler and Woodland 1997] whereby promising candidates are found using a fast method which prevents the number of potential pairs becoming too large, and then a pruning stage which reconsiders each of the short-listed pairs in more detail is performed.

The test set perplexity of this trigger varigram class model is found to be much worse than that of a trigram word model in each experiment reported, which are performed on the LOB, Switchboard and Wall Street Journal corpora. The sole exception to this is that when trained on the LOB corpus it is found that using self-triggers in the language model can give a very small gain from 413.1 to 412.2 – adding other triggers to the self-triggers has no effect at all on this reported perplexity. The triggers do improve the performance of the basic varigram class model, however.

Most other work on trigger models also notes that by far the biggest perplexity gain is obtained via the use of self-triggers – word error rate improvements are rarely reported. Excluding all except these self-triggers from the model results in what is often specifically referred to as a cache language model.

2.6.3 Cache models

A cache language model increases the probability of recently seen words on the basis that having used a particular word a speaker is then more likely to use that word again, perhaps because it is local to a particular topic or maybe because it is a word they have a tendency to use. Like any adaptive method, however, this implicit assumption about the current topic or speaker can break down if either changes, so it is also possible for the model to damage performance. In addition, if a word has already been correctly recognised then it may already be well modelled without needing to increase its

language model probability further, or alternatively if it has been mis-recognised then the error may be compounded by increasing its probability, which could be especially damaging if there is already an initial propensity to the mistake.

In [Kuhn and De Mori 1990, Kuhn and De Mori 1992] a cache model is described whereby word probabilities within each of 19 part-of-speech classes were dynamically adapted on the basis of how many times they had previously been seen in the recent history. Instead of using a hard history cut-off for the cache, in [Clarkson and Robinson 1997] an exponential decay to zero is examined[41] and found to give a small improvement in the performance of a word unigram cache when assessed on the basis of perplexity. Unfortunately in [Clarkson and Robinson 1998] it is discovered that this same cache model gives no gain in word error rate over a baseline trigram word *n*-gram except when using a mismatched training corpus – in fact a small degradation in performance is noted.

[Jelinek et al 1991] use not just a unigram cache but also bigram and trigram caches, with the interpolation parameters found via the EM algorithm. They do not flush their cache between documents and they obtain word error rate improvements which are invariant to whether they update their cache between documents or sentences. However, the reported results, which include a relative 6.1% word error rate reduction by the trigram cache – itself superior to their unigram cache – over an unstated baseline, having seen just 100 words of text, appear to have been unique to the particular experimental setup.

2.7 Combining language models

Different language models can capture different dependencies so given multiple models the question arises of how to best combine the advantages of each individual model to produce a better composite model.

2.7.1 Linear interpolation

Linear interpolation is a popular method for combining totally disparate language models. It is straightforward to implement and can be used to combine any number

[41] Which it is argued can therefore be considered to be infinite in size.

of models. An additional potential advantage is that it requires only a single implementation which can then be applied to any set of component models. It also does not require any additional computational time to prepare the combined model over and above that already used to build the component models except for that used to choose the combination weights, which can be estimated efficiently, and it is not over-sensitive to the choice of those weights.

Linearly interpolated models are each assigned a weight which is normalised such that the total weight of all the models is always 1:

$$P_{\text{linear}}(w_i \mid w_1, \ldots, w_{i-1}) = \sum_{m=1}^{|\mathbb{M}|} \lambda_m P_{M_m}(w_i | w_1, \ldots, w_{i-1}) \tag{2.30}$$

where $\mathbb{M}$ is a set of language models $M_1 \ldots M_{|\mathbb{M}|}$ and $\sum_{m=1}^{|\mathbb{M}|} \lambda_m = 1$.

The set of weights λ can be changed at any point during recognition so long as the constraint that they sum to unity is maintained, although in practice they are usually set on a per-experiment basis. The weights can be optimised on some given text by using the expectation-maximisation (EM) algorithm [Dempster et al 1977] which is guaranteed to find the weights for the best-possible perplexity on the text which it is optimised on, if run to convergence.

Despite its simplicity, linear interpolation is often remarkably effective. The fact that it bounds the performance of the combined models is both an advantage and disadvantage – it can never return a lower probability than the worst-performing model for a particular word, but the same is true of the best-performing probability. Even if all models attach a similarly high or low probability to a word then linear interpolation cannot interpret this as a vote of confidence and return an even-higher or even-lower overall probability. This limitation provides a high degree of stability and makes the model tolerant to suboptimal interpolation weight settings – often model weights can vary by as much as 10-20% of the total weight with only a small effect on perplexity.

Figure 2.1 compares the difference in per-word performance between a 4-gram word model and a 4-gram word model linearly interpolated with a 4-gram 1003-class model with weights of 0.65 and 0.35 respectively.[42] For each word in the test set the change in log probability when using the combined model relative to the stand-alone model was

[42]The models used are infact the models that will later be described in chapter 4, and the test set is the set of evaluation lattice transcripts also described therein.

calculated – the histogram shows the quantity of changes in probability of different magnitudes.

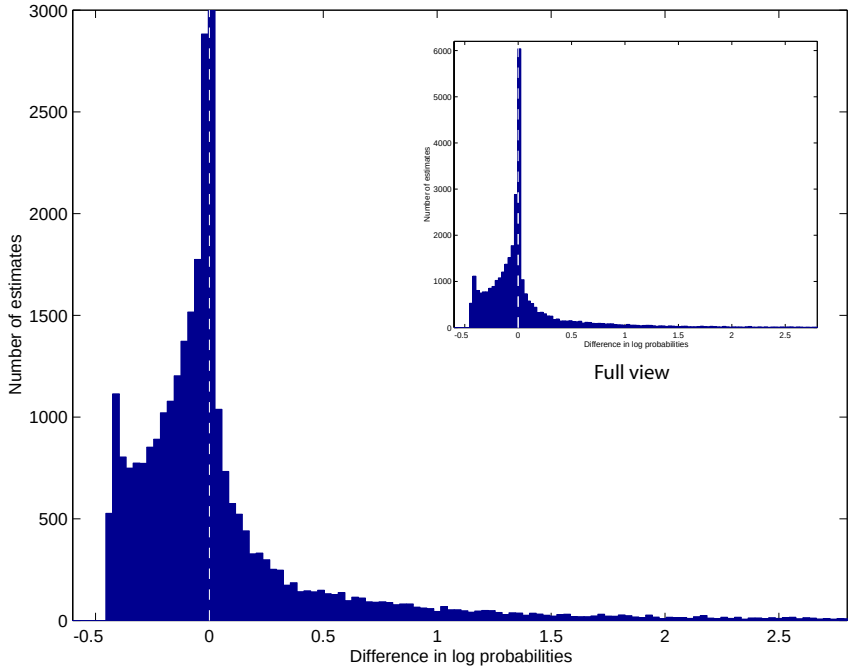


Figure 2.1: Change in log probability of each word in a test set when interpolating a word model with a class model relative to the word model alone – number of changes of various magnitudes

The figure has some interesting features. It is clear how the decrease in probability from the word model is bounded. Denoting the weight of the word model as α (in this case $\alpha = 0.65$), and denoting the models as P_w and P_c for the word and class models respectively, then the change in log probability of any given word x is:

$$\begin{aligned} & \log\left(\alpha\, P_w(x) \;+\; (1-\alpha)\, P_c(x)\right) \;-\; \log(P_w(x)) \\ = \;& \log\left(\alpha \;+\; (1-\alpha)\frac{P_c(x)}{P_w(x)}\right) \end{aligned}$$

Noting that $\frac{P_c(x)}{P_w(x)}$ is guaranteed to be positive since both $P_c(x)$ and $P_w(x)$ are always positive then the minimum value that can be returned by this expression is $\log(\alpha)$ – so in the example here this minimum value is -0.43, which corresponds with the most negative change in log probability value observed experimentally in figure 2.1. The figure shows graphically how the overall model can never be much worse but can

sometimes be many times better – even though a lot of counts are worse because the stand-alone class model is notably worse than the stand-alone word model, the lower probabilities are not low enough to be able to dominate the average.[43]

As described above, linear interpolation is in one respect limited since it has no concept of the confidence of a particular probability returned by a component model. If one model has based a probability on only a few training corpus observations whilst another has derived it from more observations then perhaps that second model is more accurate – linear interpolation cannot inherently use this information. An alternative algorithm could be developed to use this information to adapt the weights, however, although little has been published in this area to date, perhaps because of the *ad hoc* nature of any such experiments. One exception, however, is found in [Kobayashi and Kobayashi 1999] – see section 2.7.4.

For interpolation of models to give gains the models must have a degree of orthogonality – the closer to orthogonal they are then the greater the potential gains. Thus a word *n*-gram interpolated with a model using both words and classes might be expected to give a worse overall performance than a word *n*-gram interpolated with a class-only *n*-gram, even though the individual performance of the class-and-word *n*-gram might well surpass that of the class-only *n*-gram. This is because the class-only model is more orthogonal to the word model than the combined class-and-word model is likely to be – the combined disambiguating power of the models is stronger if the models are 'more different' in some way.

2.7.2 Backing off

An alternative method of taking advantage of the varying degree of confidence attached to a predicted word sequence by each of a set of models is to implement a backing off scheme, whereby one particular language model normally takes precedence but in circumstances where it has a lower confidence it delegates to another, usually less-specific, model. Sometimes that model in turn can back off to another model, and so on. A common example of this is to back off from an $(n+1)$-gram to an n-gram. Alternatively, all available models can be ranked for each probability query and a particular model picked on the basis of a confidence assessment, such as the quantity of training set observations used to compute a specific probability. When backing off care must

[43] The word-only perplexity was 174.84 whilst the combined model had a perplexity of 160.50.

be taken to ensure that the sum of the probabilities for all potential predicted words is unity.

An example of a model backing off to one with different dependencies is given in [Niesler and Woodland 1996/ICSLP], in which a word *n*-gram backs off to a non-deterministic POS class-based variable-length *n*-gram. On smaller training sets this back off model is found to perform significantly better than a baseline trigram word model, but on larger corpora no gain is observed unless the number of parameters in the model is kept deliberately low. Similarly in [Miller and Alleva 1996] a bigram word model is described which backs off not to a unigram word model but to a 256 class bigram model. Their resulting model gives a worthwhile improvement in word error rate over their baseline unigram back off version, and they also report that using this method they can build a model which has the same word error rate as their baseline but uses only two thirds of the number of parameters. In general it appears that backing off from words to classes allows improved performance when using small training sets or when smaller language models are required.

2.7.3 Maximum entropy

An alternative method of combining knowledge sources is proposed in [Rosenfeld 1994], whereby a language model is built by combining a set of features in one model rather than by interpolating together a separate model for each feature. Each knowledge source is defined in terms of a set of constraints which are imposed on the model.

Consider two example language models, each computing $P(w \mid h)$ for a history h. One estimates a bigram probability and another a trigger-pair probability:

$$P_{\text{bigram}}(\text{EXAMPLE} \mid \text{AN}) = k_{\text{EXAMPLE,AN}}$$
$$P_{\text{trigger}}(\text{EXAMPLE} \mid \text{INTRODUCE} \in h) = k_{\text{EXAMPLE,INTRODUCE}\in h}$$

When the two models are linearly interpolated the probability of each of these estimates has already been optimised globally across the training set, and so the two can only be averaged together – there is no way of taking explicit advantage of the fact that *both* the preceding word ‘AN’ and the trigger ‘INTRODUCE’ have been seen. In the Maximum Entropy framework, however, there is no such restriction.

A Maximum Entropy (ME) model consists of a set of constraints which enforce properties on the model. The bigram and trigger probability functions given above can be seen

to be constrained to always return a value, *k*, which is directly dependent on their input parameters. In an ME model, however, these constraints are relaxed so as to define *average* rather than fixed results. Given $P_{\text{bigram}}(\text{EXAMPLE} \mid \text{AN})$, for example, the ME model constraint would be that the *expectation* of this function would be $k_{\text{EXAMPLE,AN}}$ – therefore it is not necessarily the case that this value is returned in every instance. Given a set of constraints there is either an infinite or an empty set of functions which will satisfy these expectations. In practice the former case is assumed due to the vast range of possible functions. In order to assume nothing more than is specified in the constraints, a search is made for the function in that set which has the highest entropy – which can be visualised as the "flattest" of those functions – thus the titular *maximum entropy*. See [Rosenfeld 1994] for full details.

The drawback of Maximum Entropy language modelling is that even though the constraints can be defined in terms of the limited range of histories seen in the training text, as in the bigram example above,[44] the search space is still huge and optimising the parameters requires a large amount of time and storage – these can, however, be traded off for a less accurate model. Some work has been done in the area of optimising the search. [Goodman 2001/ICASSP], for example, provides a hierarchical class algorithm which allows a faster search for *n*-gram ME language model parameters and claims a 35 times speed increase, but it does depend on an approximation and can be slowed when additional constraints are added, such as a unigram cache. Other reported work in this area, such as that in [Pietra et al 1995] or [Lafferty and Suhm 1996], similarly does not provide gains for all types of constraints. [Wu and Khudanpur 2000], however, claim that the hierarchical speed-up scheme they describe can be usefully applied to any ME model.

Linguistic models

Linguistic features are used within the Maximum Entropy framework in [Zhu et al 1999]. A shallow parser is used to classify each word into one of 110 separate categories, based on parts of speech plus a few singleton classes reserved for function words. Three types of feature are used – sequences which exactly match categories, sequences which are constituents of a phrase, and constituent trigram features which select if a sentence contains a specified class triplet. Training is performed by parsing text and counting features on both an actual corpus and an artificial corpus generated

[44] As opposed to across all possible histories.

from a traditional trigram model – features which differ in count significantly between the two corpora are chosen to be modelled by the ME model. Experimental tests on the resulting model show a small improvement in perplexity and an insignificant decrease in word error rate over a trigram word model.

2.7.4 Combined class and word models

[Kobayashi and Kobayashi 1999] propose a combined class and word *n*-gram language model which uses a form of linear interpolation. For each word probability prediction they use a word and class linear interpolation weight derived via an exponential function of the number of times the word history sequence was observed in the training text. They suggest this model could be used when there is not enough data to train a full word language model sufficiently. Unfortunately, the reported work does not compare the model with a standard linearly interpolated model with fixed weights, although their results show that they do improve on the recognition performance of a standard word model. Another *ad hoc* combination method is proposed for use in speech recognition in [Smaïli et al 1999], in which a French part of speech class model is used first to predict the class of a following word so as to produce a list of candidate words which can then be assessed by the recogniser and word *n*-gram language model. Unfortunately the results are not encouraging.

2.7.5 Other combination methods

In principle, models can be combined in an infinite number of ways by picking and choosing different probability predictions from different models, although in practice the general necessity to ensure that the sum of all probabilities for a given context is 1 does restrict or at least complicate matters – in particular, perplexity clearly becomes meaningless as a measure of language model performance if this constraint is not held, although in general any scoring system could be used when predicting the most likely word in a given context at recognition time. Two such *ad hoc* combination methods will be described in section 2.8.2 below – [Ueberla 1997] and [Seymore et al 1998 / ICSLP].

2.8 Adaptive language models

Language models can be built which adapt to the most recently seen text in various ways. The precise definition of an *adaptive language model*, however, is slightly unclear. At one level an adaptive model is surely one which takes any account of its context, on which basis a unigram is not whilst a bigram is, but perhaps a more useful definition would be that any model which changes its stored parameters whilst in use is an adaptive model. But even this definition has problems – trigger models can either explicitly modify stored probability values on the basis of seen words, like the cache subset might reasonably be expected to do, or they might simply scan over the recent history at each instance and function like a distance bigram. So can a model be termed adaptive or not simply on the basis of its implementation, even if the underlying behaviour remains constant? Perhaps, but maybe a more general description of an adaptive language model would be any model which attempts to model on the basis of more than a single sentence or partial sentence of history.

2.8.1 Mixture models

Cache and trigger models both provide methods of language model adaptation, as do *mixture* models. A mixture model is any system consisting of multiple component models combined in some way, typically via linear interpolation; see, for example, [Kneser and Steinbiss 1993]. Each model has a weight associated with it and so the individual weights of each of those component models can be adjusted in some way so as to adapt to the recently seen text at some defined granularity. This can be used to implicitly or explicitly react to features such as the style of text or to encapsulate some notion of topic.

A more detailed examination of adaptive mixture models will be made in the following chapter, but it is worth noting here that none of the component models are modified when changing the weights so the only adaptation expense is to actually calculate those weights. For linear interpolation the EM algorithm provides a simple and fast method of finding weights, thus facilitating feasible dynamic adaptation.

2.8.2 Other methods

In [Ueberla 1997], language model adaptation is performed by linearly interpolating in-domain and general models but only for history events encountered in the domain-specific text; for those not encountered the non-interpolated general model is used alone. Using statistically-clustered class *n*-gram language models a small improvement on in-domain text is reported but their model falls short of the performance of a stand-alone in-domain model when that model has a reasonable amount of training data available to it, quantified in their work as "several tens of thousands of words".

An alternative adaptation scheme is the topic model combination method described in [Seymore et al 1998/ICSLP], in which the vocabulary is divided up into on-topic, off-topic and general word subsets using manually-defined topic sets. Each recognition hypothesis is matched to the five most similar topic clusters using unigram word models, and then the text in those clusters is divided into on-topic and off-topic sets. Two word *n*-gram models are then built, one predicting the on-topic and the other predicting the off-topic words, and these are then combined with a general *n*-gram model which predicts the remaining words – each word is then predicted via either the on-topic or off-topic models if it is in their prediction vocabulary, or the general model otherwise. The resulting model improves on the performance of the stand-alone general model, but does not work as well as standard linear interpolation between separate topic and general word models.

2.9 Conclusions

Various methods of language modelling have been described, with most attention paid to *n*-gram models and their variants. Reported improvements have been mentioned where appropriate and so it might be reasonable to conclude from this chapter that there are a range of new types of language model which perform better than existing models, or that adding some extra distinguishing power to an existing model is pretty much guaranteed to obtain a gain. Unfortunately, as is also concluded in [Goodman 2001/MS], this would be to draw a mistaken inference. The issue that should be considered is that it is the exception rather than the rule to find reported language modelling work which compares a proposed new model with a *fair baseline*. But what, then, is a fair baseline?

2.9.1 A Fair Baseline

It is always possible to find a model which performs worse than any other given model on a particular test set, unless the model in question assigns precisely zero probability to the text. The corollary of this observation is that it is easier to achieve gains if comparison is made with a poorer model. Gains can vanish if compared with a better model for two reasons: firstly, a better model might already incorporate the modelling power that the claimed improvement provides, for example distance-3 bigrams can be represented precisely by enough 4-grams but not by any quantity of 3-grams; and secondly, a better baseline might make any actual improvements that the new model does add become statistically insignificant.

3-grams versus 4-grams

Consider the state of the art systems described in 1997, as reported in [DARPA BN 1998]. Most of these systems used at least a 4-gram word model [Woodland et al 1998, Cook and Robinson 1998, Seymore et al 1998/BN, Chen (S.S.) et al 1998, Sanker et al 1998], and many also added in additional language modelling components such as caches, class models or forms of topic adaptation or mixture models. Even without considering any of these additional components or refinements it is still hard to ignore the fact that for some time a 4-gram word model has been a sensible minimum baseline when claiming improvements to a contemporary speech recognition system. Even if the aim of a new language model is to be particularly compact in some way, 4-grams might still be suitable as a comparison through use of pruning.

Size of training corpus

In the same way that a 3-gram model has less power than a 4-gram model and so is easier to improve on, it is also easier to improve on a *less well-trained* model, which in practice means a model which has been trained on a relatively small amount of text. It is easier because the model is likely to be less general, so any new model which behaves in a different way might happen to model the idiosyncrasies of a specific test set better, even though in an alternative or more thorough assessment this might not be the case. A good example of this is the comparison between a class *n*-gram and a word *n*-gram model – with a very small amount of training data class models can outperform word

models, but in most experiments the word model performance is significantly better than that of the class model.

Perplexity assessment

Perplexity is a fair method of assessing a language model, but it does not share a linear correlation with word error rate and so by itself can sometimes be rather misleading. It is frequently found that small or moderate improvements in perplexity can lead to no significant improvement in word error rate when a language model is tested on a full recognition system. This effect is exacerbated in some cases – a cache model, for example, aims to improve the future prediction of words which have already been successfully recognised by the model, so many perplexity gains from these improved predictions might well be from words which are already accurately recognised and therefore lead to no word error rate reduction. Perplexity takes no account of acoustic confusability. [Clarkson and Robinson 1998] note that "A reduction in perplexity will not ... translate into an improvement in recognition performance if it comes about simply by boosting the probabilities of words which were correctly recognised in the first pass", concluding that "even fairly large reductions in perplexity are no guarantee of a reduction in word error rate."

2.9.2 Choosing a Baseline

An 'improvement' has to be relative to some baseline, so it is easy to obfuscate the true performance of a language model by comparing with a relatively weak baseline. Given the previous discussion it is reasonable to argue that in most cases a fair baseline should consist of a 4-gram word model. If a new model is then assessed by interpolating it in some way with that 4-gram then in turn a fair baseline for this interpolated model is to contrast it with the performance of a 4-gram word model interpolated with a 4-gram class model, since such combined models have been well described in the literature and are known to perform well. Suitable pruning can be applied if necessary to reduce the number of parameters.

Empirically it appears to be a difficult task to improve significantly on the performance of a 4-gram word model, especially one which has been interpolated with a 4-gram class model. This may give a clue as to why such models are rarely chosen as baselines in language modelling work. On the subject of models which are assessed purely on

the basis of a trigram baseline, [Goodman 2001/MS] has this to say: “The result is yet another way to beat trigrams, assuming we don’t care about speed or space. Whether these new techniques are better than the previous ones, or can be combined with them for larger improvements, or offer a better speed or space tradeoff than other techniques, is only very rarely explored.”

2.9.3 Final conclusions

Word and class *n*-gram models have been introduced, and various methods of finding suitable classes and building robust *n*-gram models have been described. Variants on word and class *n*-grams have been examined, but few of the reported results suggest that these can improve significantly on a state of the art system. Structured models perhaps hold some hope, but as of yet no breakthrough has been made in this area and there remain many unsolved problems. Longer-distance models such as triggers, caches and distance *n*-grams have also failed to achieve worthwhile improvements. Various methods of combining information sources have also been described, and although maximum entropy models appear attractive they remain slow to train given the large number of parameters – therefore linear interpolation of separate models remains popular.

In summary, it is clearly a difficult task to improve on the performance of a linearly interpolated word and class *n*-gram model. Improved adaptation methods and better class models are two of the more promising areas to explore. The remainder of this thesis will seek to combine these two promising areas and will thus examine methods of class model adaptation. As a prelude to the research into a topic-adaptive class model presented in chapter four, the following chapter proceeds to present a detailed study of the statistical creation of topic training corpora suitable for use in an adaptive language model.

3. Topic Corpora

Chapter two ended with the conclusion that adaptive language models, and adaptive class based language models in particular, are promising areas to investigate. Existing adaptive models, as discussed in section 2.8, often take advantage of multiple training corpora from different domains in order to construct a set of distinct estimates for each domain. These are then dynamically weighted and combined when calculating model probabilities. This chapter addresses the issues involved in constructing the domain – or *topic* – corpora necessary for training such adaptive language models.

3.1 Introduction

Adaptive language models provide a feasible method for modelling long range contexts. Other models such as *n*-grams are typically limited in the length of the context that they can model by the restrictions on the amount of training data that they have access to. The practicalities of finite storage space can also add restrictions although suitable pruning can lessen these. Adding just one extra word to the length of the considered context will typically increase the size of the model's state space by many orders of magnitude, so this is not a practical method for modelling long range dependencies. Other longer range models such as distance *n*-grams and triggers were discussed in the previous chapter but as of yet none of these has had a significant impact on word recognition error rates.

In a mixture model multiple sets of language models are combined, often via linear interpolation, with each model given its own weight. In an adaptive system these mixture weights are dynamically adjusted in order to attempt to optimise the model performance to fit some specific target text. Typically each component model will have been trained to try and encapsulate a different 'type' of text than all the others.[1] Longer-

[1] Although mixtures of multiple model types can also be used to model different dependencies within text.

range adaptation can then be implicitly implemented by using a mixture of shorter-range models, such as a set of *n*-grams, through use of dynamically-adjusted model weights which react to long-range context.

The question arises of what constitutes a 'type' of text, as it was loosely described in the previous paragraph. At the lowest level, the two sets of text used to train any pair of component models must be different, or otherwise the resulting language models will be identical and nothing can be gained by combining them.[2] The question must therefore become 'how different must the text be?' – or more formally, 'how can a sublanguage be defined?' – and then in turn 'how is suitable text found?'.[3]

3.2 Overview

The task of creating training text corpora suitable for training each component of an adaptive topic model can be broken down into two stages: the segmentation of text into articles, and the clustering of those articles into coherent groupings.

3.2.1 Segmentation

Various methods can be used to delineate text into distinct segments. Some manually-tagged notion of changes in the current topic could be used, or phrase, sentence or discourse boundaries are also suitable. Text can be segmented after a given number of words or a given amount of time has passed in an audio transcription, or at each pause or change in acoustic environment. Each change in speaker perhaps suggests a new style, or there could be inherent boundaries within particular text, such as chapter, programme or article divisions. In short, there are many potential intrinsic boundaries that could be used to segment text and even more could be formed through combining these assessment schemes.

Boundaries in text can also be hypothesised on the basis of statistical analysis of text through some defined measure of distance – see, for example, [Bigi et al 1998] or [Dharanipragada et al 1999]. Automatic text segmentation is currently an active research

[2] Assuming the same type of model is being used for each component.

[3] A formal description of the issues involved in defining sublanguages can be found in chapter 3 of [Sekine 1998].

area, however, and in work such as [Yamron et al 1999] and [van Mulbregt et al 1999] it is concluded that the automatic text segmentation they are able to implement is inferior to the use of manually-defined segments, such as through application of one of the metrics described in the previous paragraph.

3.2.2 Clustering

Given segment boundaries on some text, that text can be split up into separate *articles*, defined by successive segment boundaries. Each of these articles can then be treated as a miniature text corpus of its own. Alternatively each article can be joined with any other selection of articles to form a larger corpus. Deciding whether to group particular articles or article groups together will typically depend on some definition of *similarity*. Such measures of similarity could be based on comparison of manually assigned article tags or they could use some statistical metric.

In practice, given a number of separate articles it is usually desirable to group these in some way in order to allow better language model estimates to be made from the larger amount of resulting text in each combined grouping, although this of course does depend on the definition and size of article in use. If each is in fact a single sentence, as for example in [Kneser and Steinbiss 1993], then clustering is essential in order to obtain statistically meaningful information about each topic, whilst if the articles are full stories from a newspaper or programmes from broadcast news then in some instances they may not require merging.

Each selection of grouped articles can for convenience of notation be referred to as a separate *topic*, even if grouped on some non-semantic basis. In general having fewer topics not only improves the quantity of training data associated with each topic but also reduces the overall model storage space requirements and decreases the amount of run-time processing that is necessary when calculating topic weights. Given m articles two extremes can be defined: without any grouping of articles it can be considered that corpora for m topics are available, whilst grouping them all into one single topic will allow a standard non-mixture language model to be built.

Any given article might not be similar to any other article or group, or conversely it might fit well with multiple topics. Whether to discard the article in the former case or to add it to multiple topic groups in the latter are both questions which must be answered when defining topic corpora. If hypothesising article boundaries then a

similar issue can apply at a lower level – should overlapping articles be allowed? Other issues include how many topics should be constructed, or alternatively how different must each topic be from all the other topics?

Articles are merged in an agglomerative tree-like combination process to form topic sets in, for example, [Donnelly et al 1999]. Even once topic sets have been constructed the distribution of articles amongst those initial groupings may of course still be modified – in [Walls et al 1999], for example, a *k*-means process was used to iteratively redistribute articles amongst topic sets. [Carlson 1996] concludes that agglomerative clustering works well in a test which uses 100 articles to form 10 topics, obtaining 97% correlation when compared with manually-chosen topic assignments. A more detailed examination of current work will be made in section 3.3 below.

3.2.3 Practical considerations

Various restrictions on the practical definition of a topic are inherently imposed in the same way that the choice of *n* is in effect limited in an *n*-gram language model. Firstly, each topic must be associated with enough text that a reasonable estimate of its properties can be made for the model in question. Secondly, article boundaries and topic similarity metrics are clearly also limited to some extent by what information is available to act upon – segmenting text on the basis of speaker would be difficult to achieve accurately without existing speaker labelling, for example.

3.3 Previous work

Existing work in the field of topic-adaptive modelling uses a range of methods from those discussed above and typically – but not always – employs linear interpolation of topic-specific *n*-gram models.

In [Kneser and Steinbiss 1993] a total of 15 topics are formed from different text categories such as newspaper text, scientific text and fiction. A separate bigram model is then constructed for each topic and linear interpolation weights are reestimated for each word on the basis of previously seen text using the EM algorithm and a moving window of 400 words, obtaining a 10% relative perplexity improvement over a bigram baseline of 532.1.

The work reported in [Seymore and Rosenfeld 1997/CMU, Seymore et al 1997, Seymore and Rosenfeld 1997/Eurospeech] uses a set of broadcast news articles that are pre-labelled with keywords to define 5,883 separate topics. Given the 1-best transcription from a speech recogniser, a fixed number of topics[4] which are similar to the transcription are selected. This similarity is assessed both using a TFIDF measure[5] and a naïve Bayes classifier.[6] The resulting topics are then used as the training text for building a set of *n*-gram language models, one per topic, which are then interpolated together using weights found via the EM algorithm and used to rescore a lattice output by the speech recogniser. Both classification methods are found to perform reasonably well and to be robust to noise but no worthwhile overall word error rate improvement is obtained in either case. Assessed using perplexity, the Bayes selection method is found to be marginally better than the TFIDF distances. Experiments on agglomeratively clustering the articles and building models from groups as well as leaf articles did not lead to any worthwhile gains. An additional scheme was tested which prevented articles from being clustered into topics where they did not fit just because the arbitrary topic size threshold had not been reached, obtaining a marginal improvement over the alternatives in perplexity assessments.

In [Seymore et al 1998/ICASSP], a method of unnormalised exponential interpolation of the pre-labelled topic models described above was tried, obtaining minor gains relative to linear interpolation, whilst in [Seymore et al 1998/ICSLP] an alternative non-linear topic interpolation method is proposed whereby words are modelled differently depending upon whether they are considered 'general', 'on-topic' or 'off-topic', but this performed more poorly than linear interpolation – this last method was described in more detail in section 2.8.2 on page 61.

Work on text classification in [Joachims 1996] compares the naïve Bayes classifier method with the TFIDF-based method for constructing topics and concludes that both methods

[4] 20 was found to work well.

[5] The TFIDF – Term Frequency/Inverse Document Frequency – of a word is a measure of the number of times that the word appears in a topic and how many topics the word appears in overall. A vector of TFIDFs can be constructed containing each word in the vocabulary on a per-topic basis, and then the cosine distance between vectors – that is, the angle between them – can be used to evaluate the similarity of two topics.

[6] The naïve Bayes classifier uses a probabilistic model to estimate which words are in a particular topic – in [Seymore and Rosenfeld 1997/CMU, Seymore et al 1997, Seymore and Rosenfeld 1997/Eurospeech] it assumes words in a topic are independent and uses one unigram language model per topic to calculate the similarity of another piece of text in terms of its likelihood given the model.

work well, although the probabilistic Bayes method is found to perform better overall once it is well-trained but is "very sensitive to the inaccurate probability estimates which arise with a low number of training examples." A probabilistic version of the TFIDF classifier is proposed as a compromise.

In [Iyer et al 1994] each paragraph is considered a separate article and topics are found by clustering these agglomeratively using a similarity measure based on the normalised number of content words in common between two clusters. Topics are then iteratively reclustered using a *k*-means method by building models using the existing topics and moving each article to the topic which assigns it the maximum likelihood – referred to as the 'naïve Bayes method' above.[7] Reclustering proceeds until the size of the topics becomes mostly constant. With topic weights set on a per-sentence basis using the EM algorithm, and a general model trained on all the text also included for robustness, a small reduction in word error rate is obtained with a 5000 word vocabulary and 5 separate topics – increasing this to 8 topics gave a lower perplexity but a higher word error rate. In further work reported in [Iyer and Ostendorf 1996], the topics are again agglomeratively clustered but using a similarity metric based on the combination of inverse document frequencies, described in more detail in section 3.5.1 below. An alternative but much more computationally-intensive method of topic reestimation based on the EM algorithm is also used which allows each article to be assigned to more than one topic. In [Iyer 1994] the two different methods of reclustering are compared and the EM version is found to lead to a minor relative improvement in word error rate when using 5 topics.

The similarity metric employed in [Iyer and Ostendorf 1996] is based on a refined version of the work in [Sekine 1994], adding additional normalisation. The underlying combination of the inverse document frequencies of words, normalised by the number of words in each piece of text, was itself originally described in [Sparck-Jones 1973]. Although this metric contains no term frequency component, it is plausible to propose that this simplifying assumption actually improves robustness since it must curtail the amount of noise due to its massively reduced state space. Whilst word frequency certainly gives a measure of confidence when attempting to retrieve specific articles to match some search criterion, its contribution when attempting to merge articles into groups must be rather less since there are typically far more unique words per article than there would be unique content words in a search query. Empirically, the results

[7] Although not explicitly stated it appears that trigram models were used for this.

in [Iyer and Ostendorf 1996] are good with 5 topics constructed, whilst the analysis in [Sekine 1998] is similarly encouraging.

Perplexity gains on broadcast news through use of a mixture of word *n*-gram language models are reported in [Clarkson and Robinson 1997], who also conclude that topic selection is robust to noisy word sequences such as those that might be present in the 1-best output of a speech recognition system. Sets of 30 and 50 topics constructed via a *k*-means method seeded with a selection of random articles are found to give better performance than smaller sets of topics – sets of over 50 topics are not assessed, and unigram model perplexities are used to evaluate similarity. A subsequent failure to translate perplexity gains into word error rate reductions leads to a statement in [Clarkson and Robinson 1998] on the difficulty of performing adaptive language modelling that "even fairly large reductions in perplexity are no guarantee of a reduction in word error rate." In [Clarkson 1999] it is suggested that if articles were clustered using an agglomerative rather than the *k*-means process actually used that better topic sets might result.

Although the EM method is the most popular method of finding topic weights in a mixture language model, other methods have been proposed such as the algorithm described in [Ogata and Ariki 1998] whereby a TFIDF-style selection metric is used but with an additional per-word weighting of the global unigram probability of each word.

3.4 Building topic corpora

3.4.1 Article boundaries

Automatically finding article segment boundaries is a difficult task[8] and one which is often not necessary when training a language model – most available text corpora are extracted from a medium with natural boundaries, so in practice constructs such as sentence boundaries, paragraph breaks and news article divisions are often available.

For a large text corpus, choosing to place article boundaries between all sentences risks losing information when clustering since it is likely that many adjacent sentences con-

[8]For example the current state of the art in topic boundary detection during speech recognition is examined in the regular NIST Topic Document and Tracking evaluations – see `http://www.nist.gov/speech/tests/tdt/tdt2001/index.htm` for details of the 2001 evaluation.

tain a degree of topic correlation. It will also result in an extremely large number of articles. Many corpora widely used for large vocabulary language model training are derived from news sources, however, which are frequently broken down into a series of separate stories. These existing story boundaries can therefore be used to define the set of indivisible articles. For the two corpora which will be used for the experiments in this thesis, contrast the number of sentences in each with the number of these inherent stories:

Corpus	Words	Sentences	Stories
Broadcast News 1992-96[9]	144,154,458	8,346,334	121,798
LA Times and Washington Post 1992-96[10]	25,067,264	1,061,464	32,088

These figures make it clear that using these inherent boundaries at least produces a more manageable number of separate articles. Manual inspection also suggests that these boundaries appear reasonable. An example article is given in section B.4.4, appendix B.

3.4.2 Normalisation

Before articles can be compared and clustered, it must be ensured that they are in a standard format suitable for whatever tools are in use. With the corpora described here, for example, this involved extracting the raw article text from the compound binary file format it was supplied in whilst being sure to maintain all existing article boundaries. This was performed using custom-written software. The raw text of each article was then conditioned into a standard format which added sentence start and end markers, stripped punctuation and converted each word into a consistent upper-case format. Various abbreviations were also converted into standard spoken forms by a series of processing scripts in order to attempt to normalise conventions across all corpora elements.[11]

[9] Available on CD from Primary Source Media – see `http://www.psmedia.com/`. Some Broadcast News text is also available from the Linguistic Data Consortium (LDC) – `http://www.ldc.upenn.edu/` – see catalogue item LDC98T31 or go directly to `http://www.ldc.upenn.edu/Catalog/LDC98T31.html`.

[10] Available from the Linguistic Data Consortium (LDC) – `http://www.ldc.upenn.edu/`. See for example catalogue item LDC95T21 at `http://www.ldc.upenn.edu/Catalog/LDC95T21.html`.

[11] Normalisation also made trivial stories with no text content easily distinguishable, which reduced the number of distinct stories in the Broadcast News (BN) corpus to 121,701 and in the Los Angeles Times and Washington Post (LATWP) corpus to 32,080.

3.4.3 Existing labels

Although the Broadcast News corpus used here had article index tags associated with each story labelling it with some notion of topic, one problem with pre-labelled tags is that they are not necessarily consistent due to differing classification decisions having been taken by different people, especially over as wide-ranging a text from multiple original sources as the Broadcast News corpus. Another issue is that if any additional text is required to be added it must also be tagged with the same system, which in the example of the Broadcast News text is additionally not publically documented. Manually-tagged classifications are not necessarily optimal for statistical language modelling in any case – general headings such as “sport” or “politics” might cover a huge range of different stories, and even detailed ones like “politics – presidential elections” might cover totally separate events given a wide enough era of text. In short, for general purpose language modelling it would be best if the topic assignments could be automatically derived rather than chosen to fit some subjective topic classification of unknown quality, not designed for language modelling use in the first place. Automatic topic clustering also allows freedom over the number of topics built, rather than being restricted to quantities that can be derived from the existing labelling.

3.4.4 Statistical clustering

Clustering algorithms can be broken down into a series of decisions and assumptions, but the two most fundamental components are the similarity metric used to distinguish between topics and the method and ordering used in combining articles.

Popular similarity metrics referred to earlier include the TFIDF measure and those based on probabilistic estimates. In the former the occurrence count of every word in each topic / article is compared with the count of each of those words in all the others, whilst in the latter maximum likelihood estimates – often unigram language models – are used to score text on the basis of its likelihood. The TFIDF vector of each topic represents the difference of the topic from all the others, whilst the probabilistic estimates for each topic represent just that one topic by itself out of any context. If all comparisons are to be made then this is not necessarily a disadvantage. A potential advantage of a probabilistic approach is that the probability estimates can be obtained from exactly the same form of language model as the topics are being optimised for. The inverse document frequency measure discussed in section 3.3 above is also attrac-

tive for its compactness due to the lack of a term frequency component – omitting these may reduce noise and improve the clustering consistency.

Given a chosen similarity metric, articles can then be clustered into topic sets. This can be performed agglomeratively by repeatedly searching for and making the individual best merge until a desired number of topics result. Unfortunately, however, the number of required calculations grows with order $O(m^2)$ for m articles, but approximations such as segmenting the articles into sets which are first clustered separately can be applied in order to speed up the calculation. An alternative method which scales linearly with the number of articles is to use a *k*-means system whereby, from some initial guess at a topic, articles are placed either progressively or simultaneously into the nearest cluster. This process can then be repeated iteratively until some threshold is reached. In both cases the number of target topics is typically chosen manually, based on empirical observations.

Agglomerative clustering is more feasible with simple similarity metrics, such as the inverse document frequency (IDF) measure, whilst *k*-means clustering allows more complex similarity methods to be used by virtue of not requiring each topic model to be rebuilt after every article move.

3.5 Experimental work

In the work reported here both agglomerative and *k*-means clustering methods were assessed, and both the IDF and probabilistic similarity metrics were used. Specifically, article clustering was first performed in an unsupervised iterative agglomerative manner by repeatedly searching for and then merging the two articles found to be most similar according to the IDF word co-occurrence metric – defined formally below and based on that proposed in [Iyer and Ostendorf 1996] – until a given number of article clusters was reached. Each of these groups of articles was then treated as a distinct topic for the purposes of building 4-gram language models. Topics were then reclustered in further experiments which aimed to optimise the topic clusters for the target 4-gram language models – see section 3.5.2 below for further details.

3.5.1 Agglomerative clustering

Each of the m articles is initially placed into a singleton-article topic, $T_1, \ldots, T_m$. The similarity between two topics, T_a and T_b, is defined as S_{ab} where:

$$S_{ab} = \sum_{w \in T_a \cap T_b} \frac{N_{ab}}{|T^w|} \times \frac{1}{|T_a|\,|T_b|} \tag{3.1}$$

$|T^w|$ is the number of topics that contain the word w and $|T_a|$ is the number of unique words in the topic T_a, whilst N_{ab} is a normalisation factor defined as follows:

$$N_{ab} = \sqrt{\frac{N_a + N_b}{N_a \times N_b}} \tag{3.2}$$

N_a is the number of articles contained within the topic T_a. This normalisation factor is used to stop exceptionally large clusters from forming and dominating the rest of the clustering process. The inverse document frequency measure inherently means that words which appear in many different topics, such as function words, are discounted automatically, but in addition to this a stop list of 182 high frequency function words was used[12] – these words were ignored when clustering topics, as were all one and two letter words.[13] This was used to speed up the clustering process based on the assumption that these high-frequency function words were unlikely to contribute to the distinctiveness of the topics. A similar argument can justify the removal of all one and two letter tokens but in fact these were added to the stop list to allow an efficiency improvement in the token hashing algorithm used in the implementation of the article clusterer.[14] Other than this stop list, however, the vocabulary was not limited in any way – the total number of unique words used for clustering was the full number remaining in the combined corpora: 274,047 words.

Clustering proceeds by comparing each topic with every other topic in order to find the choice of a and b which minimises S_{ab}. Topics T_a and T_b are then merged together to form a new, larger topic. At each stage all possible topic pairs are compared. This process repeats until the desired number of topics results.

[12] See appendix A for a complete list of these stopped words.

[13] The removal of all 1 and 2 letter words added an additional 28 words to the stop list – this total includes all single letters such as '`A.`' and '`B.`' as used in vocalised abbreviations, for example '`C. N. N.`'.

[14] Tokens were assumed to be at least 4 bytes long including terminating character.

The article set used to define the initial leaf topics was the complete collection of articles from the Broadcast News 1992-96 and Los Angeles Times and Washington Post 1992-96 corpora as described in section 3.4.1. Given the large number of articles, however, it was not feasible to perform agglomerative clustering using this complete article set by comparing each topic with every other before each individual merge operation. Therefore a refinement to the algorithm was applied whereby the article set was first split into groups of 1000 articles. Each 1000 article group consisted of articles which had originally followed one another in the corpus with the result that the articles in each group were generally for a consecutive range of times and dates. Each of these groups was then clustered from 1000 down to 100 topics, with the temporally-local grouping of articles also encouraging era-specific stories to merge together in an intuitively reasonable way. This process was then repeated iteratively in a bottom-up manner by using the 100 resulting topics as new indivisible articles and gathering ten sets of them into a new initial group of 1000, and repeating until the desired number of topics resulted.

3.5.2 *k*-means reclustering

In the experiments described in section 3.6.3 below, *n*-gram models were used to assess the performance of the clustering algorithm. Therefore an attempt was made to optimise the topics directly for use in these *n*-gram models by targetting them directly in the similarity metric. As discussed above, clustering articles using a relatively complex similarity metric such as a full 4-gram model is considerably more feasible via a method which does not require the models to be regularly rebuilt. A suitable solution is to use *k*-means clustering, whereby each topic is represented via a centroid and each article is reassigned to the topic whose centroid is nearest – because the number of overall topics remains unchanged after each move it is not necessary to rebuild the models until the end of an iteration. An additional motivation for reclustering rather than clustering from scratch again is that in order to build well-trained *n*-gram models it is necessary to have a good quantity of training data. Building a separate model per article to initiate the clustering would lead to severe under-training, given the 4-gram models that were used to represent each topic centroid. Distance was assessed in terms of the likelihood assigned to each article by each topic model.

The models used for this reclustering were full 4-gram models with cut-offs of 1, 3 and 3 for 2-, 3- and 4-grams respectively with Good-Turing discounting and Katz backoff – this precisely matched the models used to assess the topic quality, as described below.

Each centroid topic model was built using all the articles contained within the relevant topic. An additional difference between the IDF and probabilistic similarity metrics was that the latter used a limited vocabulary to construct each 4-gram model, with the 65,257 words used equal to less than a quarter of the total unique words represented by the IDF vectors.

3.6 Topic analysis

To assess the performance of the agglomerative clustering algorithm, sets of 4, 8, 50 and 100 topics were constructed from the entire set of 153,781 non-trivial articles.

3.6.1 Topic sizes

The tables in appendix B sections B.1 to B.4 give a breakdown of the individual topic sizes for each set of topic corpora.[15] Additionally, figures B.1 and B.2 are histograms showing the number of articles and words respectively in each topic of the 50 topic set. As can be seen in figure B.1, the number of articles per topic has no extreme outliers, but the same is not true for the number of words in a topic as shown in figure B.2. This is due to one of the topics containing stories whose average length is much longer – recall that no term frequency component is used in the clustering metric so the clusterer is not aware of precisely how many words are in each topic. It is not intuitively unreasonable for the topics to vary significantly in terms of the overall size – after all, some topics are likely to be better covered than others in the training data.

An outlying large topic is also present in the 100 topic set and has only slightly less words than the largest in the 50 topic set, as figure B.3 shows[16] – in fact by inspection these are found to correspond to the same news story. Figure B.4 demonstrates that the number of sentences and words in each topic are closely related, which is not a surprise given that each artice is large enough that the average sentence lengths will be similar. Similarly to the 50 topic case, figure B.5 indicates that the number of articles in a topic is more uniform. The clustering algorithm contains a normalisation term to stop exceptionally large clusters of articles from forming, however, so this is not unexpected.

[15]The tables and histograms referred to in this section have been placed in an appendix in order to avoid several pages of histograms interrupting the flow of this chapter – this also allows them to be presented in a single block and adjacent to examples of the topic contents.

[16]The same bin size is used as for the 50 topic histogram, allowing a fair comparison.

The remaining two figures in appendix B, figures B.6 and B.7, show the number of articles and words in each of the 100 topics after performing one iteration of *k*-means reclustering – these use the same scales and bin sizes as figures B.5 and B.3 respectively to allow easy comparison. In figure B.6 it can be seen that the reclustering process, although not having any explicit normalisation for the number of articles, actually tends towards a slightly less fragmented and marginally smoother distribution, reducing the number of articles in the largest topics and also producing a few topics which are smaller than any of those previously seen. A similar but less pronounced effect is also seen by comparing the number of words in each topic, as shown in figure B.7 relative to figure B.3.

3.6.2 Topic contents

Inspection of the topics formed via the agglomerative clustering algorithm suggests that the article groupings make intuitive sense – the most frequent words from three topics randomly selected from the set of 100 are listed in section B.4.2. The topics represented can be clearly inferred from these lists. For example, the first topic is about the war in Bosnia, the second about women's health care and the third about a façet of the US presidential election campaign. Section B.4.3 lists the opening and closing sentences from some randomly chosen articles from one of the topics, whilst section B.4.4 includes the full text of one of the smaller articles. As can be seen, the two topics illustrated in sections B.4.3 and B.4.4 are both about the O.J. Simpson murder trial that took place during the era of the training corpus – in fact these two example topics represent the largest two topics in terms of number of words in the 100 topic set. Clearly a significant percentage of the corpus is about what was the biggest story in the US over an extended period of time.

3.6.3 Topic purity

Each individual topic corpus was split into a training and test set in the ratio of 90% training sentences to 10% test sentences, with the test set sentences chosen randomly from the full topic. An *n*-gram model for each topic was trained on the 90% of training text and was then used to calculate the perplexity of the remaining 10% test text for that topic. These individual test set perplexity results can then be compared with those of general models on the same test set. All models were built as 4-grams with Good-

Turing discounting, Katz back off and cut-offs of 1, 3 and 3 for 2-, 3- and 4-grams respectively.[17]

An important issue that arises is whether there is enough training data to adequately train each *n*-gram model for a given topic corpus. To assess this, two non-topic models were built per topic set. The first of these, labelled 'general' here, used a random selection of sentences from all the training data with the number of sentences chosen to correspond with the average size of topic for each topic set. A second model for each set was also built, this time on all the training data from all topics, here called the 'overall' model. These two models allow the effect on the *n*-gram model performance of the amount of training data to be assessed.

Figures 3.1 and 3.2 plot the perplexity of the topic-specific models against those of the overall model for the sets of 4 and 8 and also 50 and 100 topics respectively. A line of equal perplexity is indicated, with any points appearing above that line representing topics for which the topic-specific model outperformed the overall model, whilst any points below the line represent a relative decrease in performance. As can be clearly seen, however, the unigram performance of each topic model improves on that of the overall model. In the 4 and 8 topic cases an almost identical result is obtained when comparing with the general model instead (not illustrated), revealing that the smaller quantity of training data does not limit the unigram performance. In the 50 and 100 topic cases, however, some relative degradation is noticeable, as is clear from figure C.1 in appendix C – vertical lines indicate the difference in the test set perplexity of each topic as assessed by the overall and general models. This illustrates that, even for unigrams, splitting the training corpus into 50 or 100 components leads to some models being more poorly trained than they would have been on the full corpus.

The result of reclustering the 4 and 8 topic sets is illustrated in figure 3.3. The distance of each point from the line of equal perplexity is the measure of how much better or worse each topic performs than the overall model. Relative to the original topic sets it can be seen that a few topics get worse and none get better. Note that different overall models are used for the original and reclustered topic sets because the 90%:10% selection of training and test text for each topic was made again using the reclustered topics. Figure C.2 in appendix C illustrates similar results for the 100 topic set and it can be seen that

[17] All language model experiments described in this chapter were performed using the HLM language modelling toolkit described in [Young et al 1997] plus the perplexity tool component of the toolkit described in [Odell et al 1996].

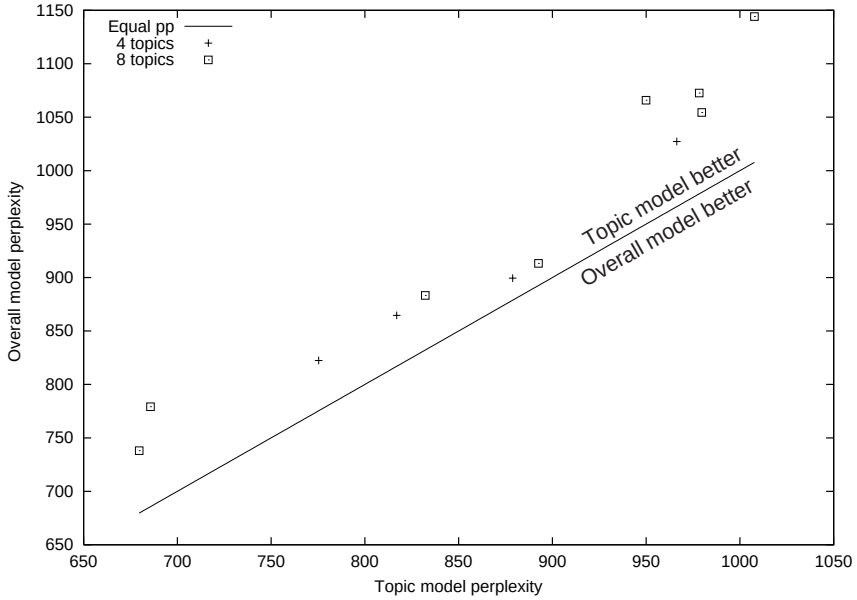


Figure 3.1: 4 and 8 agglomerative topic sets – perplexity of topic test sets plotting topic against overall unigram models

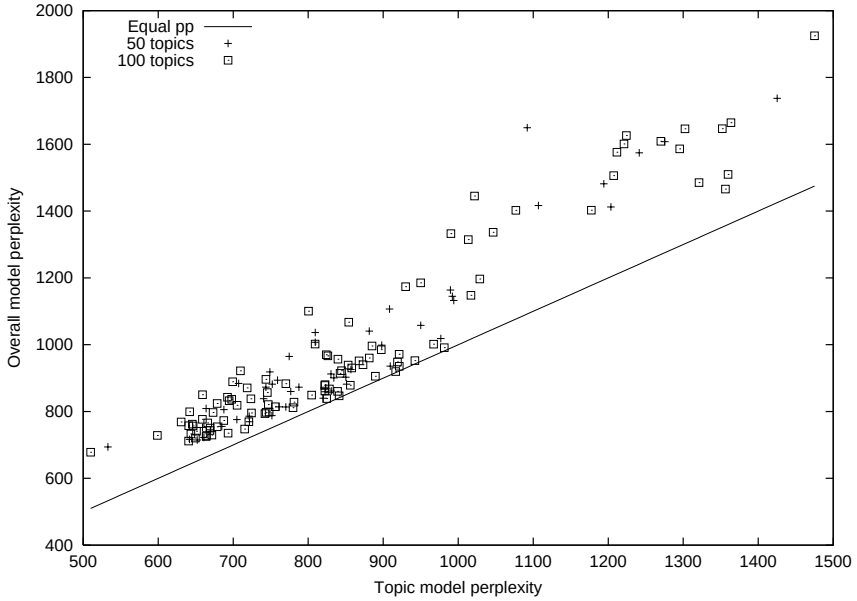


Figure 3.2: 50 and 100 agglomerative topic sets – perplexity of topic test sets plotting topic against overall unigram models

some topic unigram models actually perform worse than the overall unigram model after reclustering. It can also be seen that all the topics with the highest perplexity test sets drop significantly to lower perplexities, both with the topic and overall unigram models, whilst the vast majority of those with lower perplexity topic test sets increase in perplexity. This suggests that the overall effect of the *k*-means clustering is to smooth out the topics and move them closer to some 'average' text – all the test sets become more uniform.

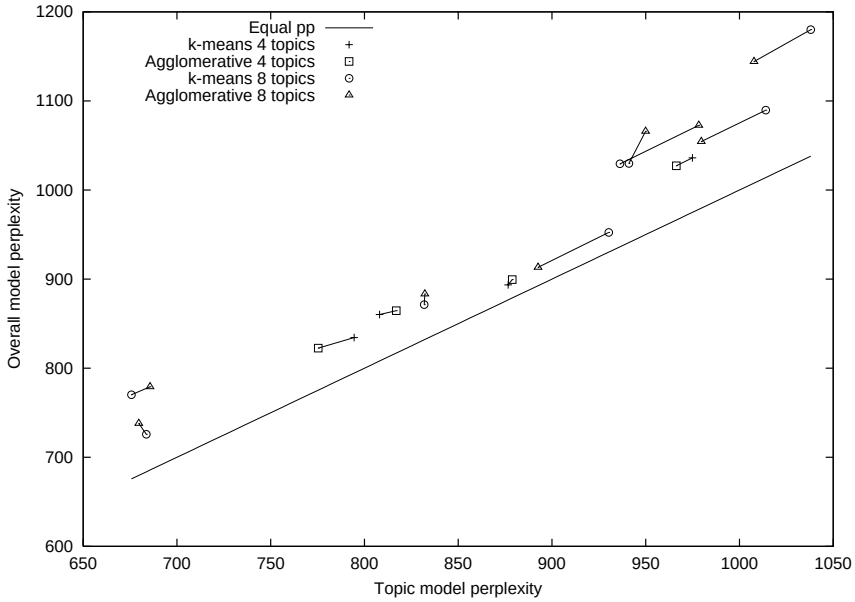


Figure 3.3: 4 and 8 topic sets – unigram perplexities of appropriate test sets plotting topic model perplexity against both agglomeratively and *k*-means clustered topic models. Lines join data points corresponding to the same topics.

Figure 3.4 shows the perplexity of the bigram topic models on each topic's test set for both the original 4 and 8 topic sets. The higher point is the more poorly trained general model in each case, as would be expected, but it is interesting to note that although all the topic models improve on the general model that relative to the overall model only small perplexity gains are observed and in a few cases the perplexity actually gets worse. This illustrates, even in this 4 topic bigram case with the training data for each topic of the order of a few tens of millions of words,[18] that the reduction in the

[18] See table B.1 in appendix B.

quantity of training data inherent through the partitioning of the corpus can lead to a drop in performance of an *n*-gram language model. It is not surprising that this effect becomes more pronounced for higher orders of *n*, as figure 3.5 illustrates – in fact, all topic models have higher perplexities than the overall model on their topic test sets, even though conversely every topic model outperforms the general model. A similar observation can be made as the number of topics increases, as can be seen from the bigram results in figure 3.6 for 50 topics and also the 100 topic results shown in figure C.3, appendix C. Combining more topics with higher-order *n*-grams, as in the 50 and 100 topic trigram models in figure 3.7 for example, leads to a consistent separation between the performance relative to the overall and general models, with all the topic models being notably inferior to the overall model and a clear trend evident.

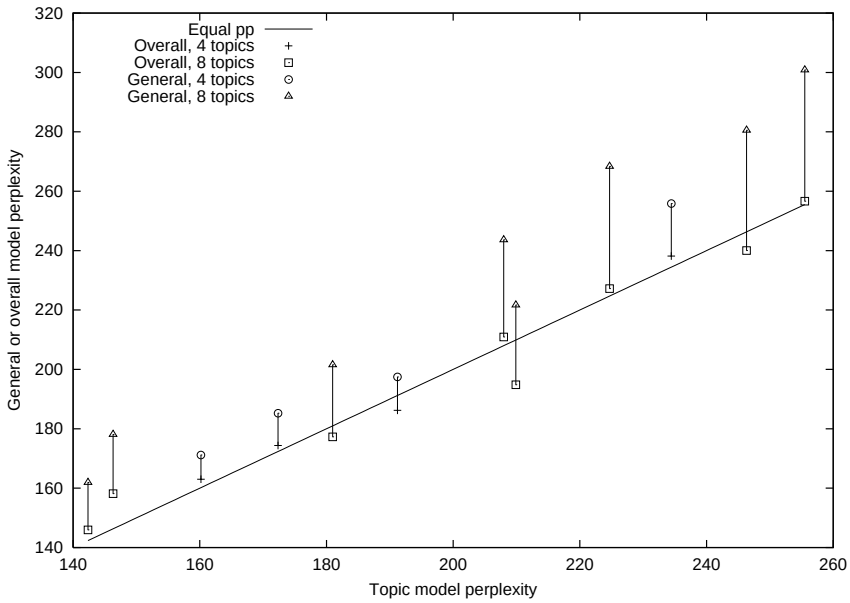


Figure 3.4: 4 and 8 agglomerative topic sets – bigram perplexities of topic model against both overall and general models. Lines join data points corresponding to the same topics.

In figures C.4 and C.5 in appendix C it can be seen that there is no overall gain from reclustering as assessed by the bigram perplexity of the 4 and 8 topic models – the average per-topic change in perplexity is close to zero. It is not surprising, then, that comparing the two figures, printed with the same scale, illustrates that several of the reclustered bigram 4 and 8 topic models also perform worse than the overall model, as

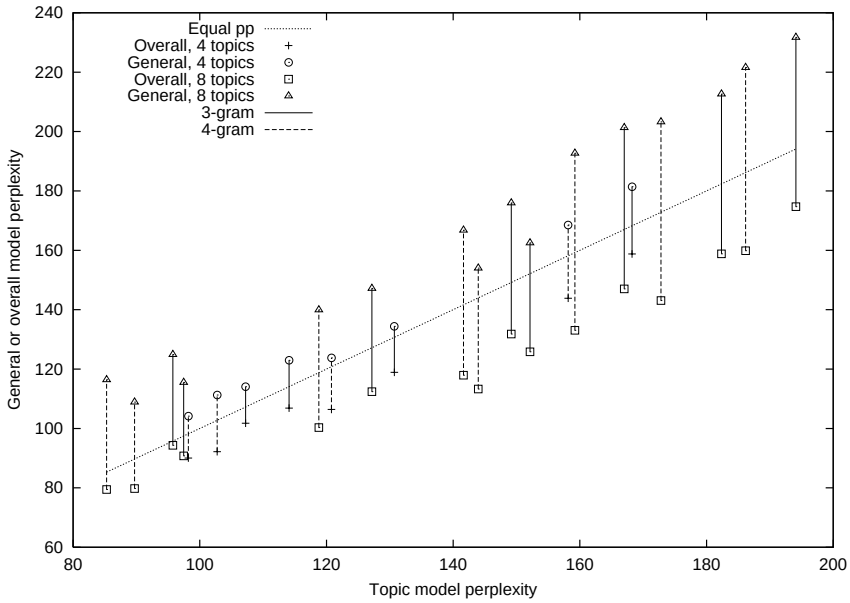


Figure 3.5: 4 and 8 agglomerative topic sets - 3- and 4-gram perplexities of topic model against both overall and general models. Lines join data points corresponding to the same topics.

for the original topic set, but all perform better than the general model. In figure 3.8 the same overall no change can be observed for the 100 topic set, but it is interesting to note a general pattern of moving towards a narrower variance in terms of the perplexity of the test texts as scored by the general model, and to a lesser extent the perplexity calculated by the topic models too. This is the same smoothing effect noted above for the unigram case and evident in figure C.2.

Comparing the 3- and 4-grams built using each of the 100 topics reveals some interesting facts, as illustrated by figure 3.9. Joined points correspond to identical training and test sets but with one point for the 3-gram perplexity and the other for the 4-gram perplexity. The data can be clearly seen to be divided so that all bar one of the topic models is worse than the overall model but that every topic model is better than the general model. Some of the topic models have perplexities almost three times lower than the general model, in fact. What is particularly interesting, however, is the relationship between the 3- and 4-gram results in each of the two partitions. Comparing to the overall model, in the lower section of the graph, it can be seen that the joining lines

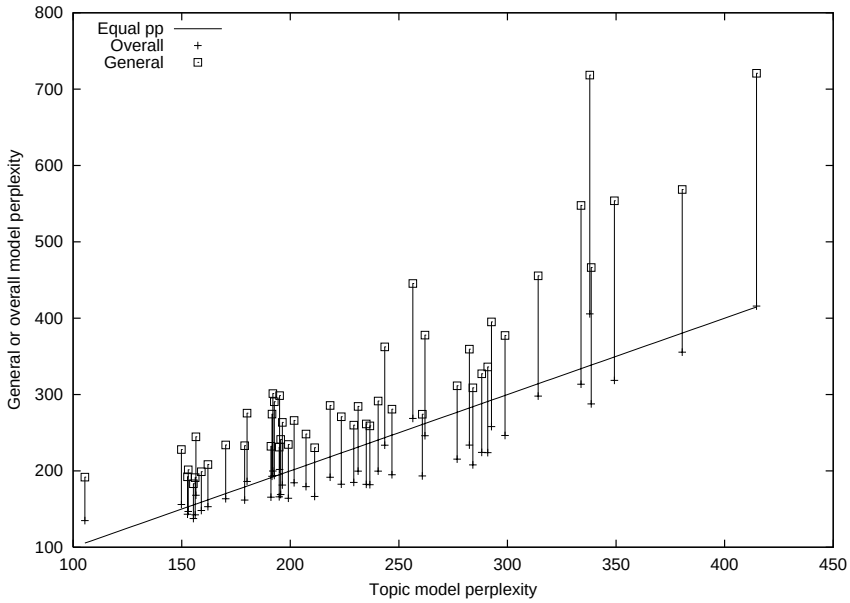


Figure 3.6: 50 agglomerative topic set – bigram perplexities of topic model against both overall and general models. Lines join data points corresponding to the same topics.

are nearer to vertical than horizontal[19] which indicates that the decrease in perplexity when moving from a 3-gram to a 4-gram is greater in the overall case than in the topic case, clearly demonstrating that the limited training data is reducing the ability for the topic model to be improved with a longer-length *n*-gram.

This observation about the behaviour relative to the overall model is in strong contrast to the changes in perplexity observed when comparing to the general model, in the upper section of figure 3.9, where the lines are almost all near to horizontal. The lines are horizontal because in most cases the perplexity does not change as the general model increases from a 3-gram to a 4-gram, suggesting that the additional word of context does not improve the model – this is because at this length of context there is so little training data that the counts are too low to provide statistically significant estimates, and many will be lost through the cut-off of 3 applied to the 4-grams. It is especially interesting, therefore, that improvements *are* observed in the topic case (the horizontal movement), since this shows that enough repeated tuples of 4 words are observed

[19]Especially when considering the differing horizontal and vertical scales.

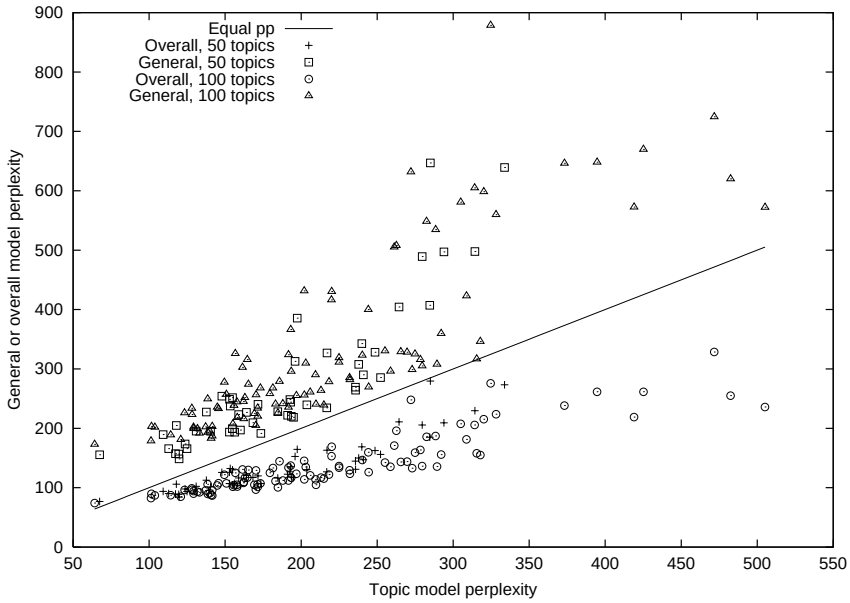


Figure 3.7: 50 and 100 topic sets – trigram perplexities of topic model against both overall and general models. Same-topic joining lines omitted for clarity.

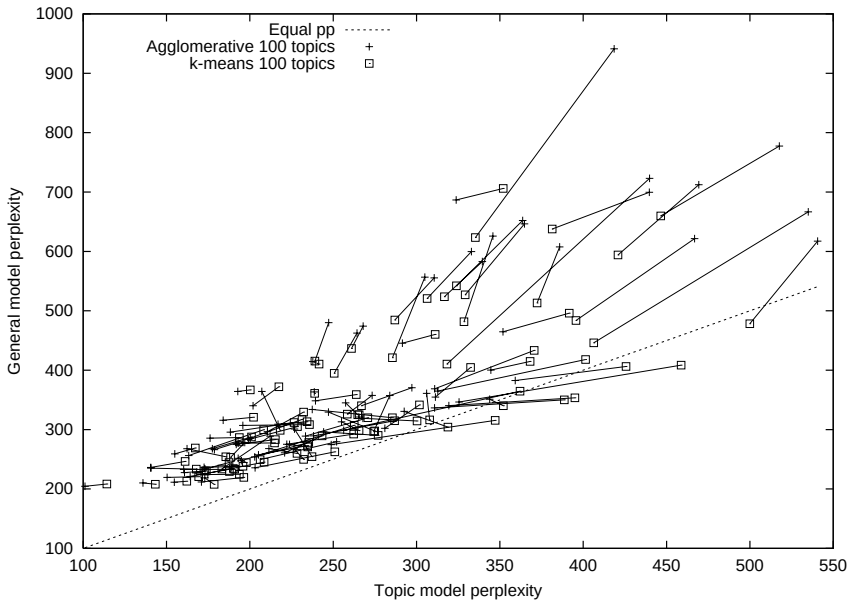


Figure 3.8: Reclustered 100 topic set – bigram perplexities of agglomerative and *k*-means clustered topic models against the general model.

Agglomerative 100 topic set – topic model perplexities of both 3- and 4-grams against overall and general models. Lines join data points corresponding to the same topic and overall/general set – that is, they join corresponding 3- and 4-gram points.

Figure 3.9: Please rotate page to read caption

that the 4-gram model provides useful additional disambiguation, or does not lose so many of its counts through cut-offs. In fact, the relative effect of the cut-offs on low frequency events can be discerned from the fact that in the general 4-gram model there are 126,109 3-grams but only 84,264 4-grams, compared to an average of 156,173 3-grams and 137,559 4-grams in the 100 topic models. This demonstrates clearly that there is coherence within the topic training sets not only in terms of word co-occurrences and frequencies, as previously deduced from the unigram results, but also in terms of tuples of at least 4 words. This is a positive result because the agglomeratively-found topics were not clustered towards any context-dependent metric,[20] either implicit or explicit. The corresponding graph after reclustering is given in appendix C figure C.6 and remains very similar despite the explicit 4-gram perplexity optimisation – in fact, the only major difference is that several of the topics become worse than the general model, so the reclustering actually damages performance.

Finally, figure 3.10 shows the improvement in perplexity when moving from a 3- to a 4-gram for the 4, 8 and 50 topic sets. As can be seen, once again more is gained in the topic models than in the general model, showing a greater degree of topic coherence than that in the random selection of articles. Figure C.7 in appendix C shows on the same scale that for all these topic sets the overall model clearly outperforms the 3- and 4-gram topic models.

3.7 Conclusions

In summary, it can be concluded that agglomerative clustering using an inverse document frequency measure results in groups of articles which share not just the word co-occurrences explicitly catered for in the similarity metric but also word frequency characteristics as tested for via the unigram perplexity. In addition experiments have shown that there is a greater degree of word sequence coherence in these topics, as assessed with longer length *n*-grams right up to the tested maximum of 4-grams, than is found in various random groupings of articles. Both of these conclusions hold for all four topic sizes tested.

An attempt to improve the topics through reclustering via a *k*-means process which directly optimised on the perplexity of the target language models led to an overall

[20] Such as an *n*-gram.

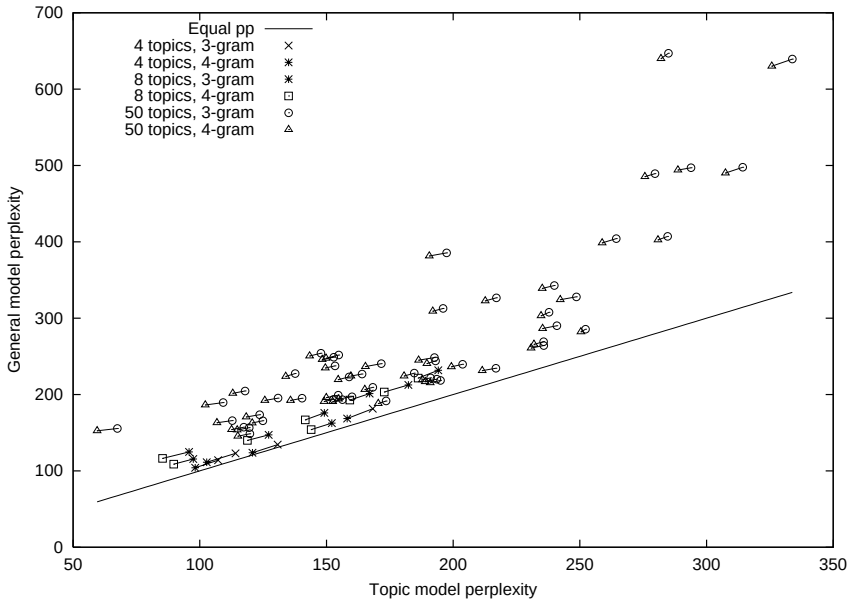


Figure 3.10: Reclustered 4, 8 and 50 topic sets – 3- and 4-gram perplexities of *k*-means clustered topic models against the general model.

marginal decrease in performance. In particular it did not increase the 4-gram performance of the topic models despite this being the exact form of the similarity metric, and it had an overall smoothing effect on the topics, making them all more similar to the average text than some of the original topics.

Overall, all the topic sets gave substantially better performance than a comparably-trained *n*-gram model built from a random selection of articles, but only in the 1-gram case did their corresponding models outperform the overall models built from all the training text. In the 2-gram case the average performance was similar to that of the overall model, whilst in the 3- and 4-gram tests the overall model outperformed the topic models in every case. Therefore to take full advantage of the topic sets something more than mere selection of an alternative 4-gram model is required. Linear interpolation of topic models to overcome the limitations of training data has met with less success than might have been hoped for, as reported in [Clarkson and Robinson 1998], and alternative combination schemes such as in [Seymore et al 1998/ICSLP] have also met with little success. And yet, as this chapter has shown, sufficient information is contained within the topic sets that it should be possible to build adaptive topic *n*-gram

models which outperform any unadapted overall model, given suitable same-topic test text. The principle problem is that as the quantity of training text is reduced so the *n*-gram models begin to decrease in their average performance, and so when trained on smaller topic corpora they underperform relative to models built with more training text. If a model could be constructed which alleviated this problem then topic adaptation could perhaps be used to obtain significant speech recognition gains.

4. Adaptive class-based models

Chapters two and three discussed various methods of language model interpolation and adaptation. In chapter two it was noted that it is much harder to improve on a state of the art large vocabulary recognition system than a smaller, less powerful and more poorly trained one, whilst chapter three concluded that word *n*-gram models suffer from severe under-training problems when built from topic corpora. In this chapter a method of improving on the current state of the art word 4-gram and class 4-gram combination is proposed through the implementation of a topic adaptation method which in addition avoids the necessity of a large increase in both model size and training data.

4.1 A topic class-based *n*-gram model

Multiple topic interpolation in a word *n*-gram model is often performed by linearly interpolating a selection of separate topic *n*-gram models, each weighted based on some assessment of how likely that topic is at a given point in time – this likelihood is often assessed on the basis of the perplexity of the particular topic model on recently-seen text.

Abstracting the same principle to class *n*-gram language models, various possibilities arise. Because the model is decomposed into a product of two probabilities, there are at least three obvious possibilities based on adapating either just the word-given-class prediction *or* the class-given-history prediction; or alternatively both could be adapted simultaneously. When the possibilities added by adapting the class map too are considered, clearly there are quite a few potential models that could be proposed. On the other hand it could be argued that a class-based language model is more general than a word model so less appropriate for modelling specific topics – in [Maltese et al 2001] for example it is concluded that “class-based LMs [make] word-based LMs less dependent on the domain of the training corpus when combined”.

Topic adaptation using class *n*-grams has various potential advantages. Firstly, as has previously been noted in this thesis, class models have the advantage that they require less training data than word *n*-gram models in order to be sufficiently well trained – this is not surprising since there are far fewer distinct history equivalence classes in a typical class model than a word model.[1] In addition each model is often smaller than a word model, so it becomes more feasible to store a larger number of separate models. If a class topic model where only the word-given-class probability is adapted is envisaged, however, then the storage space savings become even more significant – instead of the necessity of storing a full set of *n*-grams for each and every distinct topic, the only requirement in such a model is to store a set of $|\mathbb{W}|$ unigram counts.

A topic class *n*-gram model which adapts only the word-given-class component of the prediction product can be defined like this:

$$p_j(w_i \mid w_{i-n+1}, \ldots, w_{i-1}) = p(w_i \mid G(w_i), T_j) \times p(G(w_i) \mid G(w_{i-n+1}), \ldots, G(w_{i-1})) \quad (4.1)$$

where T_j is the jth topic, $G(w)$ is the class of word w and $p_j(w_i)$ is the probability of the ith word given topic T_j.

In this model both the class map and the class *n*-gram predictions are independent of the topic, and without experimental evidence it is hard to predict whether this restriction will lead to a successful model. It certainly does, however, allow space savings of many orders of magnitude over any alternative system.[2] This model uses only $(t.|\mathbb{W}|)$ parameters over and above a normal class-based *n*-gram language model, where t is the number of topics and $|\mathbb{W}|$ is the vocabulary size.

In practice it is unlikely to be the case that every word in the vocabulary is encountered in every topic text set, so a method must be developed of coping with this other than giving those words zero probability. Possible options include some form of discounting to redistribute counts or probabilities from seen to unseen words, or counts could be hard-limited to a minimum value of 1.

Individual recognition experiments could be performed with each of these topic-specific models, but the aim here is to construct an adaptable model by using a mixture consisting of all the models simultaneously and then to choose a weight for each topic. In

[1] For the sake of clarity I will assume that any class model has substantially less classes than there are words in the vocabulary.

[2] A possible space-saving alternative is to store full word *n*-grams in memory and then to build various topic-specific class *n*-grams from these dynamically, but any system of this sort is likely to be extremely slow and therefore impractical, so it is not further considered here.

practice many of these weights may be zero if it is decided that particular topics should not be active in a given context. The models can be combined by linear interpolation of just the topic-specific component:

$$p(w_i \mid w_{i-n+1}, \ldots, w_{i-1}) = \sum_{j=0}^{t} \theta_j \, . \, p(w_i \mid G(w_i), T_j) \times p(G(w_i) \mid G(w_{i-n+1}), \ldots, G(w_{i-1})) \quad (4.2)$$

where θ_j is the weight of topic j, given that $\sum_{j=0}^{t} \theta_j = 1$ for t topics. A special overall topic T_0 is also defined which has probabilities derived from the union of all the other topics – in other words, the full training text.[3]

To use a model of this form in an unsupervised manner it is necessary to devise some automatic way of determining the weights, θ_j. In virtually all reported work a form of the EM algorithm is used to optimise linear interpolation weights on the basis of some given text – typically some held-out development text for the purposes of static weights. In order to allow adaptation, the weights cannot be globally chosen for this model so in the experiments reported here two methods of selecting suitable text to optimise them on are examined – firstly via recently-seen text[4] and secondly on the existing 1-best transcription of text when rescoring a lattice. Various researchers have concluded that the choice of topic is not sensitive to noise in the text,[5] so even a poor 1-best transcription might be sufficient – on the other hand, adapting to a 1-best solution might simply reinforce existing errors.

4.2 Experimental setup

A corpus consisting of 144 million words of Broadcast News text and 25 million words of Los Angeles Times / Washington Post (LATWP) newswire text, manually segmented by programme and story boundaries into 153,886 articles, was agglomeratively clustered into sets of 50, 100, 250 and 500 topics using the IDF similarity metric as described in the previous chapter.[6] In each case every article was included in exactly one topic – no articles were omitted.

[3] In actual fact, in the experiments reported here this full training text also includes the small amount of non-topic acoustic text that is added to the word model described in the following section.

[4] The previously recognised utterance.

[5] As reported earlier in this thesis, but also see for example [Clarkson and Robinson 1998].

[6] The 50 and 100 topic sets used in this chapter are the same 50 and 100 topic sets examined in chapter three.

A word 4-gram language model was built using exactly the same set of Broadcast News and LATWP corpus text, supplemented with 850,000 words of Broadcast News acoustic text[7] and 100,000 words of Marketplace acoustic text. The HLM language modelling tools described in [Young et al 1997] were used for this task. This combined corpus, and the eras used, was chosen to be similar to the text used to train the HTK-system language models built for the 1997 HUB-4 evaluation [Woodland et al 1998] – this allows the resulting language model to be experimentally evaluated by rescoring the lattices originally built for that evaluation. Cut-offs of 1, 3 and 3 were used for the 2-, 3- and 4-gram components of the word model respectively, with Good-Turing discounting and Katz backoff.

Three class 4-gram models were also built, using specially developed tools built on top of the existing HLM system, one each for 503, 1003 and 2003 classes, with the class maps found using an implementation of the word exchange algorithm described in section 2.3.1. When performing this statistical clustering the sentence start, end and unknown word tokens were kept in singleton classes due to their special status and high frequency – all other words were free to move to any class. Two iterations of the clustering algorithm were performed for each model.[8] The vocabulary of 65,425 words from the original 1997 evaluation system was used, although due to an inability to obtain the same text used to train the original evaluation language models this meant that 105 words in the vocabulary were not encountered in the training data – the training text used was chosen as the best-possible era and genre match to that originally used, however. To work around this problem the probability mass for the unknown word token was evenly distributed amongst these 105 words and the unknown word token itself when rescoring lattices – it was due to the desire to rescore the original evaluation lattices to obtain recognition results that the same vocabulary was chosen. In all experiments a fixed ratio of 0.65:0.35 was used for interpolating between word and class models – this ratio was based on experiments performed rescoring lattices built from the 1997 HUB-4 development test set.

The word-given-class probabilities from each of the three class 4-gram models were then assigned to T_0 in equation 4.2, whilst the class-given-history component of each model was used for all reported class experiments with the given number of classes. Then, for each of the topics in each set, word-given-class probabilities were calculated

[7] Precise transcriptions including all errors.

[8] As the standard model line in figure F.2 in appendix F illustrates, additional iterations give no further useful gain.

for each of the class model sizes using the class maps constructed using the full training set corpus. The resulting topic-specific word probability maps were each assigned a topic index as per equation 4.2. Zero counts – that is, words unseen in a particular topic – were given a floor count of one. Alternative methods of redistributing probability to unseen but in-vocabulary words were investigated, such as a modified form of absolute discounting, but these led to inferior perplexity results in development experiments – it is worth noting that by interpolating word-given-class probabilities together that these estimates are inherently smoothed.

Three baseline models were defined, one for each number of classes. Each of these models consisted of the same overall word 4-gram model linearly interpolated with the appropriate static overall 4-gram class model. Experiments were then performed by altering the class model and testing to see if improvements were obtained. The performance of the adapted class models was also assessed relative to the stand-alone overall class model when the word model was omitted.

4.2.1 Lattice rescoring

Recognition experiments were performed by rescoring the existing lattices previously built for the 1997 HUB-4 evaluation using a modified version of the Lattice Toolkit utilities described in [Odell et al 1996] – these lattices were originally constructed using a 4-gram word model. All weights θ were set to zero except for one topic which was then used to find the log probability of each word in the transcription. Having found the log probability of each word according to each topic an implementation of the EM algorithm [Dempster et al 1977] was then used to find the optimal set of linear interpolation topic weights for the lowest possible perplexity on that 1-best transcription, via iterative adjustments of the weights – the iterative step was repeated until the improvement in perplexity was below 0.0001%. These weights were then used in the topic model described in equation 4.2 when rescoring the relevant lattice.

An additional set of experiments were also performed where just one topic, that which gave the lowest 1-best transcription perplexity, was given a weight of 1 and all the other topic sets were given a weight of 0. This was used as a comparison to compare the effect of single-topic adaptation with that of using multiple topics in a single model.

4.3 Experimental results

The full 1997 HUB-4 evaluation test set used to assess language model performance was chosen because of the existence of a non-branching accurate reference transcription.[9] This allowed the simple implementation of a supervised topic selection system which could be used to find topic weights in order to assess the impact of recognition errors relative to those picked from a 1-best transcription. Because of the training text mismatch described in section 4.2, the 4-gram word model lattice 1-best solution performance of 17.3% word error rate reported in the original evaluation [Woodland et al 1998] could not be reproduced, and a new baseline of 17.6% when rescoring the lattices with the newly-built 4-gram word model was obtained.[10] There were 749 separate lattices in total, with an average utterance length per lattice of 43 words.

4.3.1 Recognition Accuracy

Experiments were run with all combinations of topic and class size, with both single topic and topic mixtures and with and without interpolation with a word 4-gram. Word accuracy was assessed using the NIST scoring tools[11] used for the original evaluation. As table 4.1 shows, the best recognition error rate without using any topic model, of 17.1%, was obtained by interpolating the 1003 class model with the word model. For large vocabulary models such as this it has frequently been found that 1000 classes is more or less optimal for an interpolated class *n*-gram. Taking this model's result as the baseline to improve upon, statistical significance of any change was tested by using the Matched Pairs test at the 5% significance level.

For the stand-alone class models, significant improvements were obtained in every case for all sizes of topic model over the standard model given the same number of classes, with the 250 topic model providing the largest gains, as shown in table 4.2. This is perhaps because using more, smaller topics allows a finer choice of combined probabilities, although if the number of topics grows too large a degree of undertraining might begin to cause problems.

[9] As opposed to a reference transcription with concurrent alternative text.

[10] In [Woodland et al 1998] a corresponding word error rate for a linearly interpolated word and class model of 16.8% is reported

[11] See section 1.2.2 in chapter 1 and also [Gillick and Cox 1989].

Classes	Word	Perplexity	Error rate (%)
503	-	259.53	19.6
1003	-	219.77	18.5
2003	-	194.86	18.1
-	✓	174.84	17.6
503	✓	160.54	17.3
1003	✓	160.50	17.1
2003	✓	161.51	17.2

Table 4.1: Topic-independent baseline word recognition error rates – a ✓ in the **Word** column indicates that the class model was interpolated with a word *n*-gram model

In all cases the performance of the topic-mixture models was better than the performance when using the single best-performing topic, with the mixtures algorithm giving larger relative gains as the number of topics increased, presumably due to its ability to include more than one of the more specialised topics; for the same reason the single topic selection method performed more badly as the number of topics increased – clearly as the number of topics increases so the amount of per-topic training data must decrease, so it is not unexpected that performance of a single topic degrades.

The results were less clear-cut when interpolating with the word *n*-gram model. The only models to obtain a statistically significant improvement were the two using 50 topics and 1003 classes, with a relative WER improvement of 1.2%. The models with 50 topics performed better than those with more topics, despite the fact that the stand-alone 250 topic model was significantly better than the stand-alone 50 topic one. This disparity almost certainly exists for the same reason that a stand-alone class model generally performs better when it has more classes, even though this may not mean it is better when interpolated with a word model. When linearly interpolating two models the biggest gains are obtained when the models are at their most orthogonal or disparate in terms of what they are modelling. It must be the case that as more topics are added and the class model receives more adaptable parameters, these gains allow a degree of history equivalence class resolution which is already encapsulated implicitly within the word *n*-gram model, thus this gain is not repeated when the word model is included.

		Word error rate (%)			
Classes	**Topics**	**C+1**	**C+EM**	**W+1**	**W+EM**
503	**50**	19.1	18.9	17.2	17.2
1003	**50**	18.2	18.1	16.9	16.9
2003	**50**	17.8	17.8	17.1	17.1
503	**100**	19.0	18.9	17.2	17.3
1003	**100**	18.1	18.0	17.0	17.0
2003	**100**	17.8	17.7	17.2	17.2
503	**250**	19.1	18.8	17.3	17.3
1003	**250**	18.3	17.9	17.0	16.9
2003	**250**	17.9	17.7	17.2	17.2
1003	**500**	18.3	18.0	17.0	17.0

Table 4.2: Topic-dependent word recognition error rates. *Key*: **C** – class model only; **W** – class model interpolated with word model; **1** – only single best topic used; **EM** – multiple topics with weights found using EM algorithm

It is also worth noting that some of the stand-alone class topic models perform as well as the word model alone – there is no statistically significant difference between the word model performance and that of the 2003 class 100 and 250 topic models.

4.3.2 Perplexity

Although the experiments reported here are concerned with improving recognition accuracy, it is interesting to examine perplexity figures too. Table 4.3 provides perplexity figures for the 503 and 1003 class models, with the perplexity calculated on the reference transcriptions. As for the lattice experiments, topic weights are only adapted on lattice / utterance segment boundaries. The 503 class results are shown since they gave the best perplexity results when interpolated with a word model, and the 1003 class perplexities are included for comparison because the best recognition results were obtained with this number of classes. The table omits the non-mixture results, but these are included along with further results in tables E.1 and E.2 in appendix E – word accuracies are also included. Improvements through use of EM-chosen weights are typically around the 4-5% level for models without word-model interpolation and around 1% for those with.

			Perplexity			
			1-best		Reference	
Classes	**Topics**	**W**	**Prev**	**Curr**	**Prev**	**Curr**
503	**50**	**-**	254.41	220.40	252.65	216.16
503	**50**	✓	157.47	150.71	156.98	149.36
1003	**50**	**-**	218.47	195.72	217.10	192.20
1003	**50**	✓	158.53	152.99	158.18	151.78
503	**100**	**-**	252.01	216.86	255.54	215.25
503	**100**	✓	157.43	149.52	157.22	148.71
1003	**100**	**-**	220.28	193.45	218.69	189.14
1003	**100**	✓	158.56	151.98	158.16	150.47
503	**250**	**-**	263.40	215.31	261.73	209.20
503	**250**	✓	158.00	148.52	157.44	146.53
1003	**250**	**-**	225.85	192.66	224.24	187.37
1003	**250**	✓	158.98	151.15	158.68	149.37
1003	**500**	**-**	230.96	193.86	228.93	188.11
1003	**500**	✓	159.77	151.32	159.30	149.37

Table 4.3: Language model perplexities for topic mixture models. *Key:* **W** = word model interpolated where ticked – class model only where not ticked; **1-best** = lattice 1-best text used to choose topics; **Reference** = reference transcription used to choose topics; **Prev** = preceding utterance used in topic estimation; **Curr** = current utterance used in topic estimation

Using the 1-best transcription to optimise the weights on the basis of log likelihood clearly biases a conventional perplexity measurement so for the purposes of comparison four separate perplexity measures are presented in table 4.3, which shows the performance when picking topics on both the 1-best transcription and the reference transcription,[12] using either the current or preceding utterance. The final column, therefore, gives the best possible perplexity results that can be obtained given that the topic cannot change within an utterance. For the preceding utterance experiments, the topic weights for the first segment were set to 1 for the 'overall' model and 0 for all the topic-specific models.

Setting the topic based on the previous, just-seen reference transcription represents the only 'true' perplexity reported here, since it alone can be calculated without use

[12]The current reference transcription gives the oracle, supervised weight settings.

of recognition experiments or 'cheating'. This works relatively poorly, however, with the largest reduction when interpolating with a word model of 2.2% obtained using the 503 class model with 50 topics, whilst the best (1003-class) model for recognition performance gives a 1.5% reduction. In addition, when topic selection is based on the previous segment the 250 topic model performs relatively badly, suggesting that the topic weights are being poorly selected since the lesser numbers of topics may be performing better simply because this increases the likelihood of choosing better weights by chance.

By way of contrast, however, the perplexity values when weights are chosen on the current segment are much lower – this shows that the model described here *can* give large perplexity reductions if the weights are well-estimated, so clearly using the preceding segment is not very successful. The table also reveals that the weights which are computed from the lattice 1-best solutions are relatively close to the best that can be obtained, suggesting that the method of topic choice used for the recognition experiments is a good one. This, however, makes the assumption that the weights which obtain the best perplexity result also obtain the best recognition result, which is not guaranteed to be true. It does at least confirm other reported results which suggest that topic adaptation is pretty much robust to small amounts of noise. The best improvements in perplexity are obtained by the 250-topic models, in contrast to the best recognition results which were for the 50-topic models, but the differences in perplexity between the varying numbers of topics are too small to draw any firm conclusions.

4.3.3 Topic assessment

Examination of the weights selected by the EM algorithm suggests that the topics constructed by the clustering process are sufficiently distinct, since for each lattice the vast majority of the weights are set to zero, with just one, two or three topics taking over 99% of the probability mass - this was true even for the 500 topic model. As an example, the full set of topic weights for each lattice in the 50 topic 1003 class test set is illustrated in figure D.1 – see appendix D for this and additional details.

Experiments were performed, comparable with those in the previous chapter, on reclustering the topics using a *k*-means process on the basis of a probabilistic classifier whereby the perplexity of each training set article was calculated with each separate topic of the class model in turn – that is, with the weight of one topic set to 1 and all the others to

0. Each article was then moved into the topic of the model which gave it the lowest perplexity. Despite performing two iterations of this process, however, no significant improvement in performance was obtained.

In another set of experiments, work was repeated on the 50 topic sets using 6-gram class models. No significant improvement in performance relative to the 4-gram models was obtained.

Finally, a method of topic weight selection based on inverse document frequency similarities as used in the agglomerative clustering algorithm was investigated, calculating the similarity between the 1-best text and each topic set. This lead, however, to a decrease in performance so was not pursued any further.

4.4 Alternative class model adaptation

For the purposes of comparison, some alternative forms of class model topic adaptation were also implemented.

4.4.1 Full topic-specific class models

Full class-based LMs were built for each topic for the best performing model set – 50 topics and 1003 classes – to compare performance. A separate class map was first of all constructed for each individual topic by reclustering for one iteration from the existing overall class map. Each model then had a full 4-gram set of class histories built for it, using the same cut-offs of 1, 3 and 3 for 2-, 3- and 4-grams respectively.

Not all vocabulary words were encountered in each topic, so the same approximation that was used in the previous experiments of setting any zero counts to 1 when calculating the word-given-class probabilities was used. Two sets of experiments were performed, one with and one without this knowledge incorporated into the statistical clusterer. However in both cases it was not possible to move words which were not found in a particular topic set to new classes since no information was known about them, so they were left in their initial class – that is, the class from the initial second-iteration global class map.

The overall class model was included as a topic choice, and then for each lattice the single best-performing topic class model was chosen, with performance assessed by

calculating the perplexity of the 1-best solution from the existing 4-gram word lattices. This method without the set-to-1 knowledge obtained a word error rate of 18.8%, whilst the experiments where the clusterer knew about the set-to-1 rule gave a WER of 18.7%. Both of these results are significantly worse than the baseline class-model-only word error rate of 18.5% which was previously obtained for the overall 1003 class model on its own. For both forms of new model, 374 of the 749 lattices (50.0%) chose to use the general model as opposed to a topic model – this compares with around 10 (1.3%) for the models in the previous section. Given these poor results, on models which are already much more expensive to train and store, further work was not tried, but it seems likely that using more topics would have led to further degradations due to more poorly-trained models.

4.4.2 Changing the class map

Experiments were also performed where the class history *n*-gram probabilities remained constant but the word-to-class mapping changed on the basis of topic, with topics being chosen on the basis of 1-best transcriptions as before.

In principle it makes sense to change the class map on a per-topic basis – in a deterministic class model each word is forced into a single class even though that word might fit better in a different class in a particular context. If the topic sets contain these 'alternative' contexts then finding which class each word fits into on a per-topic basis could improve the model. On this basis, it shouldn't be necessary to change the class-given-history *n*-gram probabilities because these represent abstract class sequences – dependent on the topic context the content of each of these abstract classes should be able to vary.

Inspection of the general class map reveals that classes generally either contain words with semantic similarity – such as CAT and MOUSE or TAX and ACCOUNTANT – or with associated syntactic usage, such as PAUL'S and ANDREW'S or FROM and TO. Each class typically also contains a few 'misfit' words that have no clear connection with the contents of the rest of the class – these are usually low-frequency words which are placed there implicitly without much confidence. These semantic and syntactic clustering examples provide various possible rationales for moves. A misfit word could move to its most likely class on the basis of a particular topic, perhaps due to a different semantic usage. MOUSE, for example, could move from an 'animals' class into a 'computing'

class if a topic was about technology. The classification remains deterministic, but local changes are implicitly allowed in an attempt to improve the predictive power of the language model. This approach has the significant advantage that only one set of class history *n*-gram estimates needs to be stored, thus keeping the language model compact. It does, however, use twice as many topic-specific parameters as the successful static class map approach described previously.

In order to find suitable class maps a clusterer was developed where the class history components remained fixed at initialisation, with the initial class map taken as the overall map previously used. Only the word-given-class components of the estimate were changed with the class map, and in every other respect the word exchange algorithm continued as described in section 2.3.1.

Results using this method, again assessed using 50 topics and 1003 classes, were better than those found when the history probabilities were also modified, but not as good as those obtained when the class map was kept fixed. A word error rate of 18.3% was obtained for the clusterer aware of the set-to-1 rule, and 18.5% when it was not. The 18.3% was an improvement on the static class model's 18.5%, and not far off the 18.2% obtained by the fixed class map topic model when picking the single-best model. 125 (16.7%) of the lattices were rescored using the overall model.

4.4.3 Word *n*-gram topic models

For the sake of completeness, 50 4-gram word models using each of the 50 topic sets were also built. As per all the other models, cut-offs of 1, 3 and 3 for 2-, 3- and 4-grams respectively were applied.

Choosing the best model for each lattice on the basis of 1-best perplexity, and including the overall model as an option, a word error rate of 19.0% was obtained. This result was notably worse than that for all the class model experiments, and in fact significantly worse than the baseline word model performance of 17.1%. A total of 321 (42.9%) of the lattices were rescored using the overall model.

Even though no attempt was made to optimise the word models, this result demonstrates the clear advantage of the class *n*-gram models over word *n*-gram models when representing more than a few distinct topics – no more training data was available for the class models than was provided for the word models. Using more training data,

perhaps via use of less topics – although none of the 50 topics are particularly small, with each containing an average of 3.4 million words – might be expected to improve the word model results, but then this again demonstrates the advantages of using the class models to perform topic adaptation.

4.5 Conclusions

This chapter has shown that a statistically significant improvement in word recognition accuracy can be obtained using a topic-dependent class-based language model with weights computed after the first pass of a multi-pass recognition strategy, relative to a non-adapted model. The best overall results were obtained using 1003 classes and a 50 topic model with weights optimised using the EM algorithm, obtaining a word error rate of 16.9% versus the baseline result of 17.1%. Using topic adaptation, a class-based model can also provide equivalent performance to a word-only model whilst having a smaller memory footprint. The 2003 class 100 topic model obtains a word error rate of 17.7% against the 17.6% of the baseline word model.

The form of the class topic model also makes it feasible to perform experiments with large numbers of topics, such as the 500 used here. Even with these 500 topics the models remain sufficiently well trained, suggesting that the model described here can also be used to perform topic adaptation with only small amounts of domain-specific data. Comparison with other potential forms of topic adaptation model confirms that the principle model proposed in this chapter contains many advantages over various alternative methods, although a method of altering the class map whilst keeping the class history *n*-grams constant also gave lesser gains over the baseline system and so might reward further investigation.

5. Pair-based Class Models

The usual form of a class-based *n*-gram language model was defined by equation 2.13 back in chapter two,[1] derived via some simplifying assumptions from the preceding equation 2.12. A potential weakness of a class model like this, however, is that given its deterministic classes[2] it seems highly unlikely that the choice of class for a word can ever be optimal. As discussed in section 2.2.4, words can have many different meanings and usages. The word `BOW` was given as an example, with around 30 different definitions found in the Oxford English Dictionary. Noting that the general rationale behind a class model is to group together words with similar meanings and that it is highly improbable that there are any other words which combine the same set of 30 different usages, it follows that unless this particular word is placed in a singleton class then the standard class model will not be able to model this word in all of its different contexts. Extrapolating this observation with the fact that the vast majority of English words, in particular the more common ones, have multiple meanings implies that there must be many such approximations forced upon the standard class model.

Having noted this problem, the question remains of how to improve upon the standard model whilst remaining within the deterministic framework.[3] A class *n*-gram language model bases its probability estimates on the recent history of words and classes, so by making better use of this information it may be possible to make more accurate or specific class predictions. Such a prediction could include not only the current word but also the preceding classes or words, or some combination thereof. Making such a decision on the basis of preceding classes has two problems – firstly that the preceding classes themselves still run the risk of being 'incorrect', leading to cumulative errors, and secondly that it adds a feedback element to any word clustering process whereby changing the class of one word affects all class estimates, severely impacting on the

[1] See page 26.

[2] As opposed to a model which allows simultaneous multiple class membership, as for example that given in equation 2.18 on page 29.

[3] Non-deterministic class models present many training and run-time problems and, perhaps as a result, tend to perform badly, as discussed in chapter 2.

computational feasibility of any such process. This leaves the consideration of a word classification process which takes note of recent words rather than classes when deciding upon the class of the current word.

The 'standard' choice of considering only the current word when classifying that word has the great advantage of simplicity and of allowing models to be adequately trained with only limited amounts of training data. Since the entire point of the classification process is to place a word in a class that generally represents its properties, it follows that there must be a sufficient quantity of training data in order to make these deductions meaningful. Furthermore, since most benefit is often gained for rarer words then it seems clear that even with many tens of millions of words of training data there is unlikely to be enough data to usefully use a triplet of words for classification, which by process of elimination leaves consideration of the use of a pair of words.

Perhaps the most obvious choice for a pair of words is the current word and its immediate predecessor. Often the word immediately preceding a given word gives an indication of its part of speech, which can in some cases provide a degree of disambiguation with respect to the usage of that word. Consider the two sentences:

HE LEFT IT BEHIND
and
HE TURNED TO THE LEFT

The word HE in the first sentence indicates that the following word LEFT is a verb, whereas the preceding THE in the second sentence suggests that a noun, adjective or adverb is about to follow. Ambiguity clearly remains, however – the second sentence could for example indicate either a political or a physical change of perspective – but some disambiguation power is gained.

There are many additional sources of information which could help inform a decision on classifying a word, but for reasons of data sparsity – and computational feasibility – the work described here uses solely the immediately preceding word as the context trigger. Such a model is referred to herein by the term *pair-based* model, whilst the previously described single-word class model is referred to for reasons of clarity as the *standard* model.

An additional consideration is the number of classes to be used in a pair-based class model. Conceptually it makes sense to use the same number of classes as has been

found to give the best performance when used with the standard model for a particular situation, since the aim is to improve the classification – the resulting models can then be compared with standard class models with the same number of classes in order to assess any performance gain. In addition increasing the number of classes generally makes class models tend towards the word model performance[4] and requires more training data to be considered 'equally' well-trained. Since in practice word *n*-gram models ubiquitously perform better than class models, it is reasonable to state that it is the performance of a class model when interpolated with a word model that provides the most meaningful assessment of its ability to improve overall system accuracy.

5.1 A Pair-based model

Consider a word probability based on the preceding word, $p(w_i|w_{i-1})$, and define some class map function $G(w)$. Recall equation 2.12, reproduced here:[5]

$$\begin{aligned} P_{\text{class}'}(w_i \mid w_{i-1}) \quad &= \quad P(w_i \mid G(w_i), G(w_{i-1}), w_{i-1}) \\ &\times \quad P(G(w_i) \mid G(w_{i-1}), w_{i-1}) \end{aligned} \tag{5.1}$$

In the standard class model it is assumed that $P(w_i \mid G(w_i), G(w_{i-1}), w_{i-1})$ is independent of $G(w_{i-1})$ and w_{i-1}, and also that $P(G(w_i) \mid G(w_{i-1}), w_{i-1})$ is independent of w_{i-1}, giving:

$$P_{\text{class}}(w_i \mid w_{i-1}) \quad = \quad P(w_i \mid G(w_i)) \times P(G(w_i) \mid G(w_{i-1})) \tag{5.2}$$

Now consider a class map function $H(w_i, w_{i-1})$ which maps a word w_i to a class based not just on that word itself but also on its immediate predecessor, w_{i-1}. Using this class map function a word triplet must be considered in order to map to a class pair, thus:

$$\begin{aligned} P_{\text{pair}'}(w_i \mid w_{i-1}, w_{i-2}) \quad &= \quad P(w_i \mid H(w_i, w_{i-1}), H(w_{i-1}, w_{i-2}), w_{i-1}, w_{i-2}) \\ &\times \quad P(H(w_i, w_{i-1}) \mid H(w_{i-1}, w_{i-2}), w_{i-2}) \end{aligned} \tag{5.3}$$

In a similar fashion to the standard class model, it was decided to assume that $P(w_i \mid H(w_i, w_{i-1}),\ H(w_{i-1}, w_{i-2}), w_{i-1}, w_{i-2})$ is independent of $H(w_{i-1}, w_{i-2})$ and w_{i-2}, and

[4] As the number of classes tends towards the number of words.

[5] Bigrams are used for clarity.

that $P(H(w_i, w_{i-1}) \mid H(w_{i-1}, w_{i-2}), w_{i-2})$ is independent of w_{i-2}. These assumptions are made to keep the model compact and feasible to train, giving:

$$\begin{aligned} P_{\text{pair}}(w_i \mid w_{i-1}, w_{i-2}) &= P(w_i \mid H(w_i, w_{i-1}), w_{i-1}) \\ &\times P(H(w_i, w_{i-1}) \mid H(w_{i-1}, w_{i-2})) \end{aligned} \tag{5.4}$$

This model is similar to the standard class model form given in equation 5.2, except that w_{i-1} must now be included in the first term – if it is not then the word-given-class maximum likelihood probability estimate can only be formulated as:

$$P(w_i \mid H(w_i, w_{i-1})) = \frac{C(w_i)}{\sum_{x,y \in \mathbb{W}:\ H(x,y)=H(w_i,w_{i-1})} C(x, y)} \tag{5.5}$$

where $C(a)$ is the count of event a given some training data set and (x, y) is any word pair 'y x' – that is, x preceded by y. This definition potentially forces an underestimation of most probabilities,[6] whereas if w_{i-1} is available as an individual entity then it is possible to be more precise and allocate all the probability mass:

$$P(w_i \mid H(w_i, w_{i-1}), w_{i-1}) = \frac{C(w_i)}{\sum_{x \in \mathbb{W}:\ H(x,w_{i-1})=H(w_i,w_{i-1})} C(x)} \tag{5.6}$$

To visualise this distinction, consider each preceding word in a pair as a trigger which moves the following word in the pair into a given class. Equation 5.6 uses this trigger, w_{i-1}, to find the total count in the current class given the trigger, whilst the preceding 5.5 does not know the trigger and therefore sums over all counts which can ever be in the class in any context.

Given such a pair model, it is possible to calculate the class bigram probability of any given word by using a triplet of context thus:

$$P(w_i \mid w_{i-1}, w_{i-2}) = \frac{C(w_i)}{\sum_{x \in \mathbb{W}:\ H(x,w_{i-1})=H(w_i,w_{i-1})} C(x)} \times \frac{C(H(w_i, w_{i-1}), H(w_{i-1}, w_{i-2}))}{C(H(w_{i-1}, w_{i-2}))} \tag{5.7}$$

In practice, the second component of this product – the class n-gram probability – is smoothed and approximated using Good-Turing discounting and Katz back off as described in sections 2.4.1 and 2.4.2.

[6] In general it is not able to allocate all the probability mass because the context of each word is unknown.

5.2 Finding a pair class map

Given the definition of a pair model it is necessary to develop a suitable method for training the class map function, $H(.)$. Training involves optimising the map to fit some text, but in order to make this feasible it is necessary to develop a more compact representation of a text corpus than the list W.

The overall probability of a text corpus as predicted by a language model can be calculated by computing the product of all the probabilities of each word in that text corpus given the language model. Taking for example a trigram language model this can be efficiently calculated by counting the occurrence of each word triplet:

$$P(W) = \prod_{w,x,y \in \mathbb{W}} P(w \mid x, y)^{C(w,x,y)} \tag{5.8}$$

where W is the training text list of words $(w_1, w_2, \ldots)$ and $\mathbb{W}$ is the set of all words in W, whilst (w, x, y) is some word triplet '$y\ x\ w$' with $C(.)$ defined as previously as the count of some event.

The result of evaluating equation 5.8 will typically be a number so small that inaccuracies will be introduced by almost any computer system. Taking logs solves this problem:

$$\log P(W) = \sum_{w,x,y \in \mathbb{W}} C(w, x, y). \log \big(P(w \mid x, y)\big) \tag{5.9}$$

This equation allows a relatively compact and efficient method of repeatedly evaluating the performance of a language model by storing a series of counts and recomputing by summing over these rather than by scanning an entire corpus. In addition, if the word triplets are suitably indexed then it is possible when progressively altering a model to only reevaluate those components of the summation which were affected by each specific modification of the model. This increases the computational feasibility of making many tentative changes to the model and keeping only those which give the best overall probability improvement.

5.2.1 Objective function

Language models are typically trained so as to aim to maximise $P(W)$. Equation 5.9 provides a convenient method of evaluating this measure, and so substituting equation

5.7 into this results in an equation for evaluating the class pair language model, using bigram estimates for reasons of computational speed and storage space:

$$\begin{aligned}
\log P(W) &= \sum_{w,x,y \in \mathbb{W}} C(w,x,y).\log\big(P(w \mid x,y)\big) \\
&\simeq \sum_{w,x,y \in \mathbb{W}} C(w,x,y).\log\left(\frac{C(w)}{\sum_{z \in \mathbb{W}:\ H(z,x)=H(w,x)} C(z)} \times \frac{C(H(w,x),H(x,y))}{C(H(x,y))}\right) \\
&= \sum_{w,x,y \in \mathbb{W}} C(w,x,y).\log\left(\frac{C(w)}{\sum_{z \in \mathbb{W}:\ H(z,x)=H(w,x)} C(z)}\right) \\
&\quad + \sum_{g,h \in \mathbb{H}} C(g,h).\log\left(\frac{C(g,h)}{C(h)}\right) \\
&= \sum_{w \in \mathbb{W}} C(w).\log(C(w)) - \sum_{w,x \in \mathbb{W}} C(w,x).\log\left(\sum_{z \in \mathbb{W}:\ H(z,x)=H(w,x)} C(z)\right) \\
&\quad + \sum_{g,h \in \mathbb{H}} C(g,h).\log\big(C(g,h)\big) - \sum_{g \in \mathbb{H}} C(g).\log\big(C(g)\big)
\end{aligned} \tag{5.10}$$

where $\mathbb{H}$ is the set of all classes and (g,h) is some class sequence '$h\ g$'.

Given some initial guess at a definition for the class map function $H(.,.)$ then it is desirable to know how best to improve the class map so as to maximise $\log P(W)$. The expansion in equation 5.10 shows that this value can be expressed as the sum of four separate terms. Note that the first of these terms, $\sum_{w \in \mathbb{W}} C(w).\log(C(w))$, is independent of the class map function $H(.,.)$, therefore it is not necessary to consider it when optimising $H(.,.)$ because it remains constant for a given text corpus. Defining F_{M} as the objective function to be maximised, a method for efficiently evaluating the performance of a particular class map can therefore be specified:

$$\begin{aligned}
F_{\mathrm{M}} &= \sum_{g,h \in \mathbb{H}} C(g,h).\log\big(C(g,h)\big) \\
&\quad - \sum_{g \in \mathbb{H}} C(g).\log\big(C(g)\big) \\
&\quad - \sum_{w,x \in \mathbb{W}} C(w,x).\log\left(\sum_{z \in \mathbb{W}:\ H(z,x)=H(w,x)} C(z)\right)
\end{aligned} \tag{5.11}$$

Although this measure uses class bigram probabilities, equivalent equations for longer n-grams can be derived. In the standard class model case, however, experiments show

that little gain is obtained by using a class trigram optimisation function [Martin, Liermann and Ney 1998].

5.2.2 Initialisation

In order to be able to modify a pair class map an initial class map must first be defined. Rather than starting with all pairs distributed amongst classes using some *ad hoc* algorithm as per the standard class model case, a class map produced by the standard class model clustering procedure can be used for initialisation and then refined by adding in the additional subtleties that can be captured by the pair class model.

A standard class map $G(.)$ can first be constructed using one of the clustering methods described in section 2.3.1, mapping each word w to a class in $\mathbb{G}$. In order to use this map to specify the initial pairmap, $\mathbb{H}$ must be equal to $\mathbb{G}$ – this means that the same number of classes, $|\mathbb{H}| = |\mathbb{G}|$, must be used. This is not a restriction because $\mathbb{G}$ can be chosen before the standard class map clustering so as to match requirements for $\mathbb{H}$.

5.2.3 Optimisation

An iterative word exchange algorithm can be used to refine the pair class map $H(.,.)$. Each word pair is considered in turn and moved to the class which increases F_{M} by the greatest amount. This algorithm is greedy and therefore suboptimal – it does not consider all possible moves by all pairs simultaneously – but a more comprehensive assessment strategy would be computationally infeasible. The algorithm used, therefore, is:

1. **Initialise**: $\forall w, x \in \mathbb{W} : \; H(w, x) := G(w)$
   Set up the class map so that the mapping is identical to that defined by the standard class map

2. **Iterate**: $\forall i \in \{1 \ldots n\} \; \wedge \neg s$
   For a given number of iterations $1 \ldots n$, or until some stop criterion s is fulfilled

   (a) **Iterate**: $\forall w, x \in \mathbb{W}$
   For each word pair (w, x) in the vocabulary

   i. **Iterate**: $\forall g \in \mathbb{H}$
   For each class g

   A. **Move** word pair (w, x) to class g, remembering its previous class

   B. **Calculate** the change in F_{M} for this move

   C. **Move** word pair (w, x) back to its previous class

   ii. **Move** word pair (w, x) to the class which increased F_{M} by the most, or do not move it if no move increased F_{M}

5.2.4 Implementation

This clustering process requires consideration of far more moves than the clustering algorithms for the standard class model described in section 2.3.1 – the $|\mathbb{W}|$ individual words from the single-word case become $|\mathbb{W}|^2$ pairs. Even allowing for partial coverage of this space in any given corpus, there is still a significant increase in execution time for this clustering process of the order of tens of times.[7] The algorithm does, however, lend itself to a distributed parallel implementation since each iteration of the innermost loop (2(a)i) is independent of the outer loops – each of these tentative moves can therefore be executed in parallel.[8] No such parallel implementation was attempted for this

[7] Found empirically and relative to the similar word exchange implementation.

[8] Since the vast majority of execution time is spent in this inner loop it follows that the speedup for adding n extra processors would be near to $(n + 1)$ times, assuming independent memory storage for each processor.

work, however.

In step (2a) all word pairs (w, x) in the vocabulary are iterated over. An optimisation which is applied is to order the pairs by frequency of occurrence in the training text, such that the most frequently-occurring pair is considered first and the least frequent pair last. This means that the amount of probability mass moved with each change in the definition of $H(.,.)$ decreases on a per-move basis as step (2a) iterates through all pairs. Moving the less frequent pairs is likely to provide smaller probability gains due to the reduced mass, and so it is reasonable to suppose that given this ordering there will be some cut-off point at which worthwhile gains are no longer obtained and the current iteration of step (2a) should be ended. Indeed, at some point the counts will become so low that their statistical significance will be in question, so continuing the iteration for too many pairs will risk over-fitting the training data and therefore potentially damaging the model's performance.

When a word pair is moved from one class to another, as happens for each tentative move, then not all class pair counts are affected. Update equations can be derived to ensure that as little unnecessary work as possible is done when performing each of these tentative moves.

At the start of each round of tentative moves some values can be precomputed:

$$\begin{aligned}
\forall j \in \mathbb{H} : a[j] &= \sum_{y,z:H(y,z)=j} \begin{cases} C(w,x,z) & : \quad x = y \\ 0 & : \quad x \neq y \end{cases} \\
\forall j \in \mathbb{H} : b[j] &= \sum_{y,z:H(y,z)=j} \begin{cases} C(y,w,x) & : \quad w = z \\ 0 & : \quad w \neq z \end{cases} \\
c &= \begin{cases} C(w,w,w) & : \quad w = x \\ 0 & : \quad w \neq x \end{cases}
\end{aligned}$$

where $C(x, y, z)$ is the count of the word sequence '$z\ y\ x$' and j is a class.

When a word pair (w, x) (that is, the word w preceded by x) leaves class g and enters class h then the effect on the class counts can be computed via the following

update rules:

$$\begin{aligned}
C(g,g) &\Rightarrow -a[g] - b[g] + c \\
C(h,h) &\Rightarrow +a[h] + b[h] - c \\
C(g,h) &\Rightarrow -a[h] + b[g] \\
C(h,g) &\Rightarrow +a[g] - b[h] \\
\forall j \in \mathbb{H}, j \neq g, j \neq h : C(g,j) &\Rightarrow -a[j] \\
\forall j \in \mathbb{H}, j \neq g, j \neq h : C(j,g) &\Rightarrow -b[j] \\
\forall j \in \mathbb{H}, j \neq g, j \neq h : C(j,h) &\Rightarrow +b[j] \\
\forall j \in \mathbb{H}, j \neq g, j \neq h : C(h,j) &\Rightarrow +a[j] \\
C(g) &\Rightarrow -C(w,x) \\
C(h) &\Rightarrow +C(w,x)
\end{aligned}$$

The sum of all the elements on the right-hand sides of these update rules can be seen to be zero – this is because the overall count of the sum of all events remains constant due to the deterministic nature of the classification. Having computed these updated counts, F_{M} can be reevaluated[9] and the change in its value computed.

5.2.5 Data structures

The specific implementation used for the work reported here assumes that the class of a word is in general independent of its context, as in the standard class model, and so instead of a complete pairmap it stores a single-word class map as per the standard algorithm and adds to this an explicit list of *exception pairs* and their classes. This allows the efficient storage of the relatively small number of all the possible pairs which are actually created by the clustering algorithm when starting from a standard class map.

In order to achieve fast access to word triplet counts for the purposes of implementing the above update rules, two separate indexes were created allowing fast retrieval of word triplet counts with a given word w in the most recent position, $(w,?,?)$, or in the preceding word position, $(?,w,?)$. These linked into a tree structure descending down from the most recent word, with cumulative counts at all nodes, both internal and

[9] By recalculating the components of the summations in equation 5.11 on page 109 that have altered in value.

external, providing fast access to singleton, pair and triplet word counts. Entries were sorted by an internal word id value allowing a binary search to be used at each node for efficient retrieval.

5.3 Experiments – pair model only

All reported experiments in this chapter use 1003 classes.[10] This value was chosen because previous work, including that in the preceding chapter, showed that this number of classes resulted in class language models with the best performance when interpolated with a separate word *n*-gram language model.[11] As in the previous chapter, language modelling and lattice rescoring work was performed using the HLM utilities [Young et al 1997] and the Lattice Toolkit [Odell et al 1996], specially modified for the experiments described here.

Two iterations of the standard word exchange class clustering algorithm were run first in order to find the starting class of each word, $G(.)$, also referred to here as its *default* class.

5.3.1 Corpora

Experiments were run with the large corpus as defined in section 4.2, denoted here as the 'Full' corpus.[12] The full 65,425 word vocabulary was used and perplexity was assessed on the 1997 HUB-4 evaluation reference transcriptions, as before.

A much-reduced[13] corpus was also used but based only on 1992 broadcast news and a more limited vocabulary. The 1992 Broadcast News corpus, described here as the '1992' corpus, used all available broadcast news transcriptions from 1992 with the exception of those from the month of November which were held out for use as a test set. The training text had over 16.25 million words (over 920,000 sentences), with the test text

[10] All major experiments were also run with both 503 and 2003 classes but similar trends to those described throughout this chapter were observed so these models are not discussed further.

[11] Using a vocabulary of around 65,000 words.

[12] A mixture of broadcast news and newswire text with 144 million words of Broadcast News text and 25 million words of Los Angeles Times / Washington Post (LATWP) newswire text, plus double-weighted Marketplace and Broadcast News precise acoustic transcriptions.

[13] But still large in general terms.

containing almost 1.4 million words and over 78,000 sentences. A vocabulary was defined for the models built using the corpus that consisted of the most frequent 30,000 words from the training set.

All experiments reported here are models with *n*-gram cut-offs in both the word and class cases of 1, 3 and 3 for 2-, 3- and 4-grams respectively, with Good-Turing discounting and Katz back off.

5.3.2 Perplexity – bigram

As the pair clustering algorithm executes, the perplexity can be calculated after each move – when using a bigram model the perplexity on the training set is guaranteed to decrease,[14] but a fair assessment is to compute the perplexity on the withheld test set. Figure 5.1 shows the effect on the test set perplexity as the algorithm runs for the first few hundred moves using the 1992 corpus with a bigram pair class language model.

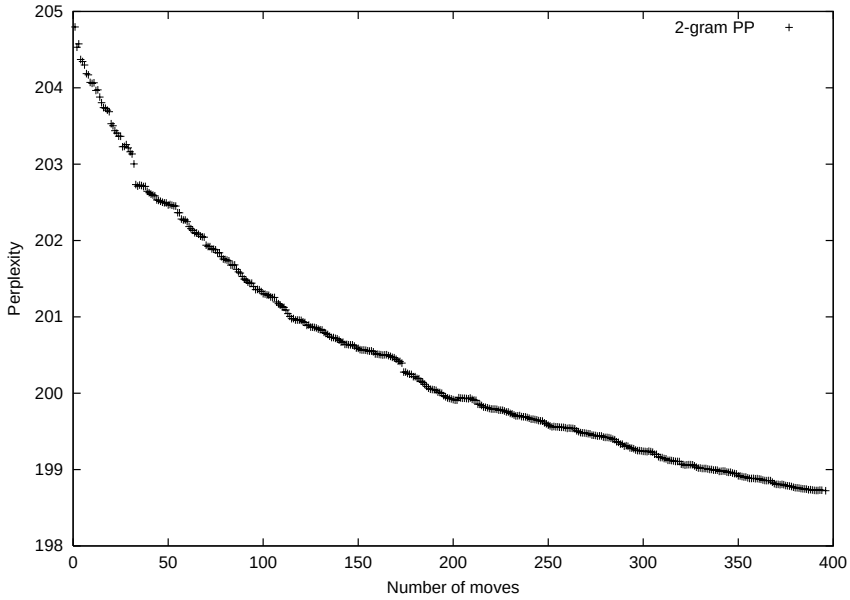


Figure 5.1: Change in test set perplexity when moving pairs from their start location when assessed using a bigram class pair model [1992 corpus]

The figure shows that the perplexity falls as pairs are moved from their default classes

[14] Assuming no cut-offs, discounting or other smoothing or approximation methods are applied.

as specified by the class map taken from the standard class language model. It is perhaps not surprising that the earlier moves have a greater effect on the perplexity due to the fact that the pairs are considered in order of decreasing occurrence count in the training text. As discussed earlier, less probability weight is being moved within the model as the counts decrease, and similarly word classification changes are likely to affect less of the test corpus as the contexts become progressively rarer.

If the algorithm is run for many more moves figure 5.2 shows how the perplexity on the test set changes. It can be seen that the gains become less and less, and in fact the model finally starts to become overtrained and performance begins to degrade slightly. Each data point in the graph is separated by 24 hours of CPU time. Table 5.1 shows the relationship between hours of CPU time, the number of pairs moved and the number of pairs actually considered for moves, plus also the percentage of all the mass in the language model that the moved pairs accounted for. The perplexity values plotted in figure 5.2 are also included for convenience.

A particularly interesting feature of figure 5.2 is the comparison with the performance of a bigram word model, with the pair model capable of obtaining much lower perplexities – this point is discussed later in more detail in section 5.4.2.

Table 5.1 reveals that, in the 1992 bigram case, running the algorithm for more than around 30,000-40,000 moves made does not provide any worthwhile gains over shorter runs, and indeed the gains made within the first several thousand moves are much greater than those obtained during later periods. As the algorithm progresses the amount of mass moved with each change to H decreases, indicating that the confidence attached to the move must also be decreasing. This suggests strongly that a cut-off point should be applied and either the algorithm terminated or a new iteration begun once no further worthwhile gains are being obtained. Indeed, after 130,000 moves a minor increase in test set perplexity is noted. An iteration cut-off point could be specified in terms of CPU execution time, but this would be inconsistent across processor architectures and data sets. Two alternative metrics are the numbers of moves considered and the number of moves made. The former of these has the disadvantage that it may need scaling with the iteration count due to the fact that subsequent iterations will reconsider pairs, whilst the latter avoids this problem and has the advantage that it is the only of these metrics which has the ability in all cases to prevent an iteration from ending when no change has actually been made! It seems reasonable, therefore, to base a decision on when to restart an iteration on this metric – this will be examined

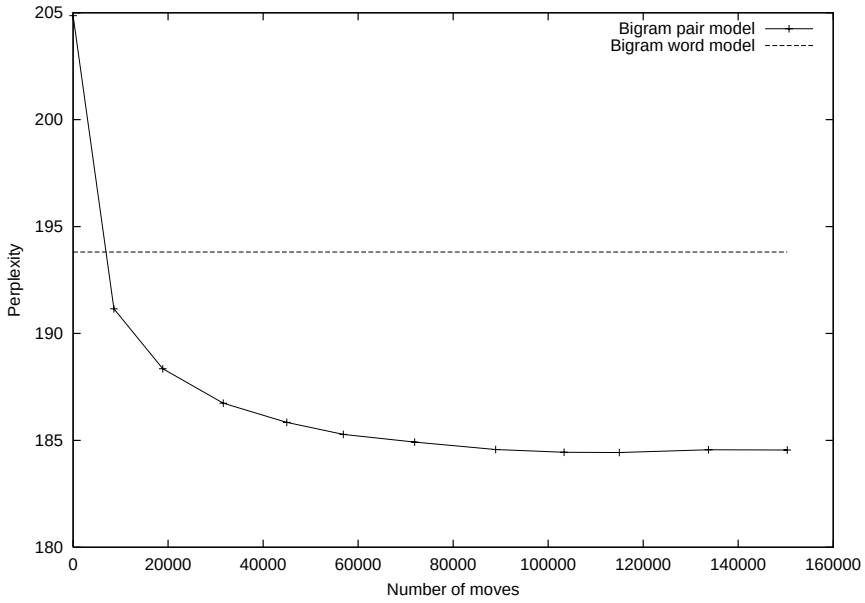


Figure 5.2: Effect on test set perplexity when moving pairs from their start location when assessed using a bigram class pair model [1992 corpus]. Each data point is separated by 24 hours of CPU time.

in section 5.6 later in this chapter.

5.3.3 Perplexity – higher order *n*-grams

How well does the algorithm perform with higher order *n*-grams, specifically 3-grams and 4-grams? Figure 5.3 shows the effect on the test set perplexity when adding pairs to a trigram language model. As can be seen the perplexity increases slightly, suggesting that the model performance is degrading as pairs are added. The performance over a longer period is shown in figure 5.4. Similar curves are seen in both cases when running a 4-gram test.[15]

Why does the perplexity get worse over the first 24 hours? The training algorithm optimises the pair map using a bigram class model and in the bigram test case this optimisation is shown to work well. This suggests that the model is able to correct deficiencies in the bigram model, but perhaps these are absent or less pronounced in the

[15] See figure F.1 in appendix F.

24 hour periods	Moves made	Moves considered	Mass moved (%)	Bigram PP
0	0	0	0	204.86
1	8599	35500	15.6	191.15
2	18875	67500	17.7	188.35
3	31609	102500	19.1	186.74
4	44968	136500	20.1	185.84
5	56877	167500	20.8	185.28
6	71897	200000	21.5	184.92
7	88956	234500	22.2	184.57
8	103350	269500	22.6	184.44
9	114982	298500	22.9	184.43
10	133750	331000	23.4	184.56
11	150312	362000	23.8	184.55

Table 5.1: Number of moves made and considered plus total probability mass moved after each 24 hour period of the pair clusterer algorithm, plus test set perplexity. (Processes logged data at multiples of 500 considered moves)

trigram model – the pair map may in fact be too specific for the more accurate trigram model and this could be leading to a decrease in performance. If this is the case then interpolating with a word model, which as discussed earlier is in practice the most useful assessment of the performance of a class model, may fix this problem by allowing the cases where the moved pairs worsen the language model to be smoothed with the probabilities from the word *n*-gram. It should be noted that although the bigram pair model bases its predictions on a word triplet it uses only pair-based estimates to produce probabilities, much like the standard class bigram. They also have the same order of parameters, unlike the significantly larger trigram models.[16]

Additionally, the question arises as to why the bigram clustering does not generalise to the trigram case with the pair class model, but it *does* in the standard class model case. The reverse of the above argument can be used – perhaps in the standard case the bigram clustering makes more general, less specific decisions. Certainly a single-word classification metric is less specific than a pair-word scheme, although a pair-word

[16]The trigram model adds 569,354 additional parameters despite the trigram cut-off of 3 – the pair model obtains significant improvements over the standard bigram with just a few hundred extra parameters.

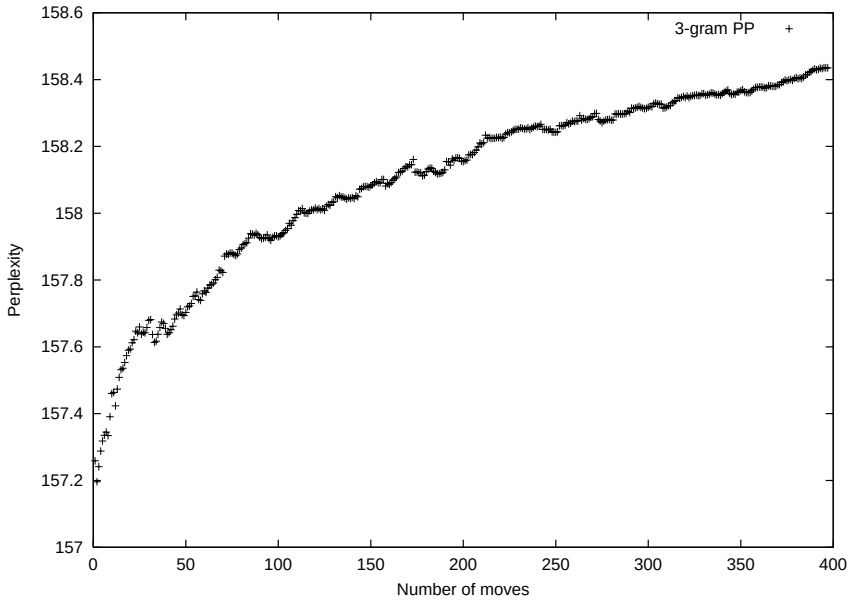


Figure 5.3: Effect on test set perplexity when moving pairs from their start location when assessed using a trigram class pair model [1992 corpus]

method incorporates *all* the power of the single-word scheme and so given suitable training it *must* perform at least as well. Unfortunately in general the only training which can ever be guaranteed to provide this is to train on the test text, which defeats the entire point of having a language model in the first place! The obvious extension is to use a trigram clustering method in the pair case when building trigram models, but unfortunately due to the amount of data and computation required (full unpruned 4-gram storage of the corpus for the trigram word context, for example) this is not practical, at least not with a large corpus – and it is with a large corpus where longer *n*-grams are useful.

Another question which arises is why after about 10,000 moves the model starts to improve again in the trigram case, and then after about 60,000 moves begins to deteriorate again. The latter deterioration occurs for the same reason it did in the bigram case, with the algorithm beginning to over-fit the training text, but the initial peak requires further examination. The fact that the change in perplexity is greatest for the initial moves is not a surprise, due to the largest pair counts being considered first. These initial increases in perplexity must be because these pair moves provide information which

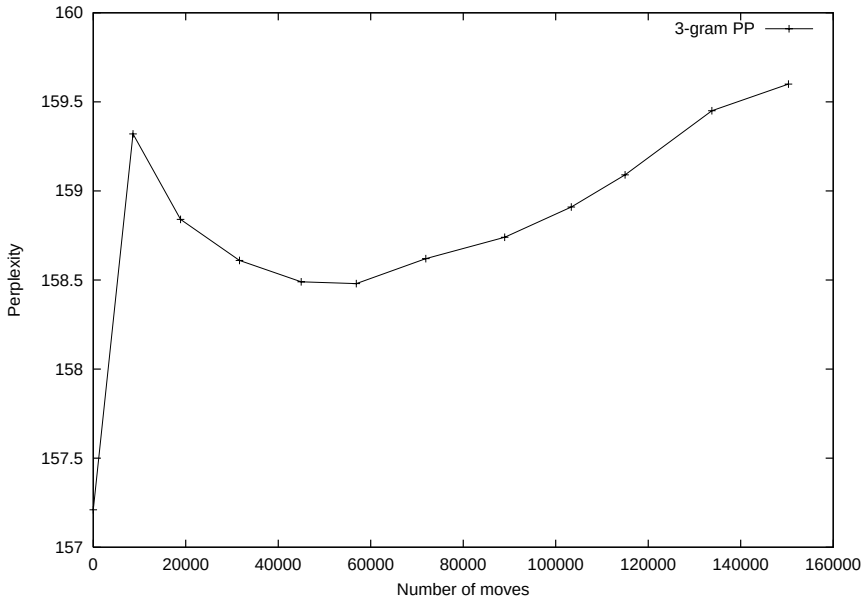


Figure 5.4: Effect on test set perplexity when moving pairs from their start location when assessed using a trigram class pair model [1992 corpus]. Each data point is separated by 24 hours of CPU time.

is already well modelled by the trigram model and so the act of making them more explicit in the pair case leads to the model being less general, therefore when it makes a mistake a performance hit is taken which outweighs the performance gains. As the amount of mass associated with each pair begins to decrease, however, the words are progressively less well modelled by the standard class model and so the pair model has a chance to step in and improve the estimates, as evidenced by the drop in perplexity after around 10,000 moves. This again suggests that interpolating with a word model might reduce the effect of the parts of the pair model which now perform more poorly. This also suggests that the pair model might be a better method for improving smaller, more poorly-trained models than larger, well-trained models.

5.4 Experiments – word and pair models

Figure 5.5 shows the effect on test set perplexity of running the pair clustering algorithm when assessed using a bigram pair model linearly interpolated with a bigram

word model, with weights of 0.4 for the pair model and 0.6 for the word model, chosen based on initial development work. The figure shows that the gains in perplexity over the standard class model previously observed when using the standalone pair model are preserved when interpolating with a word model, although the previous reduction of 10.0% drops to 6.4%. This is as might be expected – interpolation with a word model tends to reduce the gains of a stand-alone model due to the duplication of information; since the gains do not vanish completely then the pair model does actually complement the word model. The more heavily-weighted word model not only lessens gains, however, but can lessen losses – figure 5.6 shows a similar word-interpolated result when using the 3-gram and 4-gram pair models. The previously-observed decrease in performance in the pair model-only cases are transformed into gains in perplexity. These results confirm that the word model provides a method of smoothing the more poorly performing parts of the pair model, obtaining a net gain.

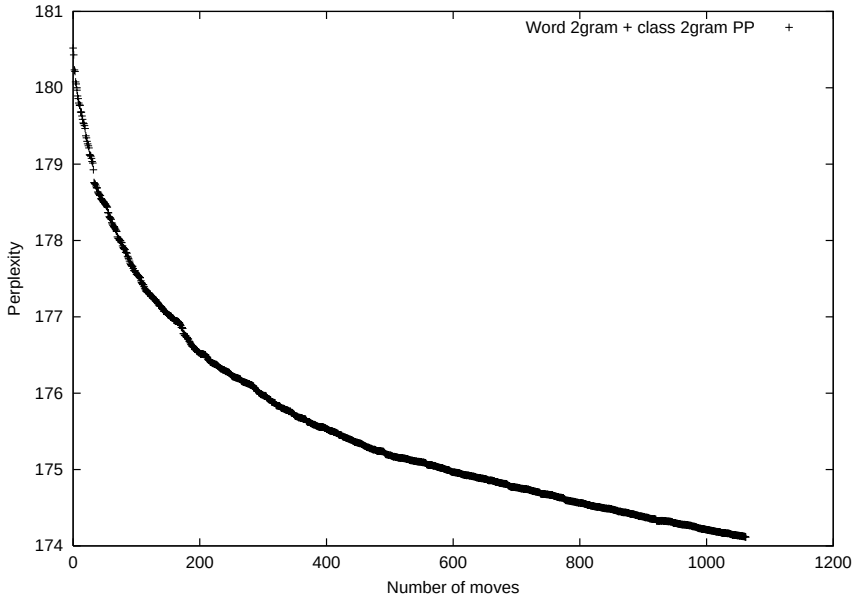


Figure 5.5: Change in test set perplexity when moving pairs from their start location, assessed using a bigram class pair model interpolated with a bigram word model [1992 corpus]

Figures 5.7 and 5.8 add many more moves to figures 5.5 and 5.6 respectively, showing the performance of the interpolated language models over many more moves. As can be seen, the majority of the gain – a reduction of about 1.5% in each case – is obtained in

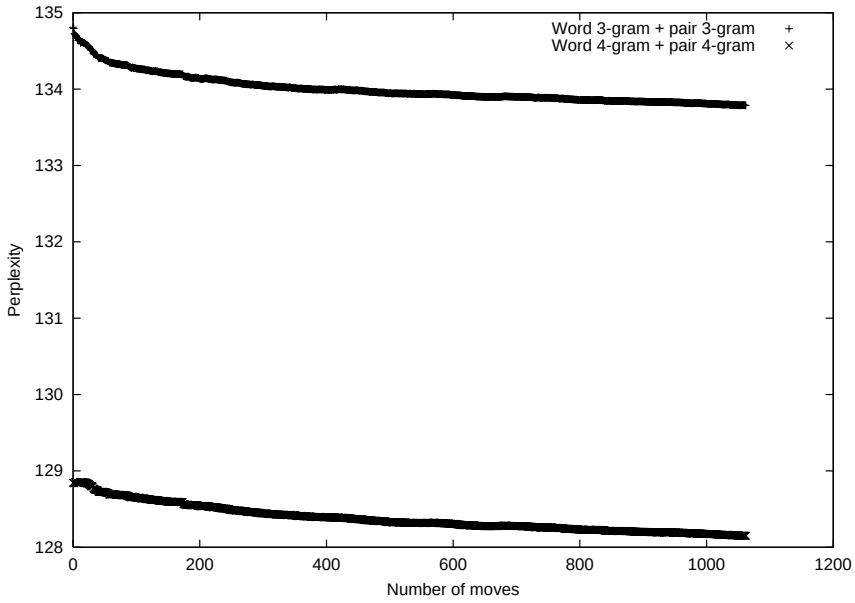


Figure 5.6: Change in test set perplexity when moving pairs from their start location, assessed using 3-gram and 4-gram class pair models interpolated with 3-gram or 4-gram word models respectively [1992 corpus]

the first 10,000 moves. After 20,000 to 30,000 moves the performance starts to degrade as observed in the standalone pair model case. This is due to an over-fitting of the training data, using pair count observations which were not found frequently enough in the training data to have any confidence in.

5.4.1 Standard class model

One question which must be asked is whether the performance of the standard class model could also be improved upon simply by performing an additional iteration of the standard word exchange clustering algorithm, which in the experiments reported here had already been run for two complete iterations in order to find an initial start state for running the pair model clustering algorithm. Figure 5.9 contrasts the effect of changes to the class map G in the class case with changes to H in the pair case for bigrams[17] – the moves in both cases were considered in order of decreasing count to

[17] A similar graph for the 4-gram case is given in appendix F – figure F.2.

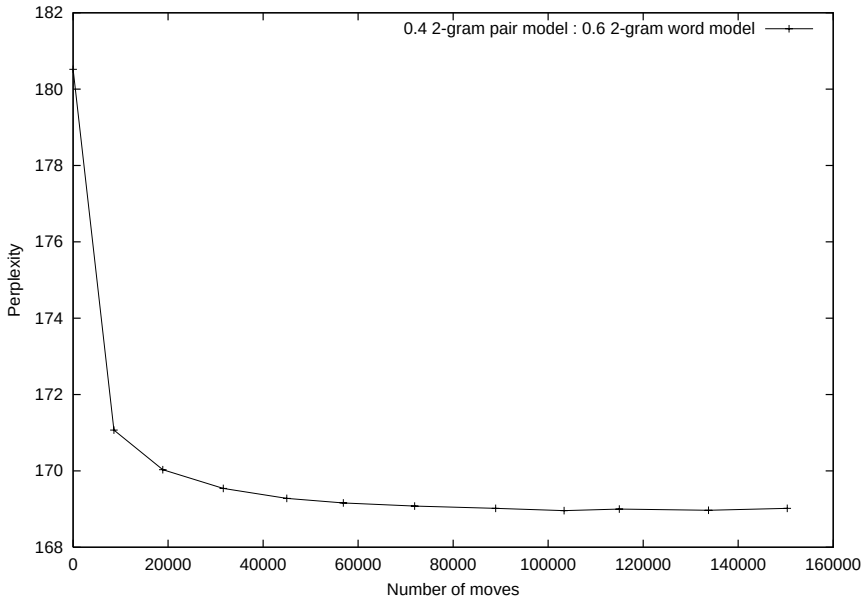


Figure 5.7: Effect on test set perplexity when moving pairs from their start location when assessed using a bigram class pair model interpolated with a bigram word model [1992 corpus]. Each data point is separated by 24 hours of CPU time.

ensure a fair comparison. A word-model was interpolated, with weights of 0.4 and 0.6 again used for the class:word model linear interpolation ratio.

The standard class model performance remains basically constant as moves are made, especially in the bigram case, whilst the pair model performance rapidly improves. This demonstrates that the gains observed in the pair model case are indeed true gains which the standard model does not obtain.[18] In fact in figure 5.9 it is perhaps surprising just how constant the standard model performance remains – what in the printed figure looks like a thick single line is in fact almost 5000 overlapping data point crosses!

The effect on perplexity of altering the interpolation parameter weight of the class model relative to the word model is shown in figure 5.10[19] with the 1992 corpus. The same result for the standard class model after two iterations is included for comparison.

[18]Or at least not with the same word exchange training method, although there is no reason to suspect that other methods would give different results.

[19]A similar graph for a trigram pair model is given in figure F.3 in appendix F.

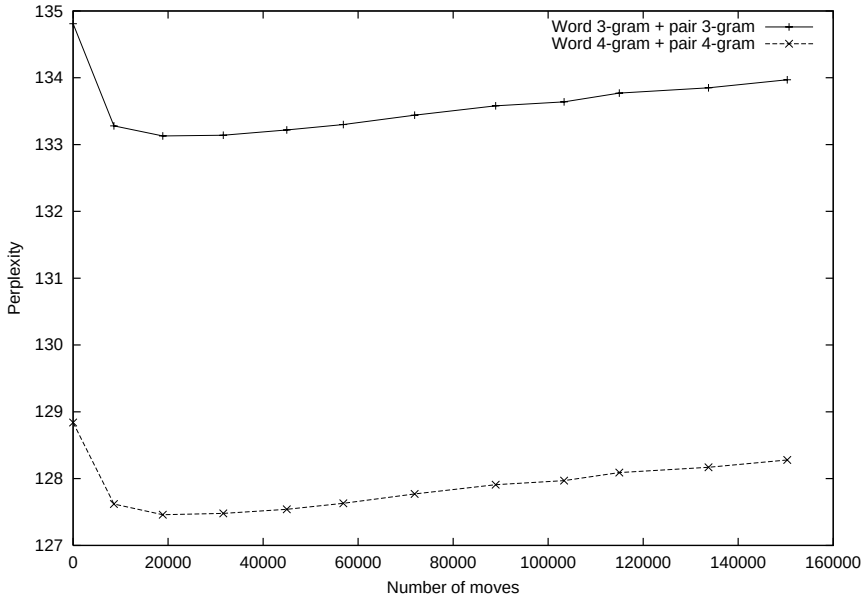


Figure 5.8: Effect on test set perplexity when running pair clustering algorithm – 3- and 4-gram models interpolated with word model [1992 corpus]. Each data point is separated by 24 hours of CPU time.

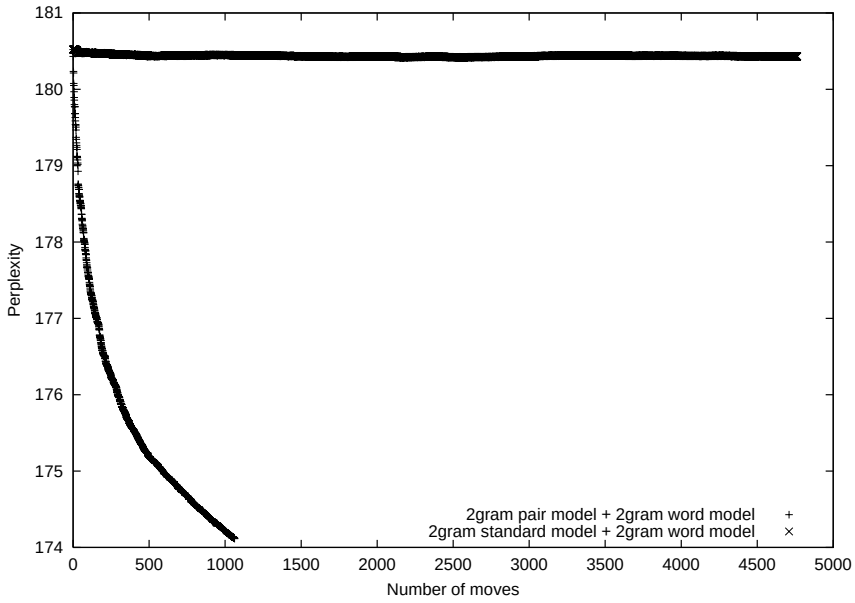


Figure 5.9: Perplexity comparison on a per-move basis between bigram pair and standard class models – each interpolated with a word bigram [1992 corpus]

As can be seen, it remains the case that the interpolated model is not overly sensitive to the interpolation weights – near the optimal setting the weight of each model can be altered by around 10% of the total weight either way without having much effect on the final perplexity. The pair map H used was obtained after 24 hours of execution of the clustering algorithm.

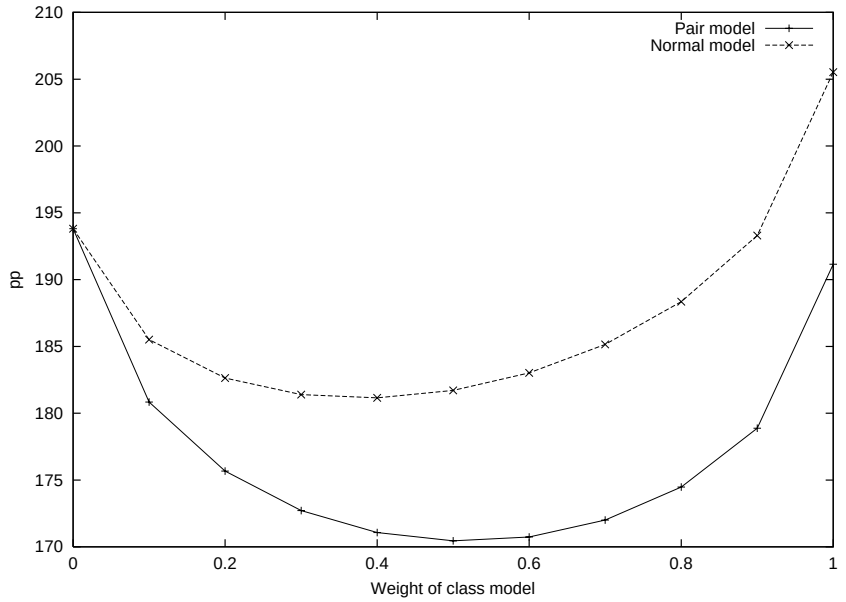


Figure 5.10: Test set perplexity when linearly interpolating bigram word and bigram class pair models together with different weights [1992 corpus]

5.4.2 Further discussion of results

Table 5.2 on the following page summarises the best results from the experiments reported so far in this chapter and provides a comparison with equivalent results for the standard class and word models. One of the best results is the improvement in the performance of the class pair bigram model over the word bigram model by almost 5% despite having significantly less parameters[20] – the word bigram has 55% more bigram parameters than the 227,147 of the class pair bigram model. Although the class model requires an additional 30,001 parameters to store the default class of each word, the

[20] All models use a cut-off of 1 for the bigrams. The 10.0% improvement over the standard bigram class model is also a good result.

word model contains 30,001 unigram probabilities versus the 1,003 in the class model; the latter, however, requires 30,001 word counts. The pair model also requires a list of exceptions to the default classes – less than 10,000 of these are required to beat the word model performance, so the resulting model is a lot more compact; see figure 5.2. In fact since there are only 1,003 class model 'word' tokens against the 65,425 of the word model, the basic underlying bigram counts can potentially be stored more compactly in the class model case given equivalent quantities of these. Overall, then, the pair class model can obtain a significant improvement in perplexity over the model with a much smaller footprint.

	Word model	Class model		
n-gram size	Weight	Type	Weight	Test set perplexity
4	1.0	-	-	144.04
3	1.0	-	-	149.93
2	1.0	-	-	193.81
4	-	Standard	1.0	150.32
3	-	Standard	1.0	157.21
2	-	Standard	1.0	204.86
4	-	Pair	1.0	150.32
3	-	Pair	1.0	157.21
2	-	Pair	1.0	184.43
4	0.6	Standard	0.4	128.95
3	0.6	Standard	0.4	134.90
2	0.6	Standard	0.4	180.53
4	0.6	Pair	0.4	127.46
3	0.6	Pair	0.4	133.13
2	0.6	Pair	0.4	168.96

Table 5.2: Overview of perplexity results on BN1992 corpus – best results obtained from experiments reported here

5.5 Experiments – Full corpus

Experiments were also performed which are directly comparable with those reported in the previous chapter, using the full training corpus and the same test set as was used

previously – the test set used to calculate the perplexity consists of all the reference transcriptions for the 1997 HUB-4 evaluation set, as described in chapter 4.

In addition to perplexity calculations, lattice rescoring experiments were also undertaken in order to obtain word error rates for a thorough assessment of the performance of the models.

5.5.1 Pair-model only

Figure 5.11 shows the effect on the test set perplexity of making moves using the pair model clustering algorithm. In the bigram case worthwhile gains are obtained – with just over 3000 pair moves made a reduction in perplexity of around 5% is observed. In the trigram and 4-gram cases, however, the perplexity again rises as moves are made, although it does level off after around 1000 moves.

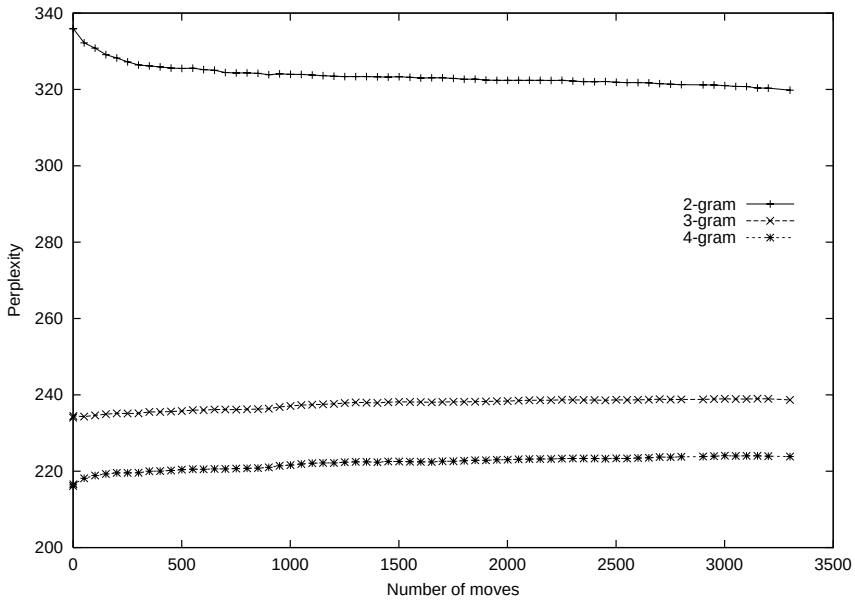


Figure 5.11: Effect on test set perplexity when moving pairs from their start location when assessed using 2-, 3- and 4-gram class pair models [Full corpus]

Considering the bigram case, it is interesting to analyse what improvements are being made to the model by using the pair map rather than the standard class map. Figure

5.12 compares the log probability of each word in the test set when assessed using each of the two models and counts how many probability differences fall in each of a range of log probabilities.[21] Each bin in the histogram is 0.03 wide. The first point of note is that the largest gains are of greater magnitude than the largest losses, suggesting that although some pairs cause a reasonably big drop in performance when they are used inappropriately they cause an even larger gain when they 'hit'. There are also no exceptionally large losses of several orders of magnitude, suggesting that even when a pair performs badly it never performs *so* badly that it hugely distorts the overall average performance. Around the zero-change point there is more mass on the gain side, so the average change in the pair model is a small gain, which is confirmed by the perplexity results in figure 5.11. Overall, the distribution is roughly symmetrical around a single narrow peak, however, so most pairs have a relatively small effect on the word probability relative to the standard class model, whether positive or negative.

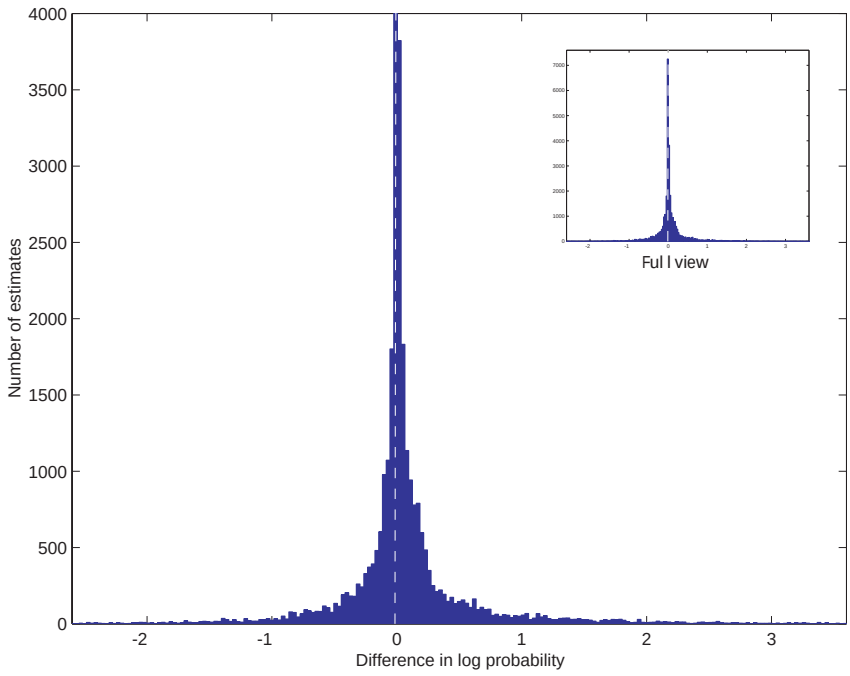


Figure 5.12: Number of changes in test set word log probabilities of various magnitudes – change from a bigram standard to a bigram pair class model

[21] It is directly comparable with figure 2.1 on page 55. The pair map used was found after 24 hours of running the pair clusterer.

Figure F.4 in Appendix F shows a similar graph for the 4-gram case – it also shows an almost completely symmetrical, narrow peak, suggesting that the 4-gram behaviour is similar to the bigram performance. The main change is that there are some losses of greater magnitude and less small gains. The overall shift to a very minor decrease in overall perplexity (barely perceptible in the histogram) is most likely to be due to some pair gains being lost due to the pairs providing redundant information when moving words to the 'correct' class which is already implicitly modelled by the 4-gram, whilst when those pairs are in a context which assigns words to the 'wrong' class they do negatively impact on the model performance. If this analysis is correct then a method of culling pairs which perform badly in the 4-gram case could be useful. However a later experiment along these lines failed to improve performance.[22]

The distribution of probabilities within the pair and standard class models can also be compared. Figure 5.13 shows a plot of the number of estimates within various log probability ranges on the test text for the standard and pair class bigram models,[23] revealing some interesting information about the change between the two models.[24] The very lowest probabilities have a similar distribution between the two models. This result is not wholly unexpected because many of the lowest probabilities represent the parts of the model which are based on very small amounts of training data, and since the pairs are created in decreasing order of their frequency within the training text it is unlikely that many pairs will be created which directly affect the words involved in these estimates.

The figure also shows that the gain in performance is obtained within the main bulk of the probabilities where there is a clear general shift towards higher probabilities – there are about 25% more probabilities in the -1 to 0 log probability range in the pair model, for example. There is also a significant gain in the -2 to -1 range, proving that the pair model is making many more confident predictions than the standard model.

[22]Using pair maps constructed for work reported later in this chapter, the pairs which had not changed class between the worst-performing and best-performing pair maps were returned to their default class. Unfortunately this had no effect on the recognition results and led to a small increase in perplexity.

[23]Figure F.5 in appendix F shows a corresponding graph for the 4-gram case.

[24]The area under the graph, weighted by the probability, is proportional to the perplexity.

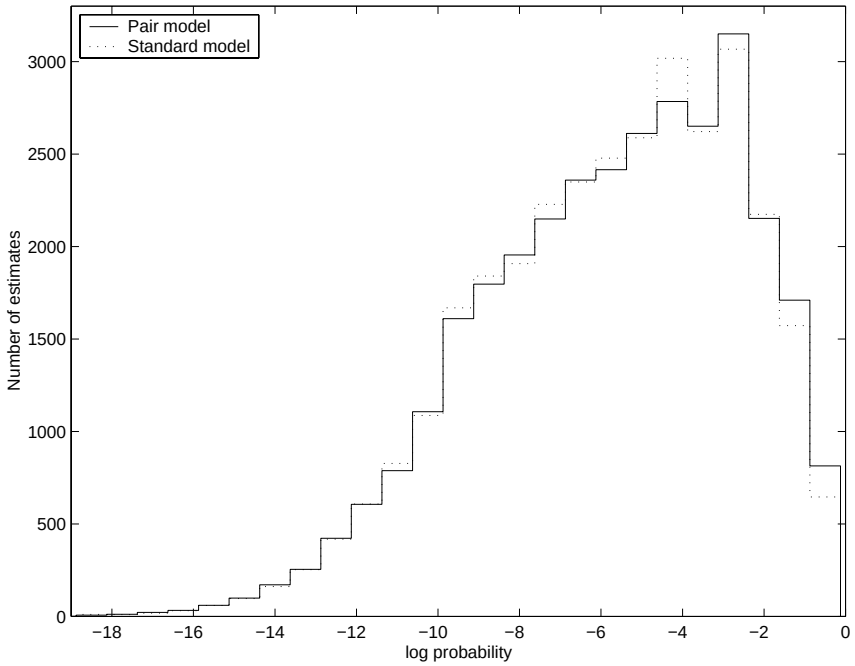


Figure 5.13: Distribution of word probabilities on the test text in pair and standard bigram class models [Full corpus]

5.5.2 Pair model and word model

The performance when interpolating the class and word models was assessed using the same linear interpolation weights as were used in the previous chapter, with 0.35 for the class model and 0.65 for the word model. In addition the same word model was used. Interpolating with the word model, figure 5.14 shows that the gain in the bigram case over the standard class model is preserved, as before, providing a percentage improvement of 4.1% in terms of perplexity that is close to the uninterpolated class pair model's 4.8% gain over the standard class model, again suggesting that the model is capturing information that is not well modelled by the word bigram.

In the trigram and 4-gram cases the perplexity stays fairly constant, replacing the small gains observed previously in figure 5.6. This change can be accounted for by the better trained *n*-grams used in the larger standard class model making it hard to achieve as many gains as the pair model was previously able to make. The initial class map used, which was the same as that used in the previous chapter, is necessarily different

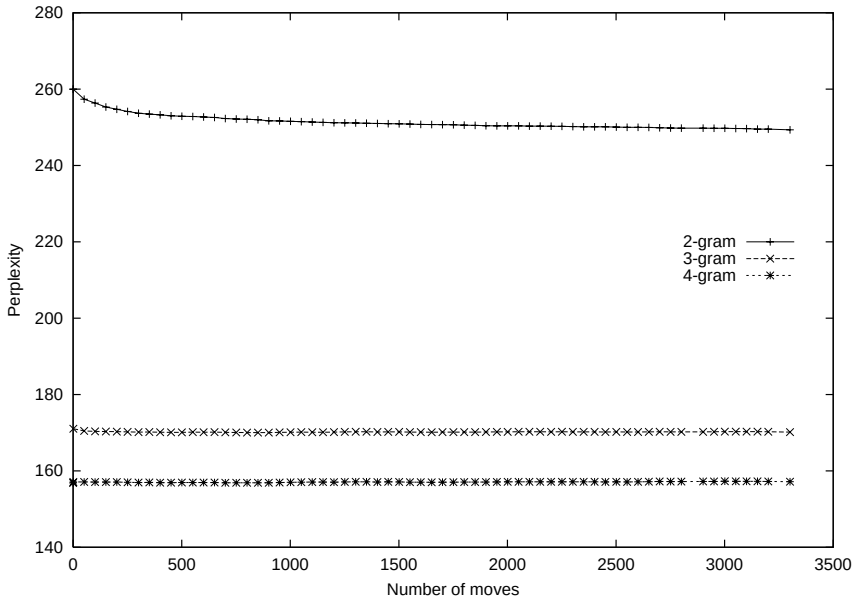


Figure 5.14: Effect on test set perplexity when moving pairs from their start location: 2-, 3- and 4-gram pair models interpolated with corresponding *n*-gram word models [Full corpus]

to that used with the 1992 corpus due to the change in the vocabulary size, so it is also possible that the standard model class map is by itself better constructed,[25] again making it harder to improve upon. But there is not only this difference in training set size to make the goal of improvement more difficult to obtain – there is also more of a mismatch between the training text and the test text, given that the test text consists of the precise acoustic transcriptions from the reference utterances and only a small component of the training text consists of precise acoustic transcriptions. As noted in the previous chapter there is also a slight vocabulary mismatch, which is likely to penalise a pair-based model more than a standard one.

5.6 Multiple Iterations

The results so far indicate that if the pair clustering algorithm is run for too many moves then performance begins to level off and then degrade, so at some point the

[25]Even allowing for the more than doubling of the vocabulary size, there is much more training text.

current iteration should be aborted and a new one begun. Methods of implementing this were discussed earlier in this chapter in section 5.3.2, and it was concluded that basing a decision on the number of moves of a pair from one class to another was reasonable. Consulting figure 5.14 on the previous page suggests that restart values of the order of up to a few thousand moves made would be most sensible.

5.6.1 Bigrams

Figure 5.15 shows the effect on the bigram class model-only test set perplexity when starting a new iteration of the pair clustering algorithm once a given number of moves have been made – restart values of 100, 500, 1000, 1500, 2000 and 3000 are shown. This graph illustrates that starting a new iteration often leads to additional improvements in perplexity. The size of the improvements does begin to decrease with the number of iterations, however, and clearly at some point a limit has to be reached. Although the graph appears to show that the lower restart values are worse, it is important to note that the horizontal axis is 'iterations complete', so for example the 15th point for the restart-at-100 line is after the same number of moves as the first point for the restart-at-1500 line. These points correspond to perplexities of 322.86 and 323.32 respectively, so the restart-at-100 model performs slightly better after the same number of moves in this case. Because, as discussed in section 5.2.5, the language model is stored as a set of pair exceptions, this means that in this case the better-performing language model uses no more parameters than the other – in fact, there is in practice a further gain because some of the moves on subsequent iterations are 're-moves' of pairs which have already changed classes in previous iterations.

The total number of pair moves considered but not made during execution of the algorithm increases as additional iterations of the algorithm are run – for example in the restart-at-500 case the first 500 moves involved considering 2,045 pairs whilst by the 21st interation the number of pairs considered in order to find 500 worthwhile moves[26] had grown 1,650% to 33,811 pairs. This value is directly proportional to the length of time the algorithm takes to execute, so there is a trade-off in CPU time to be balanced, suggesting that too low a restart value is inefficient – but if only the final model size and performance is of interest then this issue is far less relevant. Figure 5.16 shows figure 5.15 replotted on a horizontal scale of number of moves made, where each data point represents the end of an iteration. This graph allows performance comparisions

[26] As defined by the pair clustering algorithm.

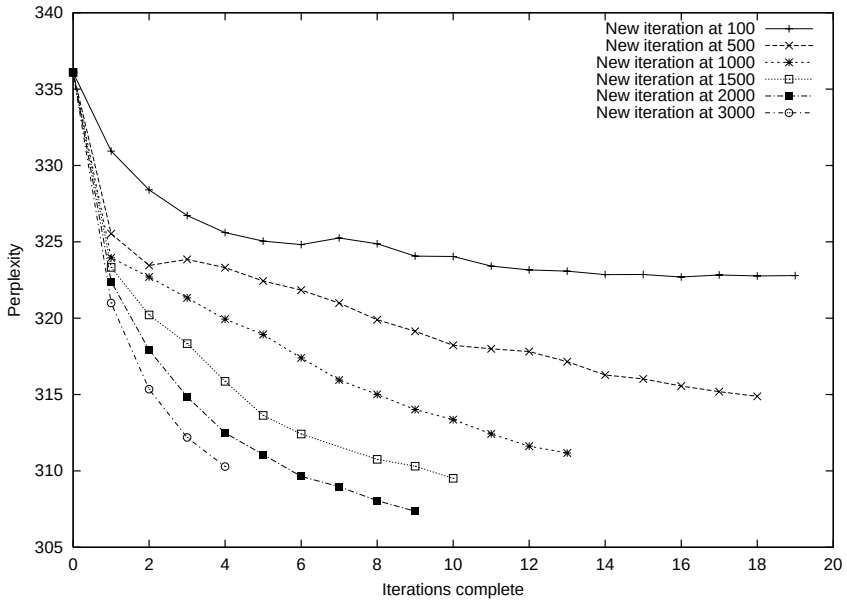


Figure 5.15: Bigram pair-model perplexity on test set when starting a new iteration after a given number of moves – plotted by number of iterations [Full corpus]

to be more easily made. As can be seen, there are some small differences in the performance of the models – restarting at 2000 seems to give the best performance, with lower restart points each in turn leading to a decrease in performance, whilst a higher value of 3000 is also worse.

For comparison, figure 5.17 shows the effect of running multiple iterations when restarting after lower numbers of moves – data is plotted for restart values of 1, 10, 50 and 100 moves.[27] A similar result is observed, with the performance dropping slightly as the restart value decreases but without any large differences.

5.6.2 3- and 4-grams

Given figure 5.11 it does not come as a surprise that the perplexity of the stand-alone 3-gram and 4-gram class pair models gets progressively higher as subsequent iterations

[27] A restart value of 100 is included in both graphs for ease of comparison.

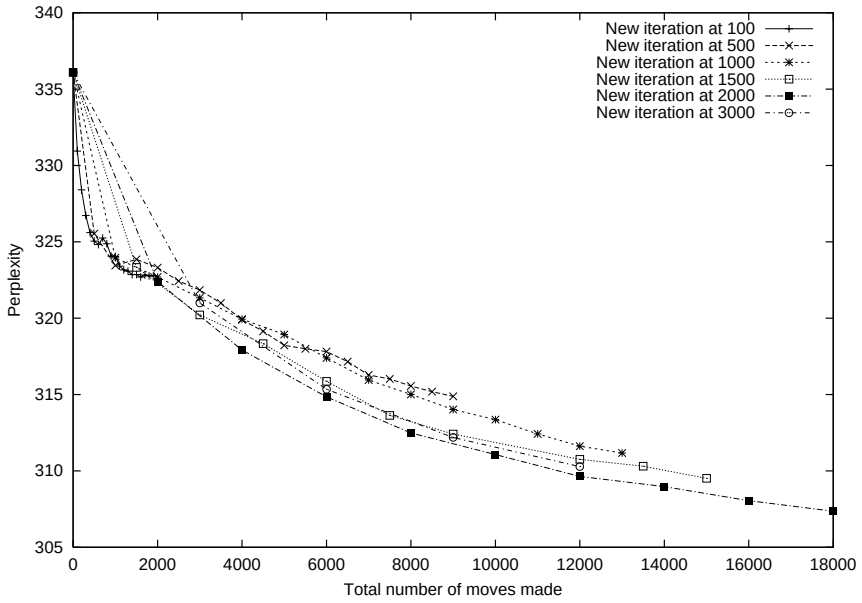


Figure 5.16: Bigram pair-model perplexity on test set when starting a new iteration after a given number of moves – plotted by number of moves [Full corpus]

are undertaken. Figure 5.18 shows the results for the 3-gram case.[28] The order of model performance in the 2-gram case is roughly reversed in the 3-gram and 4-gram case, although since all models get rapidly worse initially this is perhaps not too interesting – after a few thousand moves the perplexity stays fairly constant, however, and does not deteriorate further.

5.6.3 Interpolating with a word model

As previously discussed, one of the most useful assessments of the performance of a language model is its behaviour when interpolated with a full word *n*-gram model – with the 1992 corpus this led to small but worthwhile perplexity gains, but in figure 5.14 it was shown that with the Full corpus this gain is reduced. Figure 5.19 shows that in the trigram case this situation is not significantly improved by running multiple iterations of the pair clusterer algorithm, although it does indicate that as in figure 5.18 the restart-at-500 and restart-at-1000 models remain the best-performing. The

[28] Figure F.6 in appendix F shows the corresponding similarly-shaped graph for the 4-gram case.

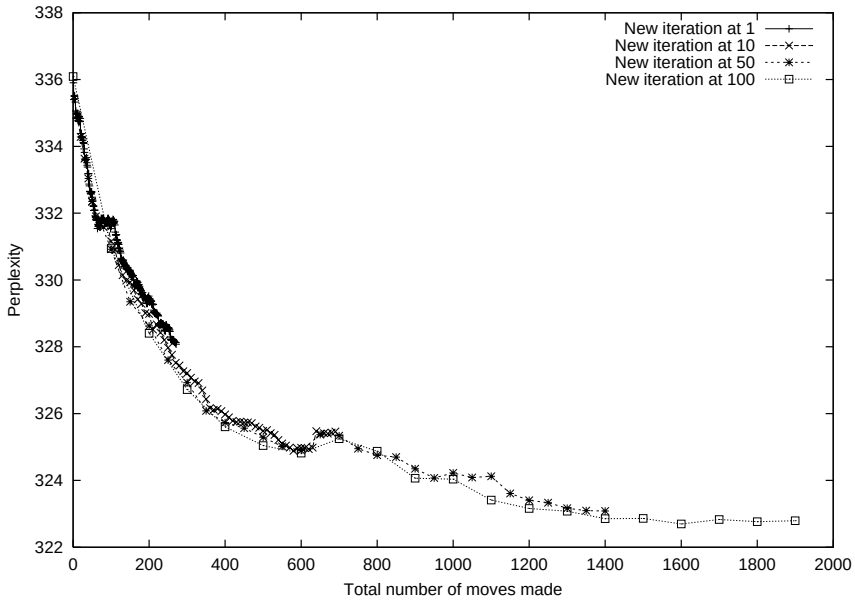


Figure 5.17: Bigram pair-model perplexity on test set when starting a new iteration after a small number of moves – plotted by number of moves [Full corpus]

perplexity values appear to have a general downwards trend across iterations but the changes are too small to draw any meaningful conclusion – it can however be noted that the increases in perplexity without the word model are stabilised to an overall small reduction in perplexity, but performing multiple iterations does not appear to have any effect on the quality of the models. Appendix F presents comparative graphs for smaller restart values – see figure F.7 on page 178 – and for the 4-gram case, where the changes in perplexity are even smaller – see figure F.8.

Figure 5.20 shows how the bigram pair model performance compares when running multiple iterations and interpolating with a bigram word model. The perplexity drops further with each iteration shown.

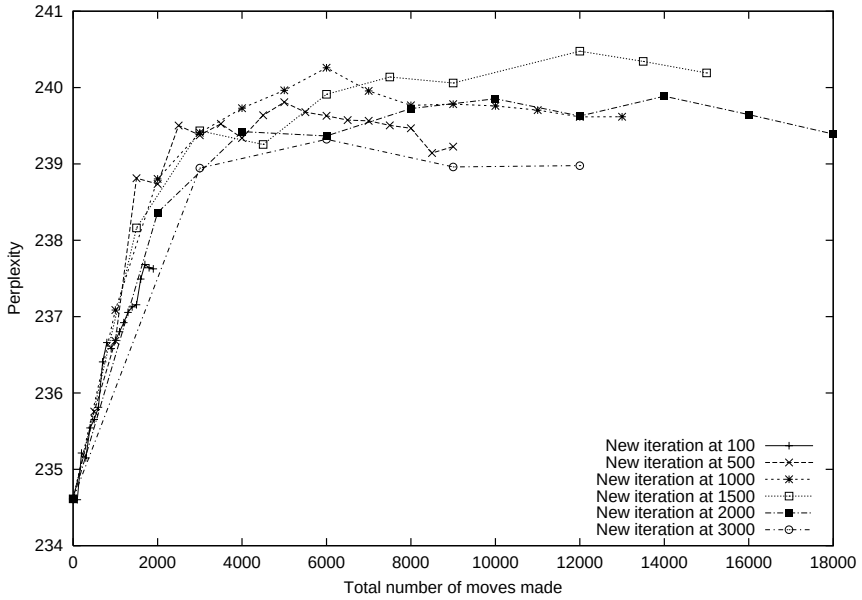


Figure 5.18: Trigram pair-model perplexity on test set when starting a new iteration after a given number of moves – plotted by number of moves [Full corpus]

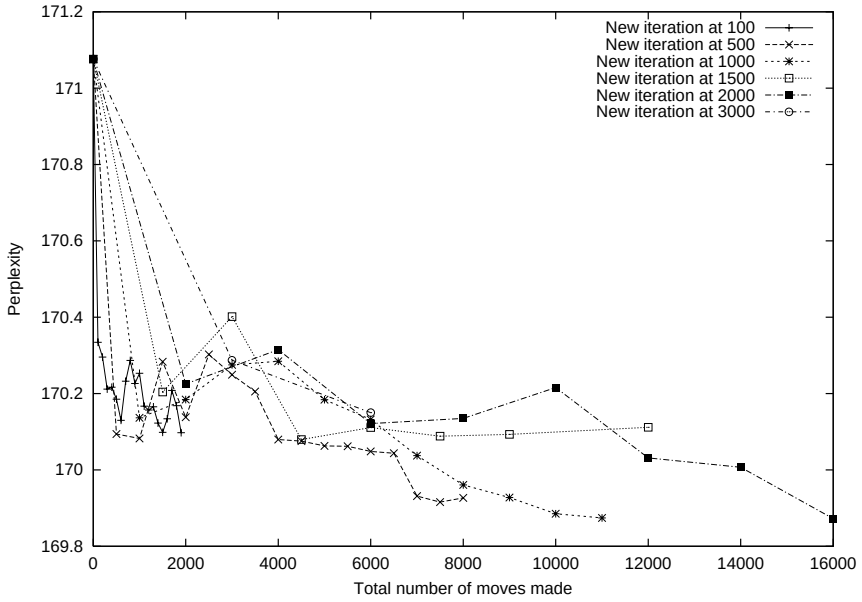


Figure 5.19: Trigram word + pair-model perplexity on test set when starting a new iteration after a given number of moves – plotted by number of moves [Full corpus]

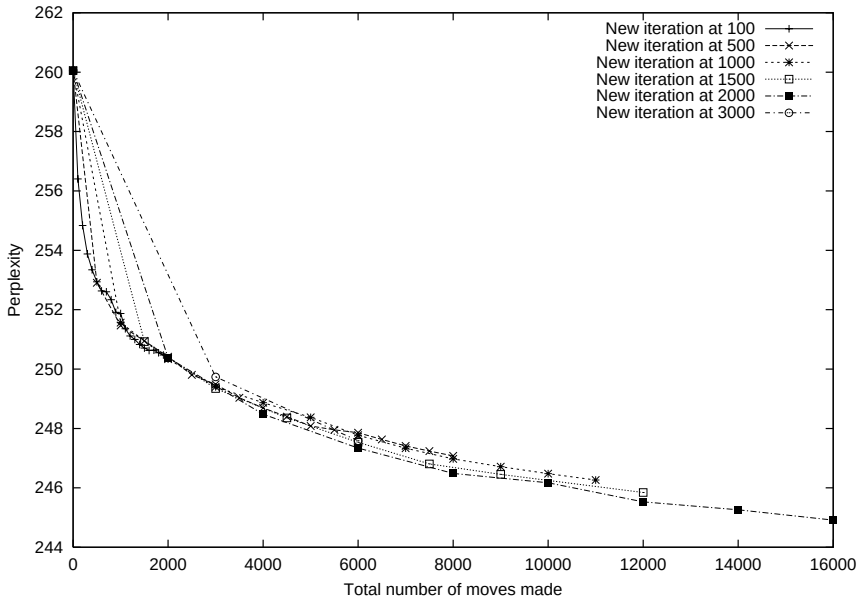


Figure 5.20: Bigram word + pair-model perplexity on test set when starting a new iteration after a given number of moves – plotted by number of moves [Full corpus]

5.6.4 Summary of results

Table 5.3 on the following page summarises the best results on the Full corpus[29] – it shows similar relative changes to table 5.2 on page 126, with the biggest exception being that the standard bigram class model is much more inferior to the bigram word model than it was on the 1992 test, resulting in the gains of the bigram pair model not being large enough to surpass the word model. This is most likely due to the number of classes being more than halved, relative to the number of vocabulary words. Referring to the Full corpus alone, minor gains are observed in the word-interpolated 3- and 4-gram case over the standard class model whilst significant gains are seen in both the standalone and word-interpolated bigram cases.

[29]These perplexity values are slightly lower than those reported in the previous chapter. This is because a different language toolkit utility was used to calculate these – the effect is due to the use of out-of-vocabulary words in the predictions in the previous chapter which were not included by the tool used in the work reported in this chapter, which instead limited the history context to that including in-vocabulary words. Word error rates, however, are directly comparable.

	Word model	Class model		
n-gram size	Weight	Type	Weight	Test set perplexity
4	1.00	-	-	170.87
3	1.00	-	-	182.96
2	1.00	-	-	264.72
4	-	Standard	1.00	216.57
3	-	Standard	1.00	234.61
2	-	Standard	1.00	336.09
4	-	Pair	1.00	216.57
3	-	Pair	1.00	234.61
2	-	Pair	1.00	307.36
4	0.65	Standard	0.35	157.12
3	0.65	Standard	0.35	171.08
2	0.65	Standard	0.35	260.06
4	0.65	Pair	0.35	156.61
3	0.65	Pair	0.35	169.92
2	0.65	Pair	0.35	247.08

Table 5.3: Overview of best perplexity results [Full corpus]

An additional experiment to interpolate the standard and pair class models together along with the word model to see if any additional gains are observed was performed. Figure 5.21 shows the effect on the test set perplexity of interpolating the two class models together with a fixed word model weight of 0.65 – the pair map used was that resulting from the restart-at-500 model's 11th iteration, and bigrams were used for the class models. The horizontal axis shows the weight of the standard class model relative to the pair class model, so at 0.0 the relative weight of the pair model is 1.0 – the overall weight of the two class models is scaled to 0.35 to linearly interpolate with the 0.65 weighted word model. The graph also shows the result of using a lower word model weight of 0.50 in an attempt to allow for the greater mass of the two class models – this, however, led to an increase in perplexity. Overall the figure clearly shows that in the bigram case there is nothing to be gained by interpolating both the pair and standard class models together with the word model, relative to interpolating just the pair and word models.

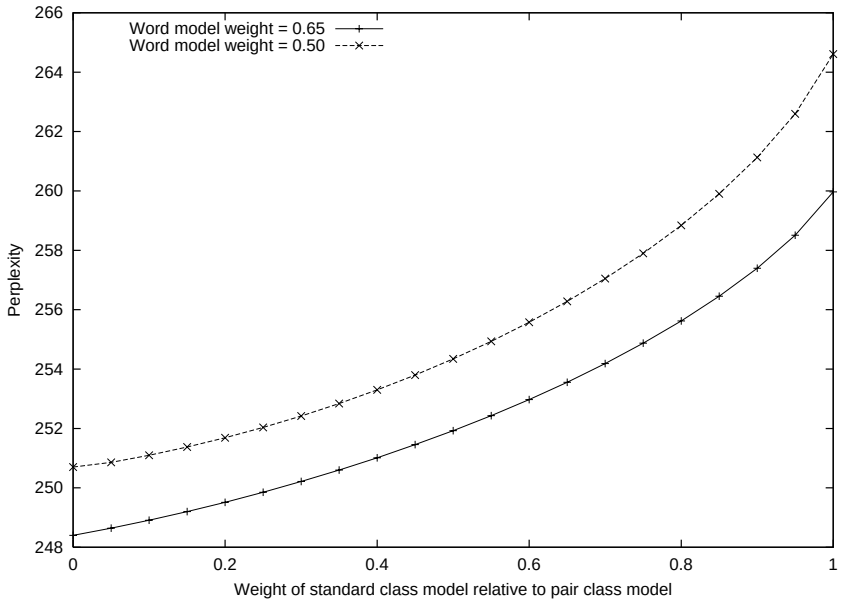


Figure 5.21: Word bigram models interpolated with bigram standard class and bigram pair class models – perplexity by class model weights [Full corpus]

In figure 5.22 the same graph is plotted, but this time using 4-gram class models.[30] The difference in the shape of the curve is striking – in the 4-gram case using half the class model mass for the standard model and half for the pair model gives the overall best result, obtaining a much more worthwhile gain than that of the pair and word model alone. This suggests that the pair model's modelling ability is more divergent from that of the standard class model in the 4-gram case than the 2-gram case. Interpolating with the standard class model smooths some of the estimates which interpolating with the word model wasn't already able to improve.

5.7 Recognition results

The most significant test of a language model intended for speech recognition is, inevitably, to test it in a speech recognition system. To this end, the pair model trained on the Full corpus was assessed by rescoring the lattices originally built for the 1997

[30] A similar result is obtained in the 3-gram case.

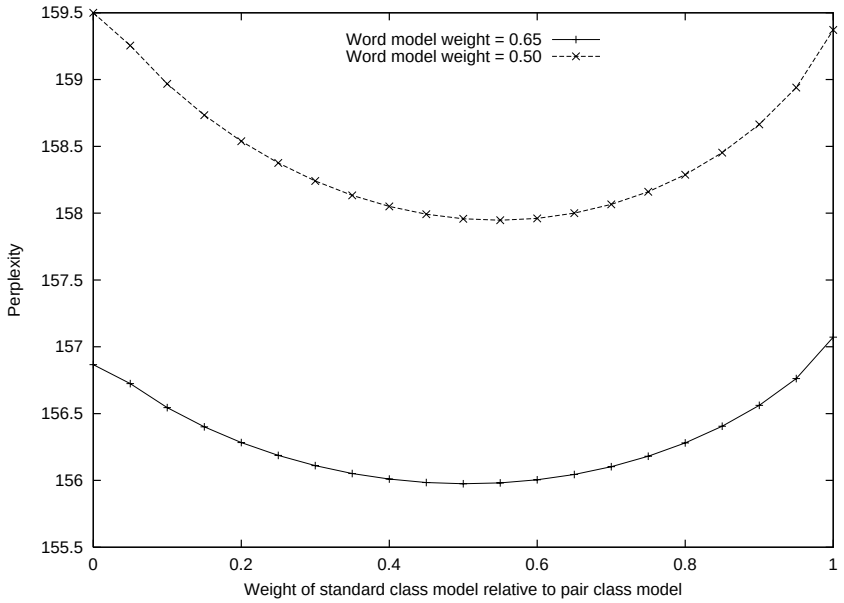


Figure 5.22: Word 4-grams interpolated with 4-gram standard class and 4-gram pair class models – perplexity by class model weights [Full corpus]

HUB-4 CU-HTK evaluation system [Woodland et al 1998], as described in the previous chapter. When interpolating word and class models the same word to class model linear interpolation weights of 0.65 to 0.35 respectively were used as in the previous chapter and in the Full corpus perplexity experiments reported in the previous two sections. Recalling table 4.1 on page 96, the baseline performance of the unmodified word and class models when linearly interpolated together with these weights is a word error rate of 17.1%.[31] Because of the powerful baseline models and the partially mismatched test set, plus the existing observation that the pair model gives its best relative gains when assessed using a smaller training set, large improvements over this result cannot reasonably be expected. Table 5.4 shows the baseline results for various sizes of standard *n*-gram class and word language model.

Because the system tested here is the same as that used for the perplexities reported on the Full corpus in this chapter, corresponding word error rates and perplexities can be compared. Since the best perplexity results were obtained when performing multiple iterations of the pair clustering algorithm, it makes sense to use these same language

[31] This value is calculated using the standard NIST scoring tools.

Model	*n*-gram size	Word error rate (%)
Class	4	18.47
Class	3	19.28
Class	2	22.63
Word	4	17.63
Word	3	18.22
Word	2	21.17
Word + Class	4	17.08
Word + Class	3	17.77
Word + Class	2	21.13

Table 5.4: Baseline word error rate results for standard class and word models

models to rescore the lattices.

5.7.1 Pair model only

Figure 5.23 shows how the word error rate changes as moves are made in the bigram case for a range of iteration restart values. The improvements in bigram test set perplexity reported throughout this chapter are preserved when the model is used in an actual speech recognition task, with progressive decreases in word error rate as further iterations of the pair clusterer are run, although the graph shows the gains beginning to decrease after around 10,000 moves. The overall reduction in word error rate is therefore around 3.8% relative to the standard class model.

The behaviour of the 3-gram and 4-gram models is also found to be consistent with the test set perplexity results reported in the previous section. Figure 5.24 shows the 3-gram pair model behaviour, whilst the similarly-shaped 4-gram result is included in appendix F, figure F.9. In the 3-gram case the word error rate deteriorates rapidly before stabilising within a range of between a 1.0% and 2.3% relative increase in error rate. The equivalent range in the 4-gram case is 2.0% to 3.6%.

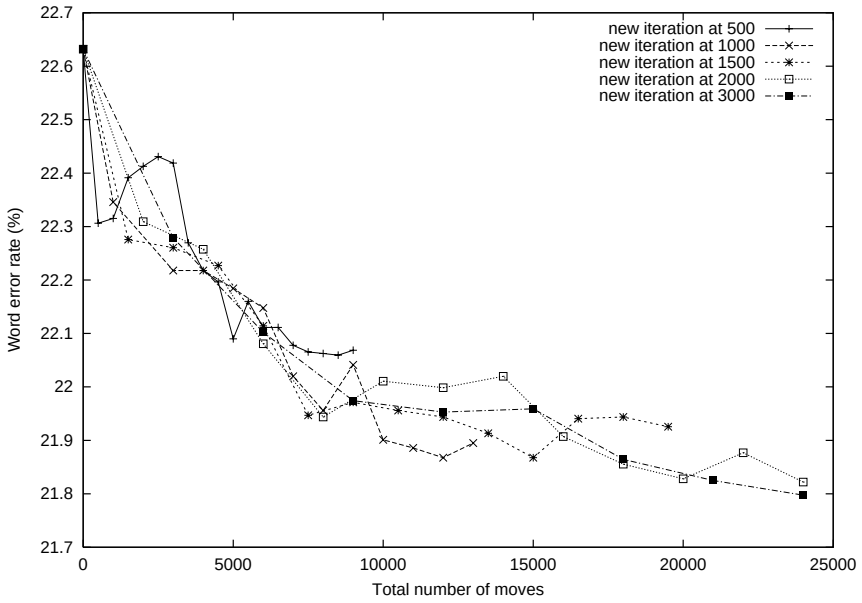


Figure 5.23: Bigram pair class model word error rates shown for different restart points – each point represents the end of an iteration

5.7.2 Pair and word models

The bigram pair class model maintained its performance gain over the standard class model when interpolated with a word model in previous experiments, and figure 5.25 shows that the same is true when calculating word error rate in this lattice rescoring speech recognition system. A relative reduction in word error rate of 3.2% is observed as the system gradually converges as further iterations of the clustering algorithm are run. The same improvement in word error rate is also observed relative to the bigram word model, since the baseline performance of the interpolated bigram word and class models is almost identical to the bigram word model alone – see table 5.4. This is a clear demonstration both that the bigram pair class model is better-performing than the standard model, even when interpolated with a word model, and that it models word sequences which are not already well modelled by the bigram word model.

When interpolating pair 3- and 4-gram models with equivalent word models on the Full corpus, almost no change in perplexity was observed as the pair clustering algorithm was executed, so it is reasonable to expect a similar result when performing lattice rescoring. Figure 5.26 shows that this expectation is fulfilled, and additionally

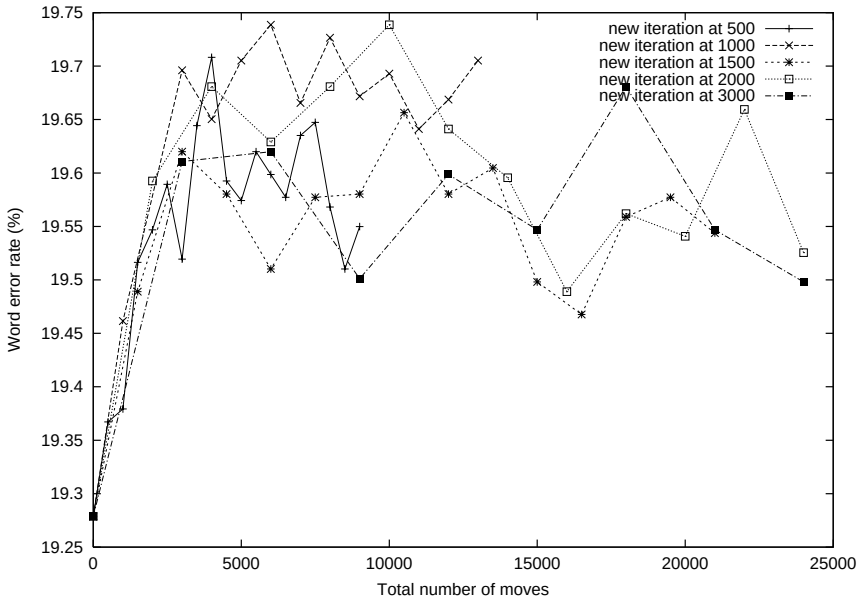


Figure 5.24: Trigram pair class model word error rates shown for different restart points – each point represents the end of an iteration

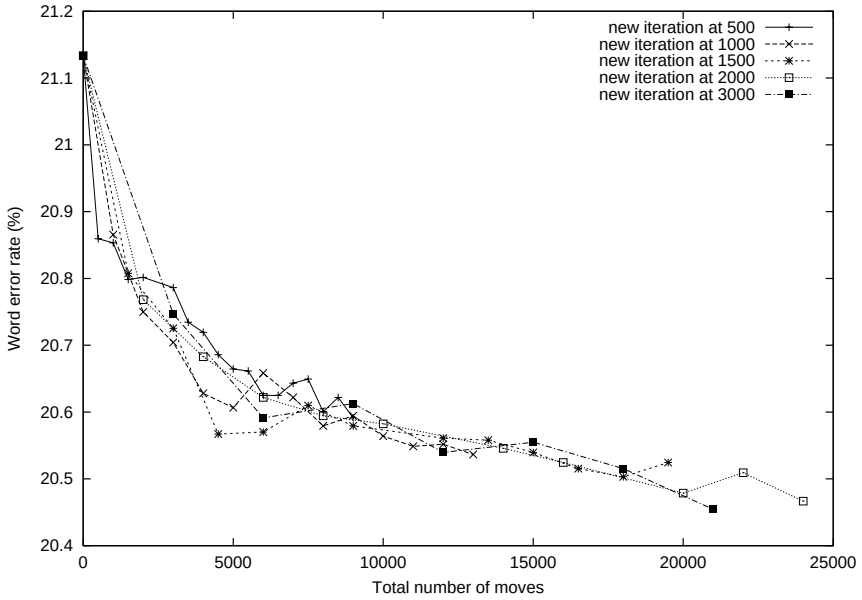


Figure 5.25: Bigram pair class + bigram word model word error rates shown for different restart points – each point represents the end of an iteration

the word error rate initially increases by about 0.9% relatively, before falling back down to its starting point and staying roughly constant – the subsequent jitter in the figure is probably mostly accounted for by noise. The 4-gram result, shown in figure F.10 in appendix F, has a similar shape, with the initial but then recovered relative increase in word error rate equal to about 1.0%.

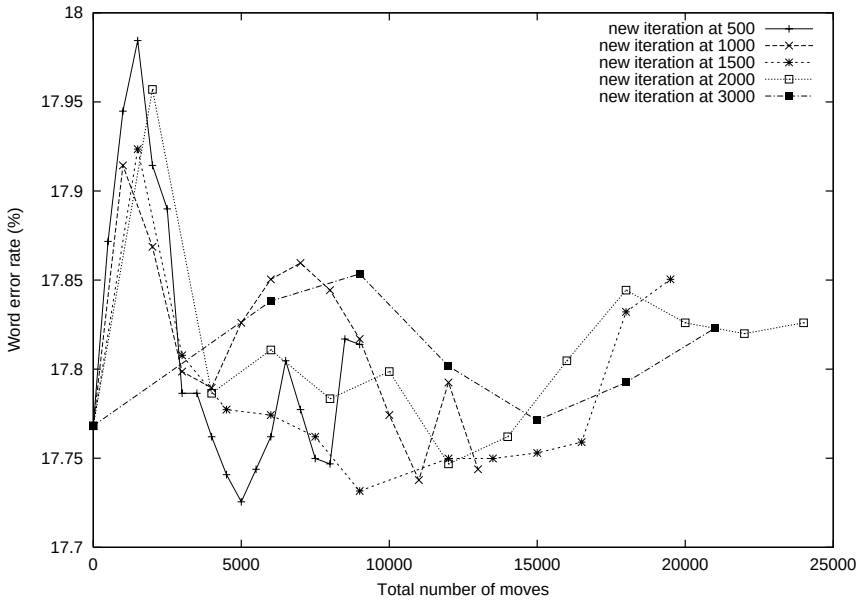


Figure 5.26: Trigram pair class + trigram word model word error rates shown for different restart points – each point represents the end of an iteration

If an oracle was used to select the best observed 4-gram pair model interpolated with a 4-gram word model result then it would obtain a 17.02% word error rate.[32] This difference from the baseline result of 17.08% is not a worthwhile improvement in word error rate, so it must be concluded that on this test set at least that the pair class model can give no improvements on this speech recognition task when used in anything other than 2-gram form. The 2-gram form itself, however, gives a significant reduction in word error rate when compared with other 2-gram models. Table 5.5 summarises the best results observed and incorporates the baseline and perplexity results from earlier in this chapter for ease of comparison.

[32]This result is observed, for example, after 16 iterations of the restart-at-500 algorithm and after 13 iterations in the restart-at-1000 case.

	Word model	Class model			
n-gram size	Weight	Type	Weight	Perplexity	Word error rate (%)
4	1.00	-	-	170.87	17.63
3	1.00	-	-	182.96	18.22
2	1.00	-	-	264.72	21.17
4	-	Standard	1.00	216.57	18.47
3	-	Standard	1.00	234.61	19.28
2	-	Standard	1.00	336.09	22.63
4	-	Pair	1.00	216.57	18.47
3	-	Pair	1.00	234.61	19.28
2	-	Pair	1.00	307.36	21.80
4	0.65	Standard	0.35	157.12	17.08
3	0.65	Standard	0.35	171.08	17.77
2	0.65	Standard	0.35	260.06	21.13
4	0.65	Pair	0.35	156.61	17.02
3	0.65	Pair	0.35	169.92	17.73
2	0.65	Pair	0.35	247.08	20.45

Table 5.5: Overview of best results in terms of both perplexity and word error rate

5.8 Conclusions

A word-pair class-based *n*-gram language model has been developed and its behaviour in terms of perplexity and word error rate examined in detail. Experiments have shown that the model, when trained using a bigram algorithm and used in a bigram form, can obtain substantial reductions both in perplexity and in word error rate, surpassing the performance of a standard bigram class language model significantly. This increase in accuracy is maintained even when models are interpolated with a word *n*-gram language model. The stand-alone pair model also has significantly better performance than the word model whilst having substantially less parameters. It could be argued that the bigram pair model actually considers a triplet of words and so should be compared with a trigram standard model, but it is clearly the case that the pair model only explicitly models word pairs and not word triplets and so the comparison with the bigram standard class model is in fact fair. As further evidence, the standard trigram class model adds an additional 4,307,353 trigram parameters over and above those of

the bigram model, which is many orders larger than the few thousand pair exceptions necessary to obtain a significant improvement in the bigram pair model.

Performance with 3-gram and 4-gram models is disappointing on the Full corpus test set, but the correlation across all results between the behaviour when assessed with perplexity and with word error rates suggests that word error rate reductions could have been achieved if recognition experiments had been performed on audio well-matched by the 1992 corpus. The pair model is likely to be particularly sensitive to domain-matching, so the relatively smaller 1992 corpus facilitated this by reducing the vocabulary size and range of coverage of the text.

In the 1992 case in particular, the bigram pair class model demonstrated how improvements in performance can be made relative to a bigram word model whilst actually substantially reducing the memory footprint – these gains are likely to be even greater with smaller vocabularies and more restricted domains, presenting a promising technique for highly compact language models suitable for use in mobile phone or PDA predictive or corrective text entry systems.

5.8.1 Future work

Having proposed and then examined the performance of the pair class model it seems likely that extending the pair clustering algorithm to be based around 3- or even 4-grams would provide a pair class map which generalises much better to higher order *n*-grams, although issues of feasibility with respect to its computer-based implementation would need to be resolved first.[33] Considering the observation of small gains with the 1992 test even in these higher order cases, these gains could be substantial. Alternatively a method of culling poorly performing pairs from higher-order *n*-gram models could be investigated, although work in this area to date has proved unsuccessful.

[33]This would also possibly be at the expense of poorer performance with lower-order *n*-grams.

6. Conclusions

Two methods of adapting class *n*-gram language models to react to the current text have been described and assessed.

6.1 Review of conducted work

Throughout this thesis all experiments have been performed using the most testing systems feasible. Word 4-gram models combined with 4-gram class models, each trained on extremely large corpora, were used as baselines to improve upon, and not just perplexity but also recognition experiments were performed.

6.1.1 Topic construction

After examining existing topic construction methods, topic sets of varying sizes were formed using an agglomerative inverse document frequency method with broadcast and newswire news stories as the basic building blocks. Examination of the quality of these topics was encouraging for all topic sets, with above-average correlation of tuples of 4 words within topics and significant term frequency distribution differences between topics.

An attempt to recluster and improve the topics via a similarity metric directly equivalent to the target 4-gram language models was less successful, confirming the relative success of the agglomerative inverse document frequency method.

Assessment of *n*-gram models built using the topic sets revealed severe problems of data sparsity for 3-gram and 4-gram models, motivating the class model topic adaptation proposed in the following chapter.

6.1.2 Topic class model adaptation

A new form of topic model adaptation, implemented via modification of the word-given-class component of class *n*-gram models, was proposed and its performance assessed. This method not only allows compact storage of large numbers of topics but also allows reliable estimates of topic parameters to be made from the relatively small quantities of text available in topic corpora. Significant reductions in word error rate were obtained over the baseline system.

In one set of experiments a model with 500 topics was constructed, using just 62MB of memory to store all the topic parameters without any compression applied. This compares extremely favourably with other topic adaptation methods such as storing 500 word 4-gram models, which would require of the order of 2GB. Even for this fine partitioning of the corpora, analysis showed that each of the 500 topic components remained well-trained.

An attempt at topic reclustering which this time directly targetted the topic class models again failed to lead to any worthwhile improvement. Additional comparisons with other forms of topic adaptation confirmed the supremacy of the new approach.

6.1.3 Pair map class model

Based on the observation that it must be the case that the classification function in a standard *n*-gram class model is sub-optimal in some contexts, a new model was proposed which allows each word to change class based on the most-recently seen word of context. Having defined this model, a novel clustering scheme was described in order to train this model. Experiments assessing the performance of this new model relative to a standard class model revealed that although it is difficult to improve on the baseline system, significant improvements in a bigram test system were obtained with only a minimal increase in the number of parameters. Small gains with longer-length *n*-grams were obtained on a more constrained test, leading to the conclusion that further gains might be obtained in more restricted domains.

It was also found that a bigram pair model with just 1,003 classes could substantially outperform a bigram word model having significantly more parameters, rendering the model highly appropriate for use in restricted space applications, especially given the expected further gains on more focussed domains.

6.1.4 Combined topic and pair adaptation

Both new models allow adaptation using a minimal number of additional parameters, unlike other proposed adaptation methods. Combining the two could lead to further gains. Initial testing on the full recognition system used throughout this thesis led to no improvement in the 3-gram and 4-gram cases, but on a 2-gram test set an improvement was obtained. The standard 1,003 class model had a 2-gram baseline word error rate of 22.6%. The 50 topic class model reduces the 22.6% to 22.2%,[1] whilst the pair class model reduces it to 21.8%. Combining the two, however, gives a large word error rate reduction to 20.3%, which is also significantly lower than the bigram word model baseline of 21.2%. The model combination was implemented by adapting the word counts used in the pair model on the basis of each topic. Only the single best topic was used for the pair model experiment – as was noted in chapter four, mixtures of topics lead to better performance so this result could perhaps be further improved. This result represents a significant reduction in word error rate and holds a great deal of potential for future work.

6.2 Further work

The aspects of topic clustering reported in this thesis were investigated thoroughly, but it is possible that by using an additional article rejection metric or threshold that the quality of the topics could be increased further. Additionally, *k*-means clustering based on the inverse document frequency scores might achieve further refinement.

The topic-adaptive class model outperformed all model variants tested, but it is of course possible that it could be improved further. Although the set-to-one behaviour for unseen words in each topic was the best that was achieved and is inherently smoothed by interpolation with other topics, it is clearly rather crude and so further research into this might prove beneficial.

The topic model could be applied to different domains – during development good results were also obtained on different era broadcast news text, but it could also be tested on more restricted domains and with even less training text per topic. In addi-

[1] This result was not listed previously. 22.2% is the result when using the single-best topic; 22.1% is obtained when using a mixture of topics.

tion, alternative methods of setting the topic weights based on a rolling analysis of all previously seen text could be developed.

The pair-based class model perhaps has the greatest number of opportunities for improvement. An updated training algorithm based on clustering using 3-gram or 4-gram scores might reasonably be expected to present similar improvements relative to the 3- or 4-gram baselines as the current 2-gram algorithm provides over the 2-gram baseline. The pair model could also be applied to smaller domain problems where the difficulty of training the pair map is reduced, or some other method of constructing the pair map could be proposed. It might also be beneficial to allow groups of pairs to be moved all at once in order to increase the confidence attached to each move. A context-dependent class map which used more than just the single most recent word might also lead to improvements.

Finally, the combination of the topic and pair adaptation methods holds an array of possibilities for further research, as outlined in the previous section.

6.3 Summary

Two new methods of adapting the performance of a class based language model on the basis of recently seen text have been proposed and assessed in this thesis, and it has been shown that it *is* possible to perform unsupervised adaptation on a very general corpus and obtain not only a perplexity but also a recognition gain. The usefulness of the described topic adaptation method in state of the art systems has been clearly demonstrated, and the pair class map adaptation scheme shows much promise as well as direct application for use in systems where only small models will fit such as mobile phone text prediction systems. Hopefully this work will form the basis of future research to develop these ideas and their applications.

A. 182 word stoplist

This appendix contains the full stop list of words which were ignored when performing the inverse document frequency article clustering described in chapter 3.

'CAUSE	'EM	A
ABOUT	ACTUALLY	AFTER
AGAIN	AIN'T	ALL
ALSO	AM	AN
AND	ANOTHER	ANY
ANYTHING	ARE	AREN'T
AROUND	AS	AT
BE	BECAUSE	BECOME
BEEN	BEFORE	BEING
BETWEEN	BUT	BY
CAME	CAN	CAN'T
COME	COULD	COULDN'T
DID	DIDN'T	DO
DOES	DOESN'T	DON'T
DONE	DOWN	ELSE
EVEN	EVER	EVERY
FOR	FROM	GET
GO	GOING	GOOD
GOT	HAD	HADN'T
HAS	HASN'T	HAVE
HAVEN'T	HE	HE'LL
HE'S	HELLO	HER
HERE	HERE'S	HI

HIM	HIS	HOW
I	I'D	I'LL
I'M	I'VE	IF
IN	INTO	IS
ISN'T	IT	IT'LL
IT'S	ITS	JUST
KNOW	LAST	LET
LIKE	LONG	LOT
MADE	MAKE	MAN
MANY	MAY	ME
MEAN	MIGHT	MORE
MOST	MUCH	MY
NEED	NEW	NEXT
NO	NOT	NOW
OF	OH	ON
ONE	ONLY	OR
OTHER	OUR	OUT
OVER	OWN	PROBABLY
PUT	QUOTE	REALLY
RIGHT	SAID	SAME
SAY	SAYING	SAYS
SEE	SHE	SHE'LL
SHE'S	SHOULD	SHOULDN'T
SO	SOME	SOMETHING
STILL	SUCH	SURE
TAKE	TELL	THAN
THAT	THAT'S	THE
THEIR	THEM	THEN
THERE	THERE'S	THESE
THEY	THEY'D	THEY'LL
THEY'RE	THEY'VE	THING
THINGS	THIS	THOSE
THOUGH	THROUGH	TO

TOO	UP	US
VERY	WANT	WAS
WASN'T	WE	WE'D
WE'LL	WE'RE	WE'VE
WELL	WENT	WERE
WEREN'T	WHAT	WHAT'S
WHEN	WHERE	WHETHER
WHICH	WHILE	WHO
WHO'VE	WHY	WILL
WITH	WON'T	WOULD
WOULDN'T	YEAH	YES
YET	YOU	YOU'D
YOU'LL	YOU'RE	YOU'VE
YOUR		

B. Topic details

This appendix gives breakdowns of the different topics used in chapters 3 and 4, comparing the size of each topic corpus. It also contains example contents for selected topics. The number of words given is the number of tokens in the training text which includes sentence start and sentence end tokens. All data is for the agglomeratively-clustered topics unless otherwise stated.

B.1 4 topics

Topic	Articles	Sentences	Words
1	47539	2694345	50021094
2	32102	2364573	41948083
3	38356	2383999	44064114
4	35784	1964880	33188428

B.2 8 topics

Topic	Articles	Sentences	Words
1	26057	1437026	26895019
2	17968	1769894	29178255
3	20598	1624704	28858588
4	17758	759295	15205526
5	21482	1257319	23126075
6	18829	958014	16772900
7	16955	1006866	16415528
8	14134	594679	12769828

B.3 50 topics

Topic	Articles	Sentences	Words
1	6425	303555	6129286
2	3831	194597	3591473
3	4373	397936	6827656
4	4884	216639	4103770
⋮	⋮	⋮	⋮
48	2399	134807	2539245
49	2215	133872	2501971
50	2661	105910	2091509

B.3.1 Histograms

See the text in chapter 3 for a discussion of the histograms included in this section.

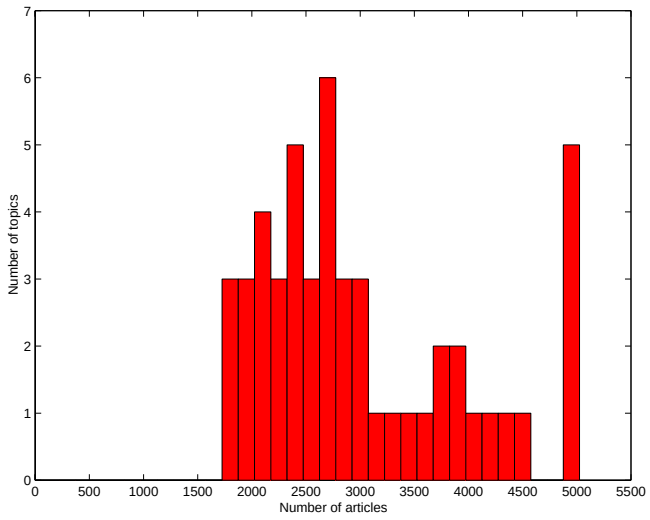


Figure B.1: 50 topics – number of topics with a given number of articles (bin width of 150 articles)

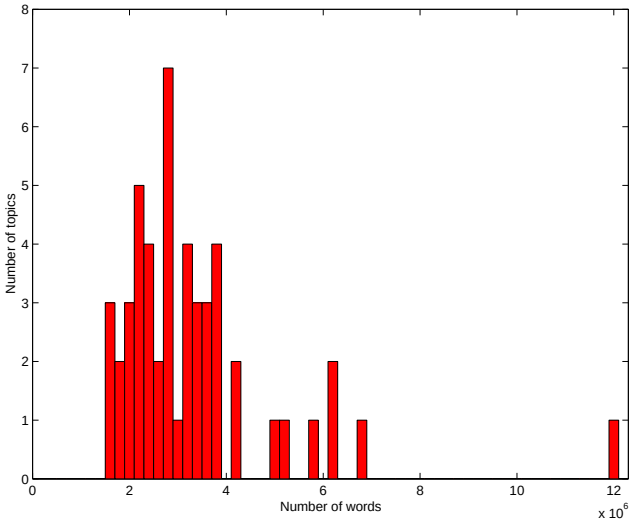


Figure B.2: 50 topics – number of topics with a given number of words (bin width of 200,000 articles)

B.4 100 topics

This table lists the details for only the first and last few topics as well as the two largest topics. The histograms in the following section provide a more complete breakdown.

Topic	Articles	Sentences	Words
1	2450	123496	2407366
2	1762	116822	2015873
3	1295	112198	1948623
⋮	⋮	⋮	⋮
6	3064	275461	4909721
⋮	⋮	⋮	⋮
85	4996	774689	11833029
⋮	⋮	⋮	⋮
98	1544	49632	1253160
99	877	53796	849284
100	918	48524	874276

B.4.1 Histograms

Please refer to the text in chapter 3 for a discussion of the histograms included in this section.

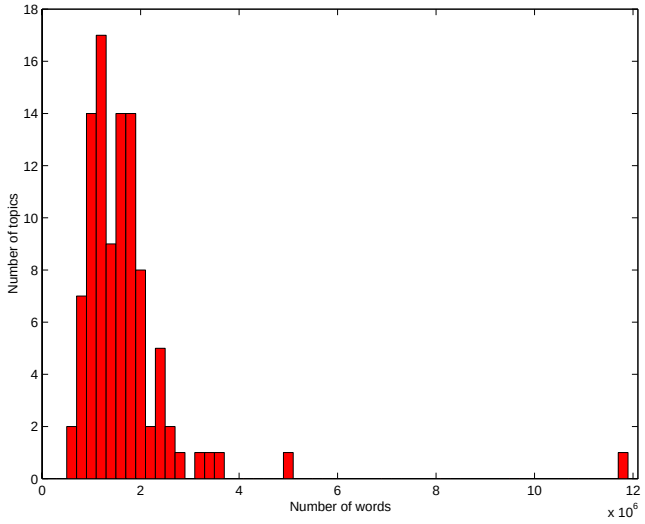


Figure B.3: 100 topics – number of topics with a given number of words (bin width of 200,000 words)

(Please turn to the following page for further histograms)

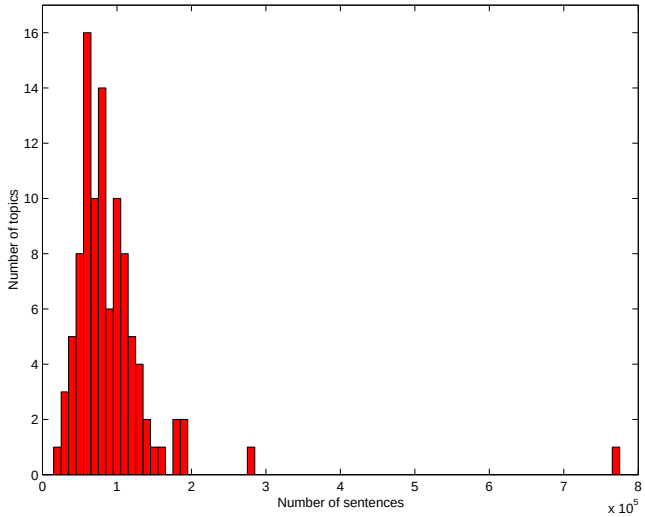


Figure B.4: 100 topics – number of topics with a given number of sentences (bin width of 10,000 sentences)

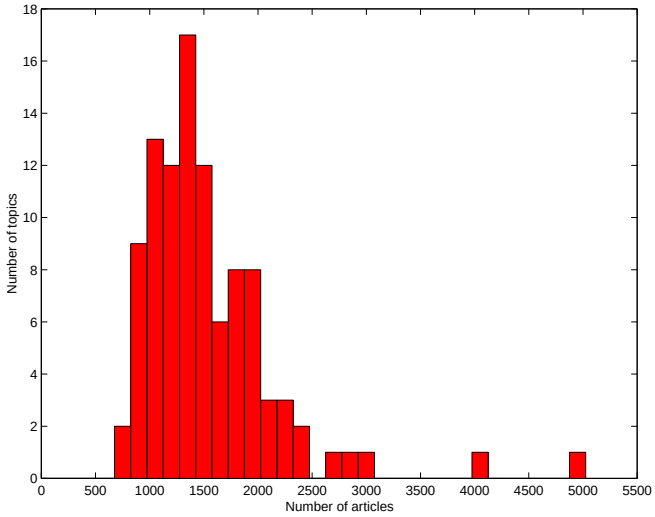


Figure B.5: 100 topics – number of topics with a given number of articles (bin width of 150 articles)

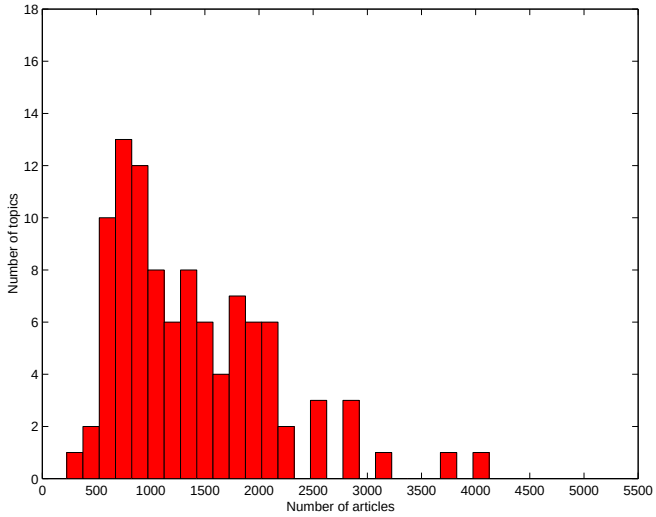


Figure B.6: Reclustered 100 topics – number of topics with a given number of articles (bin width of 150 articles)

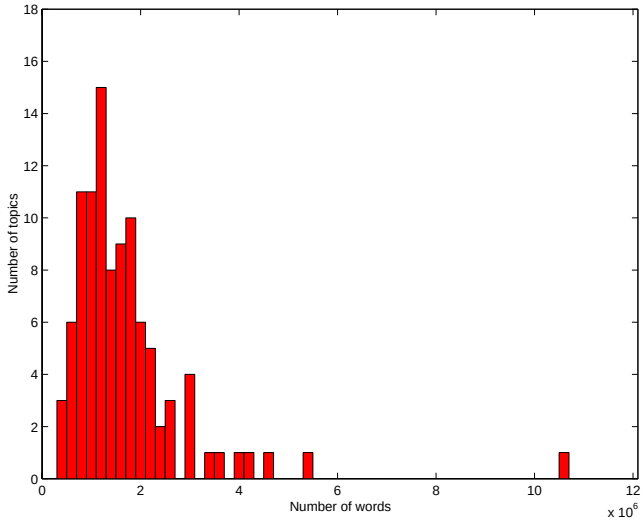


Figure B.7: Reclustered 100 topics – number of topics with a given number of words (bin width of 200,000 words)

B.4.2 Most frequent word examples

This section contains lists of the most frequent words in each of three randomly selected topics from the 100 topic corpus. Note that the stoplist has been applied for clarity, although single letter initials have been included.

Topic 1

Word	Count	Word	Count	Word	Count
U.	10589	N.	9942	BOSNIA	8260
THINK	7583	PEOPLE	7317	BOSNIAN	6361
SERBS	6239	WAR	5946	PEACE	5347
UNITED	5319	PRESIDENT	4958	S.	4699
SARAJEVO	4086	TWO	3972	SERB	3628
TIME	3502	MILITARY	3355	C.	3327
TODAY	3220	BACK	3168	WORLD	3056
TROOPS	2984	STATES	2949	NATO	2905
NATIONS	2835	AIR	2700	FIRST	2606
GOVERNMENT	2557	PLAN	2470	THREE	2441
CLINTON	2341	WAY	2315	FORCES	2301
FORCE	2252	INTERNATIONAL	2205	AMERICAN	2097
AGAINST	2097	AGREEMENT	2089	MUSLIMS	2062
SERBIAN	2037	THOUSAND	2035	STATE	1898
MUSLIM	1859	HUNDRED	1830	COUNTRY	1827
YEARS	1808	PART	1786	GENERAL	1741
UNDER	1706	WEEK	1673	GROUND	1652
SECRETARY	1636	FIGHTING	1626	SITUATION	1621
POINT	1595	YEAR	1562	CITY	1560
YUGOSLAVIA	1559	MR.	1557	DAY	1519
END	1474	FIVE	1450	FACT	1427
QUESTION	1423	SECURITY	1404	SERBIA	1397

Topic 2

Word	Count	Word	Count	Word	Count
PEOPLE	6879	THINK	6214	DOCTOR	4264
N.	3720	TWO	3470	CANCER	3426
WOMEN	3283	HEALTH	3273	YEARS	3217
TIME	3183	C.	2801	MEDICAL	2652
WAY	2466	PATIENTS	2462	DISEASE	2418
BACK	2322	CARE	2311	LIFE	2294

Topic 3

Word	Count	Word	Count	Word	Count
THINK	10692	PEOPLE	7382	CLINTON	6785
PRESIDENT	6187	TWO	4293	BUSH	4233
BILL	4032	CAMPAIGN	3756	PEROT	3745
TIME	3280	REPUBLICAN	3043	BACK	3010
PARTY	2903	N.	2708	COUNTRY	2520
WAY	2467	PERCENT	2353	POLITICAL	2232
DEMOCRATIC	2224	DEMOCRATS	2175	YEARS	2135
HOUSE	2129	STATE	2100	VOTE	2089

B.4.3 Example topic articles

This section contains the opening and concluding sentences from four randomly selected articles within the second-largest of the 100 topic set, topic ‘6’.

Note that the <s> and </s> tokens represent the special start and end of sentence words respectively.

Example 1

<s> GOOD MORNING </s>
<s> I'M RIKKI KLIEMAN </s>
<s> DID O. J. SIMPSON TOSS A DUFFEL BAG INTO A TRASH CAN AT LOS ANGELES INTERNATIONAL AIRPORT JUST HOURS AFTER HE ALLEGEDLY KILLED HIS EX WIFE AND HER FRIEND </s>
<s> IF YOU'RE PROSECUTOR MARCIA CLARK THEN THE ANSWER IS YES </s>
<s> IF YOU'RE THE DEFENSE THE ANSWER IS NO </s>
⋮
<s> HE'S GOING TO ORDER ALSO IN WRITING ON MONDAY THE DECISION ON WHETHER OR NOT THERE HAS IN FACT BEEN A WAIVER AND WHETHER YOU CAN GO INTO KARY MULLIS' CHARACTER AND LIFE STYLE </s>
<s> SO WE'LL WAIT AND SEE ON MONDAY </s>
<s> THANKS SO MUCH GREG </s>
<s> WHAT WE'RE GOING TO DO RIGHT NOW IS TAKE A COMMERCIAL BREAK </s>
<s> WHEN WE COME BACK WE'LL TAKE A LOOK AHEAD </s>

Example 2

<s> WELCOME BACK TO C. N. N.'S WORLD NEWS I'M JOE OLIVER IN ATLANTA </s>
<s> AND I'M JEANNE MESERVE IN WASHINGTON WITH THIS SPECIAL LOOK AT THE WEEK'S DEVELOPMENTS IN THE O. J. SIMPSON CASE </s>
<s> THE PROSECUTION FOUGHT FOR DAMAGE CONTROL THIS WEEK AND THE DEFENSE SUFFERED A SETBACK OF ITS OWN </s>
⋮
<s> THEY ANNOUNCED THEY'RE DROPPING SEVERAL DOMESTIC VIOLENCE WITNESSES AND COULD CONCLUDE THEIR SIDE OF THE CASE BY THE END OF THE COMING WEEK </s>
<s> JIM HILL FOR C. N. N. LOS ANGELES </s>

Example 3

<s> I THINK TO SPEED THINGS UP THE WHOLE THING WAS KIND OF LIKE A PRIORITY MATTER </s>
<s> WERE YOU TOLD BY ANYONE IN THE LABORATORY ON JUNE FOURTEENTH WITHDRAWN </s>
<s> DID GREG MATHESON TELL YOU ON THE MORNING OF JUNE FOURTEENTH MR. YAMAUCHI DON'T BE IN A RUSH HERE JUST BE CAREFUL </s>
⋮
<s> YOU CAN SEE IT'S ALSO REPEATED AT ELEVEN THIRTY P. M. </s>
<s> EASTERN TIME AND BECAUSE TO THE COURT HOLIDAY ON MONDAY C. N. N.'S LIVE COVERAGE OF THE TRIAL RESUMES TUESDAY NOON EASTERN NINE A. M. </s>
<s> PACIFIC </s>
<s> I'M JIM MORET </s>

Example 4

<s> NICOLE BROWN IN THE END WOULD NOT BE EXCLUDED AS THE SOURCE OF THOSE ITEMS </s>
<s> AND NOW HOW MANY GENETIC MARKERS IF YOU COMBINE THE CELLMARK TESTING ON THE THREE ITEMS FROM THE NAIL CLIPPINGS AND SCRAPINGS WITH THE D. ONE S. EIGHTY MARKER HOW MANY GENETIC MARKERS WERE TESTED IN ORDER TO TRY TO DEMONSTRATE THAT NICOLE BROWN COULD NOT BE THE SOURCE OF WHAT WAS UNDER HER FINGERNAILS OR ON HER FINGERNAILS </s>
<s> I WOULD OBJECT TO THE WORD COMBINED </s>
<s> SUSTAINED </s>
⋮
<s> THANK YOU NATALIE AND LET'S GO INTO THE COURTROOM </s>
<s> THE JURORS HAVE VIEWED THE FIRST WAVE OF AUTORADS WITH RESPECT TO ITEM THIRTEEN THAT IS THE SOCK FOUND IN O. J. SIMPSON'S MASTER BEDROOM </s>
<s> LET'S LISTEN IN NOW TO THE PROCEEDINGS </s>

B.4.4 Full article example

This is the complete text of one of the shortest articles in the largest of the 100 topics – topic '86'.

<s> THE PROSECUTION IN THE O. J. SIMPSON CASE HAS WON ITS CHALLENGE TO JUDGE LANCE ITO'S PROPOSED JURY INSTRUCTIONS REGARDING THE UNAVAILABILITY OF FORMER L. A. P. D. DETECTIVE MARK FUHRMAN AS A WITNESS </s>
<s> THE COURT OF APPEAL HAS ORDERED JUDGE LANCE ITO EITHER TO VACATE HIS ORDER REGARDING THESE PROPOSED JURY INSTRUCTIONS GIVEN THE FACT THAT HE SUSTAINED THE CLAIM OF A PRIVILEGE ASSERTED BY THE FORMER DETECTIVE OR IN THE ALTERNATIVE THE ATTORNEYS FOR BOTH SIDES WILL APPEAR IN THE COURT OF APPEAL ON MONDAY MORNING TO ARGUE THIS MATTER </s>
<s> EACH SIDE WILL HAVE ONLY ONE COUNSEL PRESENT AND THE ARGUMENT WILL BE LIMITED TO TEN MINUTES PER SIDE </s>
<s> ONCE AGAIN THE PROSECUTION IN THE SIMPSON CASE HAS APPARENTLY PREVAILED HERE </s>
<s> THE COURT OF APPEAL HAS PUT A HALT TO THE PROPOSED JURY INSTRUCTION REGARDING THE UNAVAILABILITY OF DETECTIVE MARK FUHRMAN AS A WITNESS IN THIS CASE </s>
<s> C. N. N.'S MARK WATTS SPOKE WITH DEFENSE ATTORNEY JOHNNIE COCHRAN THIS AFTERNOON </s>
<s> JOHNNIE COCHRAN HAS INDICATED HE IS NOT HAPPY WITH THE RULING BUT HE WILL ARGUE IT ON MONDAY </s>
<s> THE DEFENSE WILL NOT REST UNTIL THE MATTER IS SETTLED </s>

C. Additional topic assessment graphs

This appendix includes additional graphs which amplify the discussion on topic purity in chapter 3. See section 3.6.3 for full details.

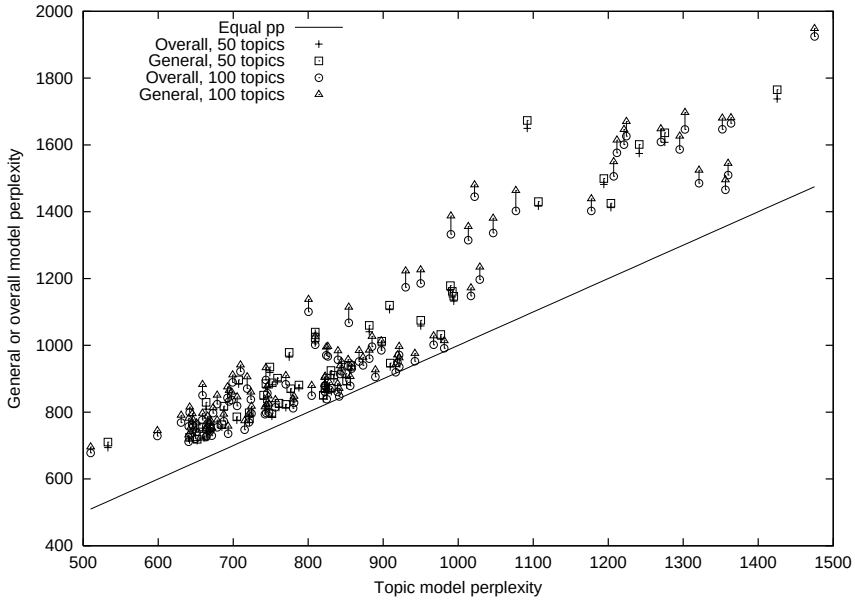


Figure C.1: 50 and 100 agglomerative topic sets – unigram perplexities of topic test sets plotting topic model perplexity against both overall and general models. Lines join data points corresponding to the same topics.

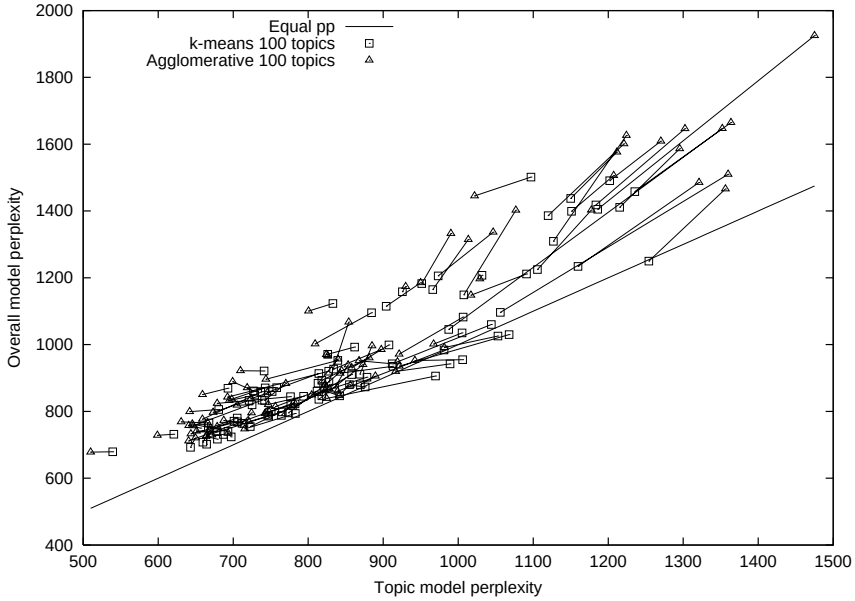


Figure C.2: 100 topic sets – unigram perplexities of appropriate test sets plotting topic model perplexity against both agglomeratively and *k*-means clustered topic models. Lines join data points corresponding to the same topics.

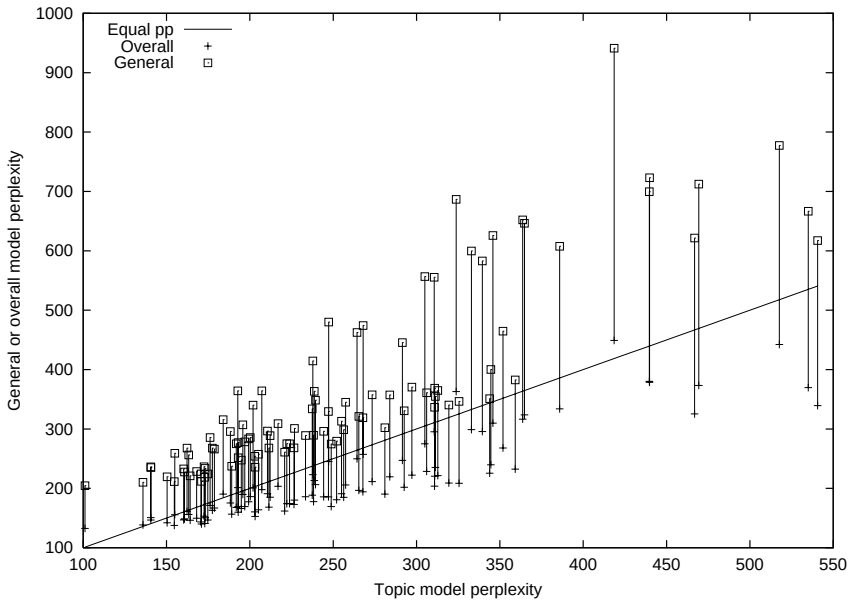


Figure C.3: Agglomerative 100 topic set – bigram perplexities of topic model against both overall and general models. Lines join data points corresponding to the same topics.

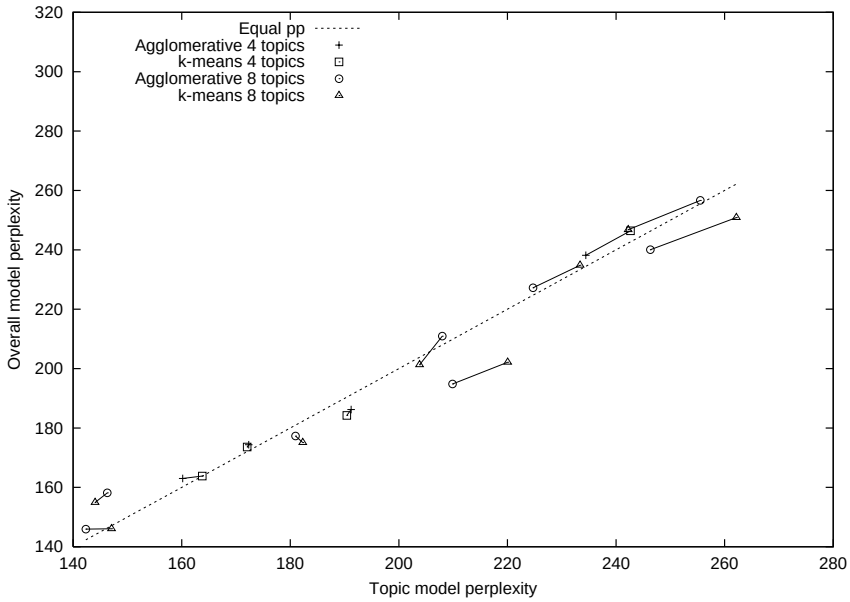


Figure C.4: Reclustered 4 and 8 topic sets – bigram perplexities of both agglomerative and *k*-means clustered topic models against overall model.

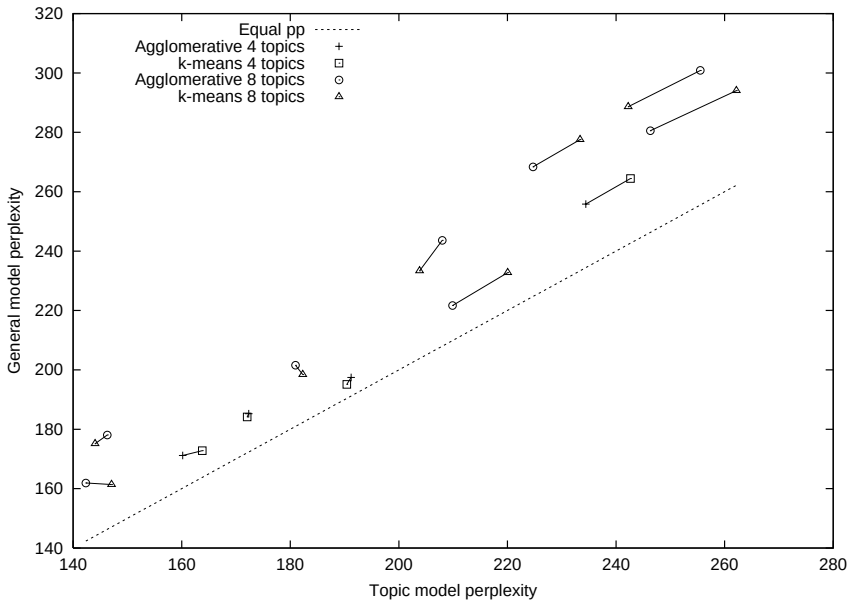
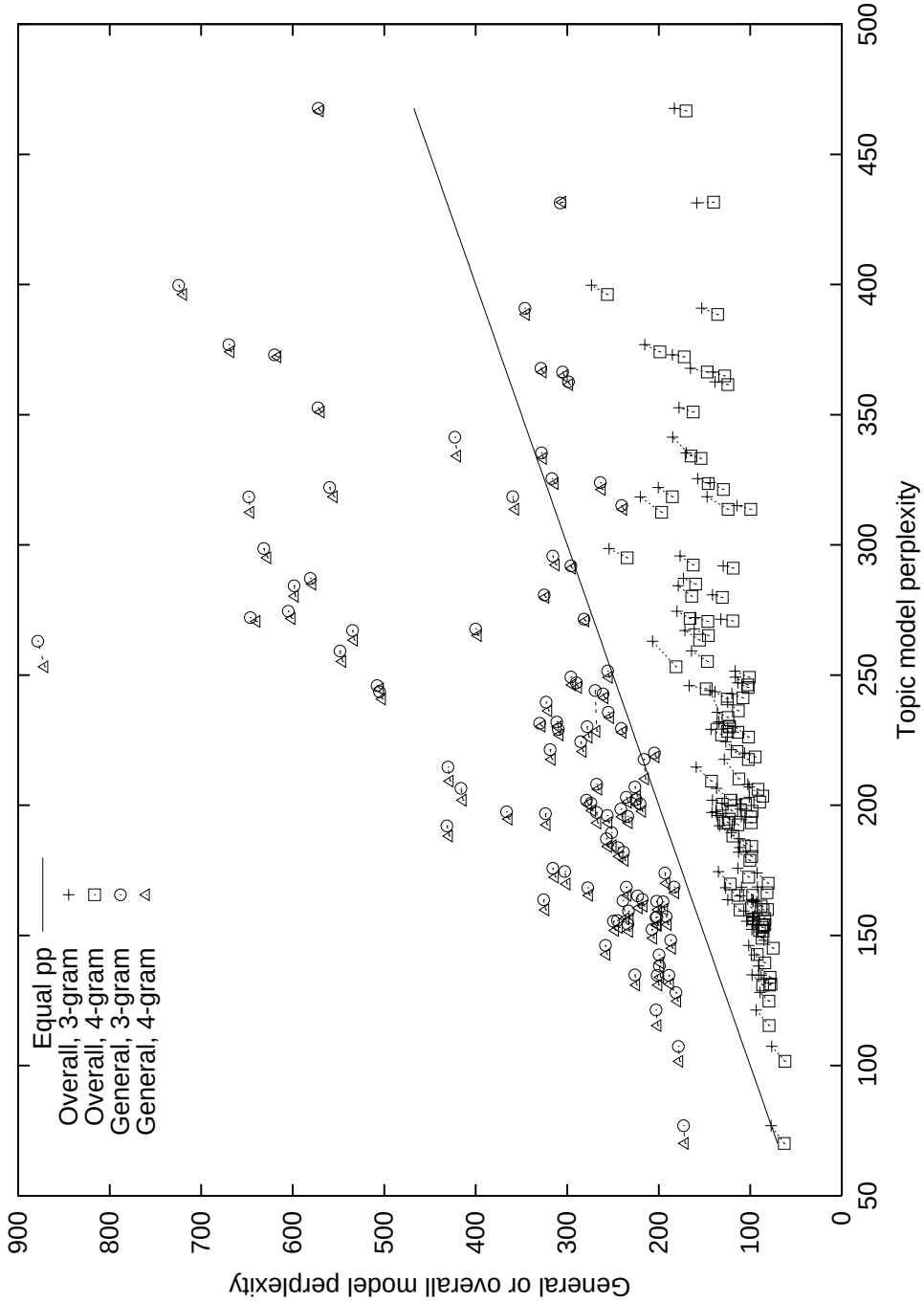


Figure C.5: Reclustered 4 and 8 topic sets – bigram perplexities of both agglomerative and *k*-means clustered topic models against general model.



Reclustered 100 topic set – topic model perplexities of both 3- and 4-grams against overall and general models. Lines join data points corresponding to the same topic and overall/general set – that is, they join corresponding 3- and 4-gram points.

Figure C.6: Please rotate page to read caption

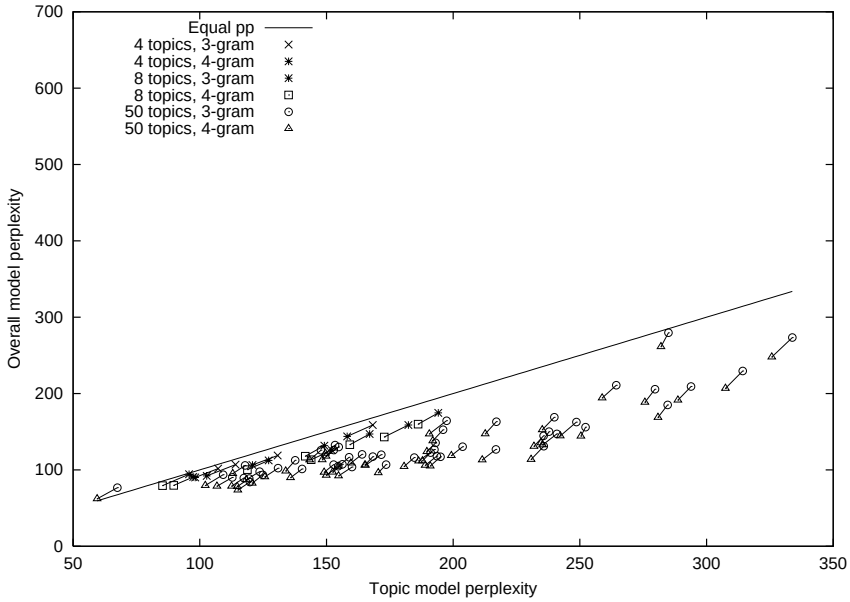


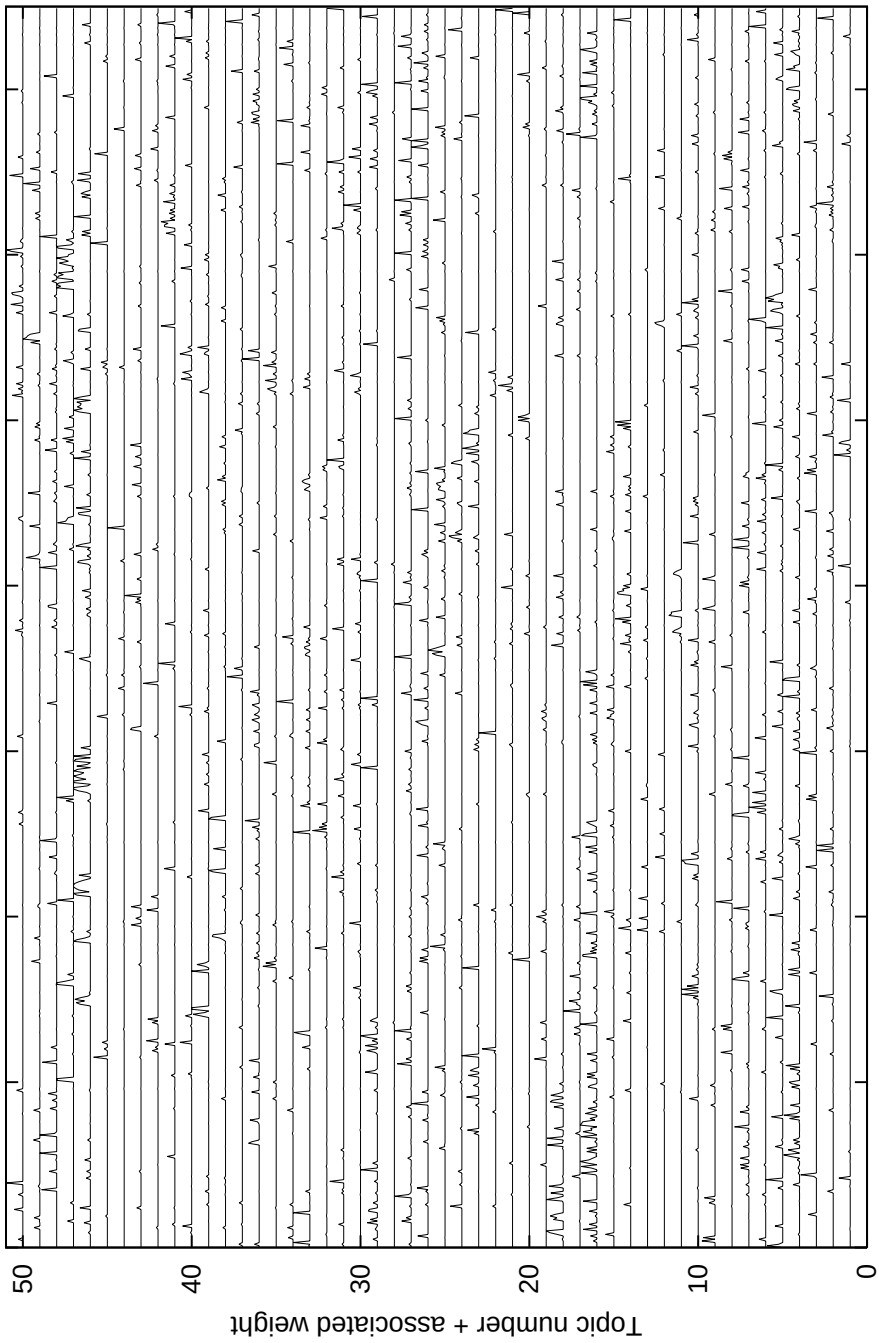
Figure C.7: Reclustered 4, 8 and 50 topic sets – 3- and 4-gram perplexities of k-means clustered topic models against the overall model.

D. Topic weights

This appendix gives example topic weights as computed with the EM algorithm in the work reported in chapter 4. This allows a measure of topic purity to be assessed and also shows the correlation between adjacent lattices. Primarily, however, it is presented to show for interest's sake how the weights change. It can be clearly seen that for each lattice there are only a very few topics which have any significant weight, and it is frequently the case that a single topic has a weight of nearly 1. This suggests that both the topic clustering and the weight selection methods were successful in finding significantly different topics and choosing between them. Furthermore, all topics can be seen to be used with a significant weighting for at least one utterance.

Each horizontal line represents the weight of a single topic, thus in this 50 topic example there are 50 such lines. As each line travels across the graph its height corresponds to the weight of that topic for the corresponding lattice as given by the horizontal axis. The vertical distance between each line corresponds to a weight of 1, with the line at its lowest point having a weight of 0. Should any line reach up to the base of the line above it, therefore, then that topic has a weight close to 1 at that point.

Please turn to the next page to view the graph



1003 class *n*-gram language model constructed using the 50 topic set – weight of each topic shown for each lattice in the evaluation set. See text on previous page for further details.

Figure D.1: Please rotate page to read caption

E. Additional topic model results

				Perplexity				
				1-best		Reference		Word Error
Classes	**Topics**	**W**	**EM**	**Prev**	**Curr**	**Prev**	**Curr**	**Rate (%)**
-	-	✓	-	**174.84**				**17.6**
503	-	-	-	**259.53**				**19.6**
503	-	✓	-	**160.54**				**17.3**
1003	-	-	-	**219.77**				**18.5**
1003	-	✓	-	**160.50**				**17.1**
2003	-	-	-	**194.86**				**18.1**
2003	-	✓	-	**161.51**				**17.2**
503	50	-	-	260.92	226.32	258.68	222.16	19.1
503	50	✓	-	254.41	220.40	252.65	216.16	18.9
503	50	-	✓	157.91	151.56	157.40	150.49	17.2
503	50	✓	✓	157.47	150.71	156.98	149.36	17.2
1003	50	-	-	223.80	200.04	221.82	196.53	18.2
1003	50	✓	-	218.47	195.72	217.10	192.20	18.1
1003	50	-	✓	159.06	153.83	158.58	152.75	16.9
1003	50	✓	✓	158.53	152.99	158.18	151.78	16.9
2003	50	-	-	199.44	182.17	198.51	179.24	17.8
2003	50	✓	-	195.31	179.32	194.58	176.32	17.8
2003	50	-	✓	161.01	156.32	160.78	155.30	17.1
2003	50	✓	✓	160.33	155.80	160.16	154.57	17.1

Table E.1: Language model perplexities. *Key:* **W** = word model interpolated where ticked – class model only where not ticked; **EM** = weights found using EM algorithm where ticked – single best topic where not ticked; **1-best** = lattice 1-best text used to choose topics; **Reference** = reference transcription used to choose topics; **Prev** = preceding utterance used in topic estimation; **Curr** = current utterance used in topic estimation. *Continued overleaf...*

				Perplexity				
				1-best		Reference		Word Error
Classes	Topics	W	EM	Prev	Curr	Prev	Curr	Rate (%)
503	100	-	-	266.07	224.95	261.86	219.81	19.0
503	100	✓	-	252.01	216.86	255.54	215.25	18.9
503	100	-	✓	158.10	150.91	157.29	149.54	17.2
503	100	✓	✓	157.43	149.52	157.22	148.71	17.3
1003	100	-	-	226.10	198.87	225.32	194.72	18.1
1003	100	✓	-	220.28	193.45	218.69	189.14	18.0
1003	100	-	✓	159.11	153.09	158.80	151.86	17.0
1003	100	✓	✓	158.56	151.98	158.16	150.47	17.0
2003	100	-	-	201.39	181.79	200.17	178.13	17.8
2003	100	✓	-	196.55	177.96	195.82	174.43	17.7
2003	100	-	✓	160.97	155.80	160.71	154.56	17.2
2003	100	✓	✓	160.37	155.04	160.17	153.57	17.2
503	250	-	-	275.62	225.63	270.41	219.91	19.1
503	250	✓	-	263.40	215.31	261.73	209.20	18.8
503	250	-	✓	158.69	150.12	157.83	148.61	17.3
503	250	✓	✓	158.00	148.52	157.44	146.53	17.3
1003	250	-	-	233.59	200.07	232.68	195.14	18.3
1003	250	✓	-	225.85	192.66	224.24	187.37	17.9
1003	250	-	✓	159.63	152.45	159.38	151.01	17.0
1003	250	✓	✓	158.98	151.15	158.68	149.37	16.9
2003	250	-	-	207.28	182.77	205.29	178.62	17.9
2003	250	✓	-	201.78	177.83	201.31	173.53	17.7
2003	250	-	✓	161.70	155.43	161.22	154.05	17.2
2003	250	✓	✓	161.07	154.41	160.93	152.69	17.2
1003	500	-	-	238.31	202.08	237.82	197.42	18.3
1003	500	✓	-	230.96	193.86	228.93	188.11	18.0
1003	500	-	✓	160.27	152.95	160.12	151.60	17.0
1003	500	✓	✓	159.77	151.32	159.30	149.37	17.0

Table E.2: *...continued from overleaf*. Language model perplexities. *Key:* **W** = word model interpolated where ticked – class model only where not ticked; **EM** = weights found using EM algorithm where ticked – single best topic where not ticked; **1-best** = lattice 1-best text used to choose topics; **Reference** = reference transcription used to choose topics; **Prev** = preceding utterance used in topic estimation; **Curr** = current utterance used in topic estimation

F. Additional pair model results

This appendix contains additional graphs assessing the performance of the pair class *n*-gram language model, provided to complement those included in the main body of chapter 5. These graphs provide results for different values of *n* to those *n*-gram results given previously.

F.1 1992 corpus

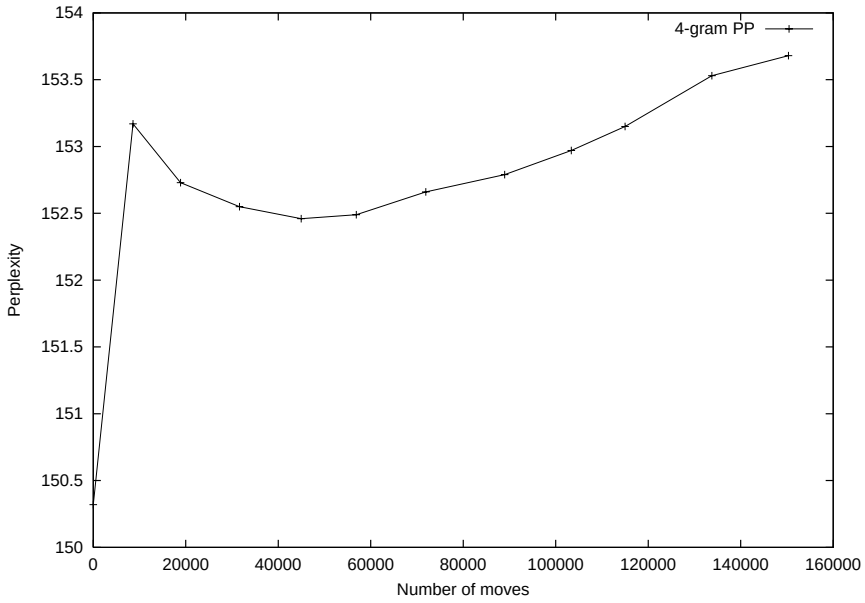


Figure F.1: Effect on test set perplexity when moving pairs from their start location when assessed using a 4-gram class pair model [1992 corpus]. Each data point is separated by 24 hours of CPU time.

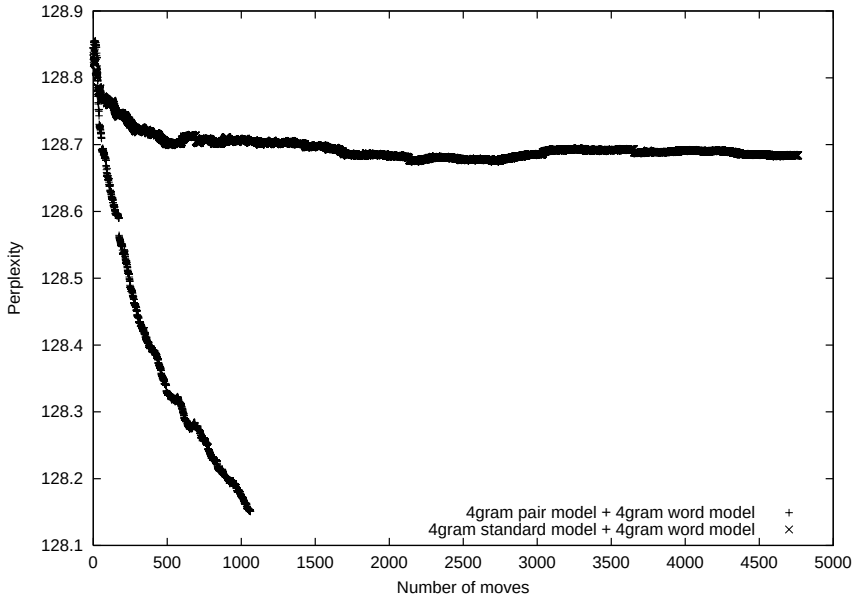


Figure F.2: Perplexity comparison on a per-move basis between 4-gram pair and standard class models – each interpolated with a 4-gram word model [1992 corpus]

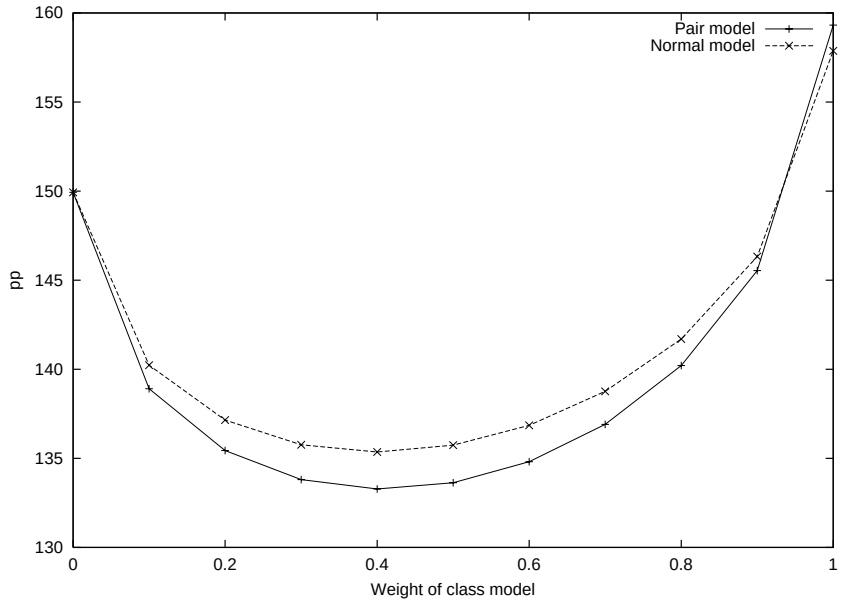


Figure F.3: Perplexity when linearly interpolating 3-gram word and 3-gram class pair models together with different weights [1992 corpus]

F.2 Full corpus

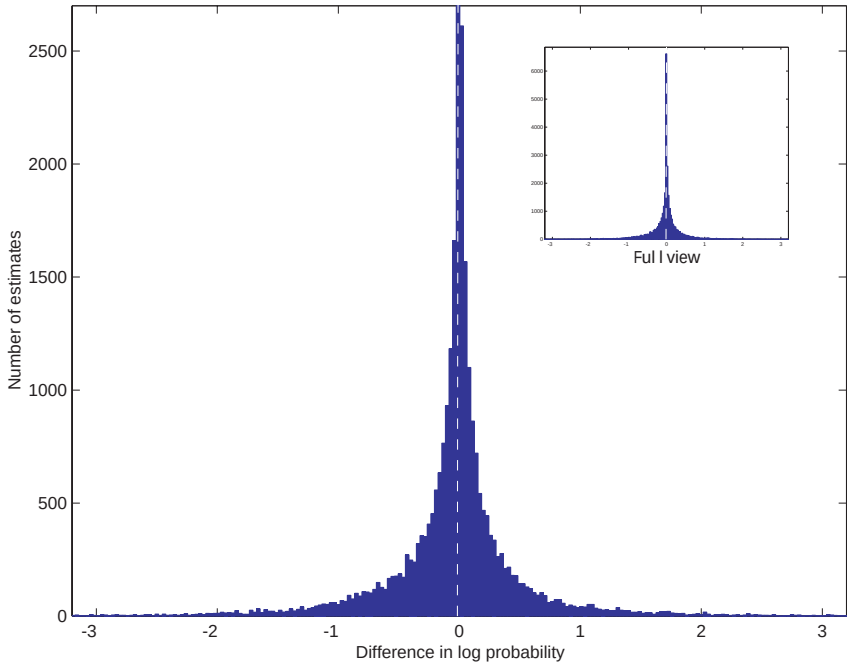


Figure F.4: Histogram showing number of changes in log probability of various magnitudes – change from a 4-gram standard to a 4-gram pair class model [Full corpus]

Next graph on following page...

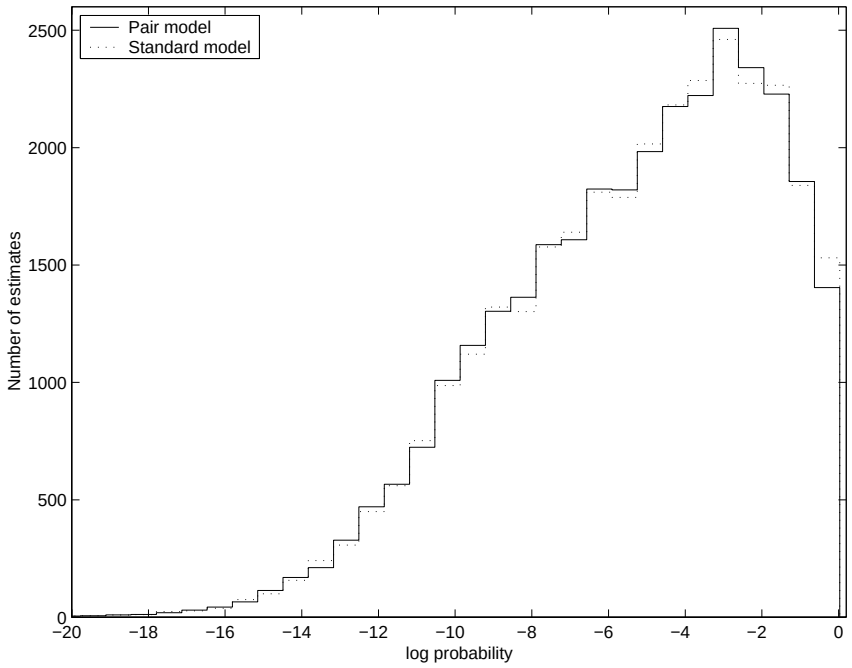


Figure F.5: Distribution of word probabilities on test text in pair and standard 4-gram class models [Full corpus]

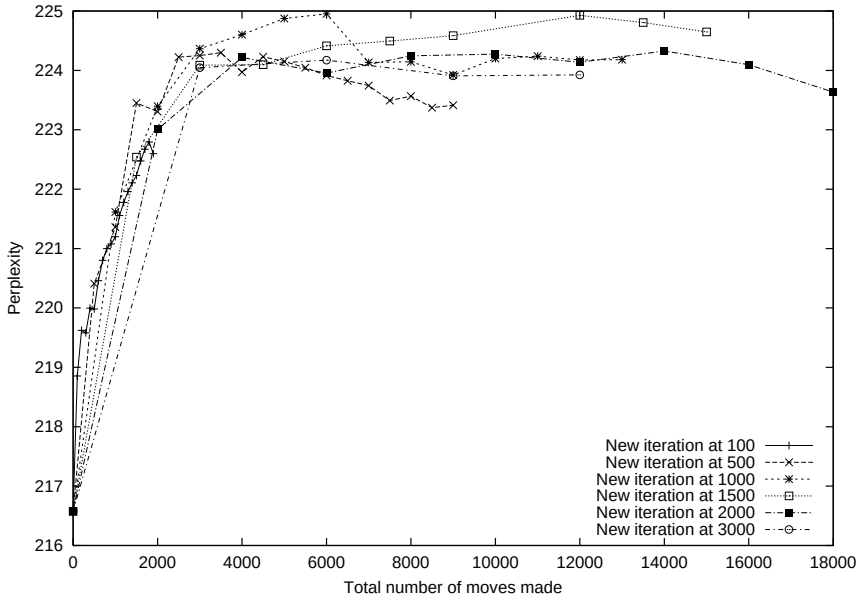


Figure F.6: 4-gram pair-model perplexity on test set when starting a new iteration after a given number of moves – plotted by number of moves [Full corpus]

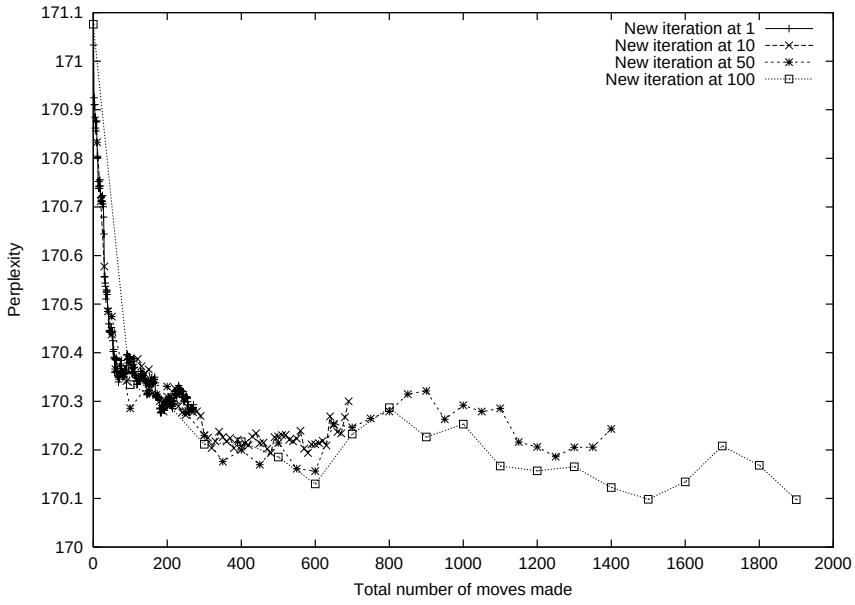


Figure F.7: Trigram word + pair-model perplexity on test set when starting a new iteration after a small number of moves – plotted by number of moves [Full corpus]

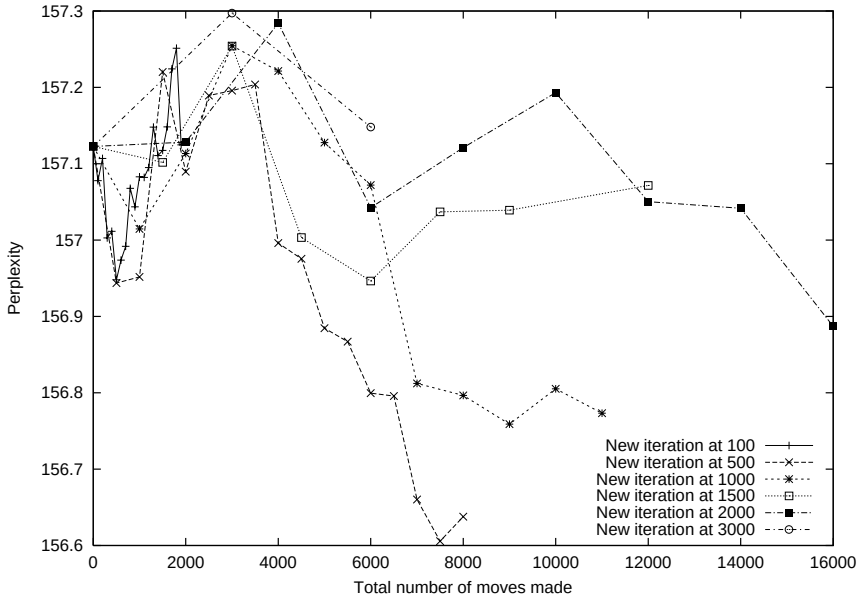


Figure F.8: 4-gram word- + pair-model perplexity on test set when starting a new iteration after a given number of moves – plotted by number of moves [Full corpus]

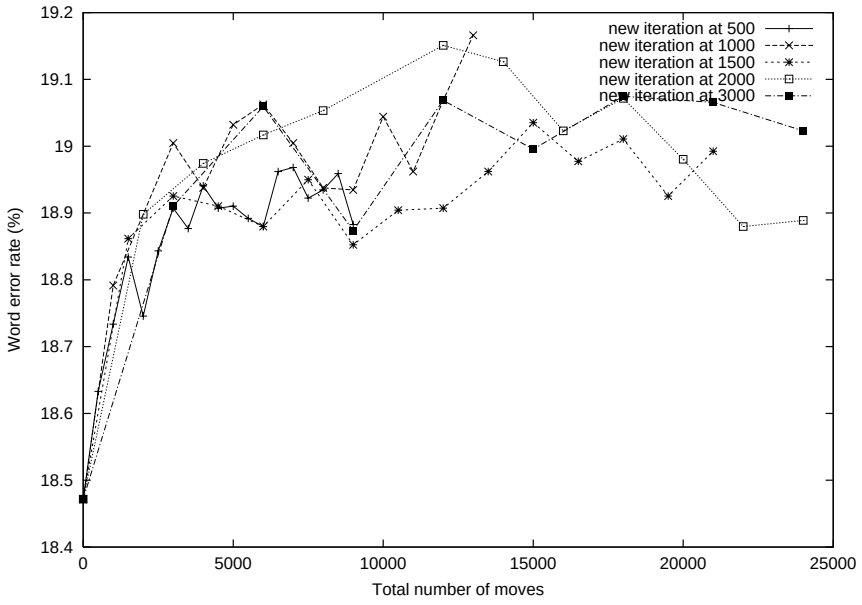


Figure F.9: 4-gram pair class model word error rates shown for different restart points – each point represents the end of an iteration

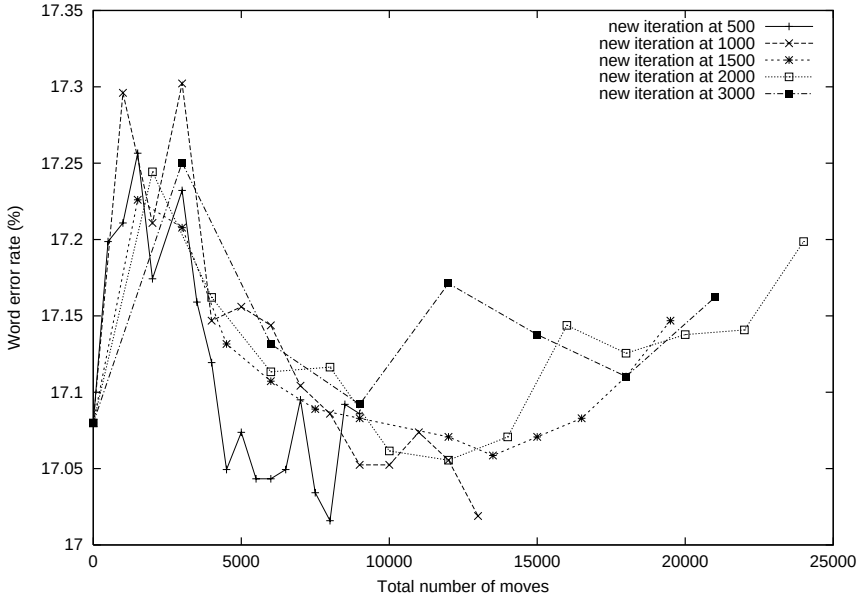


Figure F.10: 4-gram pair class + 4-gram word model word error rates shown for different restart points – each point represents the end of an iteration

Bibliography

[Ando et al 1998] A. Ando, A. Kobayashi and T. Imai, "A Thesaurus-Based Statistical Language Model for Broadcast News Transcription"; *Proceedings of the International Conference on Spoken Language Processing* 1998 vol. 6 pp. 2383-2386

[Bahl et al 1983] L.R. Bahl, F. Jelinek and R.L. Mercer, "A Maximum Likelihood Approach to Continuous Speech Recognition"; *IEEE Transactions on Pattern Analysis and Machine Intelligence* March 1983, vol. 5 no. 2 pp. 179-190

[Bayes 1763] Rev. T. Bayes, "An Essay Toward Solving a Problem in the Doctrine of Chances"; *Philosophical Transactions of the Royal Society*, London, 1763, vol. 53, pp. 370-418

[Bigi et al 1998] B. Bigi, R. De Mori, M. El-Bèze and T. Spriet, "Detecting Topic Shifts using a Cache Memory"; *Proceedings of the International Conference on Spoken Language Processing* 1998, vol. 6 pp. 2331-2334

[Bimbot et al 1994] F. Bimbot, R. Pieraccini, E. Levin and B. Atal, "Modèles de Séquences à Horizon Variable: Multigrams"; *Proceedings of the XXth JEP, Trégastal, France* 1994

[Bimbot et al 2001] F. Bimbot, M. El-Bèze, S. Igounet, M. Jardino, K. Smaili and I. Zitouni, "An alternative scheme for perplexity estimation and its assessment for the evaluation of language models"; *Computer Speech and Language* 2001, vol. 15 pp. 1-13

[Bonafonte and Mariño 1996] A. Bonafonte and J.B. Mariño, "Language Modeling using X-Grams"; *Proceedings of the International Conference on Spoken Language Processing* 1996

[Bonafonte and Mariño 1998] A. Bonafonte and J.B. Mariño, "Using X-Gram for Efficient Speech Recognition"; *Proceedings of the International Conference on Spoken Language Processing* 1998 vol. 6 pp. 2559-2562

[Brown et al 1992] P.F. Brown, P.V. de Souza, R.L. Mercer, V.J. Della Pietra, J.C. Lai, "Class-Based N-gram models of Natural Language"; *Computational Linguistics*, vol. 18-4 pp. 467-479

[Carlson 1996] B.A. Carlson, "Unsupervised Topic Clustering of Switchboard Speech Messages"; *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing* 1996, vol. 1 pp. 315-318

[Chelba and Jelinek 1998] C. Chelba and F. Jelinek, "Exploting Syntactic Structure for Language Modeling"; *Proceedings of COLING-ACL (17th International Conference on Computational Linguistics and the 36th Annual Meeting of the Association for Computational Linguistics)* 1998, vol. 1 pp. 225-231

[Chelba and Jelinek 1999] C. Chelba and F. Jelinek, "Recognition Performance of a Structured Language Model"; *Proceedings of the European Conference on Speech Communication and Technology* 1999, vol. 4 pp. 1567-1570

[Chelba and Jelinek 2000] C. Chelba and F. Jelinek, "Structured Language Modeling"; *Computer Speech and Language* 2000, vol. 14 pp. 283-332

[Chen (S.) et al 1998] S. Chen, D. Beeferman and R. Rosenfeld, "Evaluation Metrics for Language Models"; *DARPA Broadcast News Transcription and Understanding Workshop* 1998, pp. 275-280

[Chen (S.S.) et al 1998] S. Chen, M.J.F. Gales, P.S. Gopalakrishnan, R.A. Gopinath, H. Printz, D. Kanevsky, P. Olsen and L. Polymenakos, "IBM's LVCSR System for Transcription of Broadcast News Used in the 1997 HUB4 English Evaluation"; *DARPA Broadcast News Transcription and Understanding Workshop* 1998, pp. 69-74

[Clarkson 1999] P.R. Clarkson, "Adaptation of Statistical Language Models for Automatic Speech Recognition"; *Ph.D Thesis, University of Cambridge* 1999

[Clarkson and Robinson 1997] P. Clarkson and T. Robinson, "Language Model Adaptation Using Mixtures and an Exponentially Decaying Cache"; *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing* 1997

[Clarkson and Robinson 1998] P. Clarkson and T. Robinson, "The Applicability of Adaptive Language Modelling for the Broadcast News Task"; *Proceedings of the International Conference on Spoken Language Processing* 1998, vol. 5 pp. 1699-1702

[Clarkson and Robinson 1999] P. Clarkson and T. Robinson, "Towards Improved Language Model Evaluation Measures"; *Proceedings of the European Conference on Speech Communication and Technology* 1999, vol. 5 pp. 1927-1930

[Clarkson and Robinson 2001] P. Clarkson and T. Robinson, "Improved language modelling through better language model evaluation measures"; *Computer Speech and Language* 2001, vol. 15 pp. 39-53

[Cook and Robinson 1998] G.D Cook and A.J. Robinson, "The 1997 Abbot System for the Transcription of Broadcast News"; ; *DARPA Broadcast News Transcription and Understanding Workshop* 1998, pp. 49-54

[DARPA BN 1998] DARPA Broadcast News, "Proceedings of the Broadcast News Transcription and Understanding Workshop"; *Morgan Kaufmann Publishers* 1998

[Darroch and Ratcliff 1972] J.N. Darroch and D. Ratcliff, "Generalized Iterative Scaling for Log-Linear Models"; *The Annals of Mathematical Statistics* 1972, vol. 43 pp. 1470-1480

[Deligne and Bimbot 1995] S. Deligne and F. Bimbot, "Language Modeling by Variable Length Sequences: Theoretical Formulation and Evaluation of Multigrams"; *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing* 1995 pp. 169-172

[Dempster et al 1977] A.P. Dempster, M.M. Laird and D.B. Rubin, "Maximum Likelihood from Incomplete Data via the EM Algorithm"; *Journal of the Royal Statistical Society* 1977, vol. 39 pp. 1-38

[Dharanipragada et al 1999] S. Dharanipragada, M. Franz, J.S. McCarley, S. Roukos and T. Ward, "Story Segmentation and Topic Detection in the Broadcast News Domain"; *DARPA Broadcast News Transcription and Understanding Workshop* 1999

[Donnelly et al 1999] P.G. Donnelly, F.J. Smith, E. Sicilia and J. Ming, "Language Modelling with Hierarchical Domains"; *Proceedings of the European Conference on Speech Communication and Technology* 1999, vol. 4 pp. 1575-1578

[Fischer and Kunzmann 2000] V. Fischer and S.J. Kunzmann, "Acoustic Language Model Classes for a Large Vocabulary Continuous Speech Recogniser"; *Proceedings of the International Conference on Spoken Language Processing* 2000

[Gallwitz et al 1996] F. Gallwitz, E. Nöth and H. Niemann; "A Category Based Approach for Recognition of Out-of-Vocabulary Words"; *Proceedings of the International Conference on Spoken Language Processing* 1996

[Gillett and Ward 1998] J. Gillett and W. Ward, "A Language Model combining Trigrams and Stochastic Context-Free Grammars"; *Proceedings of the International Conference on Spoken Language Processing* 1998, vol. 6 pp. 2319-2322

[Gillick and Cox 1989] L. Gillick and S. Cox, "Some Statistical Issues in the Comparison of Speech Recognition Algorithms"; *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing* 1989, vol. 1 pp. 532-535

[Good 1953] I.J. Good, "The Population Frequencies of Species and the Estimation of Population Parameters"; *Biometrika* 1953, vol. 40 (3,4) pp. 237-264

[Goodman 2000] J. Goodman, "The State of the Art in Language Modeling"; *Language Technology Joint Conference ANLP-NAACL2000 – Applied Natural Language Processing and the North American Chapter of the Association for Computational Linguistics* 2000; tutorial – slides available from `http://www.research.microsoft.com/~joshuago/`

[Goodman 2001/ICASSP] J. Goodman, "Classes for Fast Maximum Entropy Training"; *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing* 2001

[Goodman 2001/MS] J. Goodman, "A Bit of Progress in Language Modelling (Extended Version)"; *Microsoft Technical Report MSR-TR-2001-72* 2001, see in particular chapter 11; `http://www.research.microsoft.com/~joshuago/` – also due to appear in a shortened form in *Computer Speech and Language*

[Hacioglu and Ward 2001] K. Hacioglu and W. Ward, "Dialog-context Dependent Language Modelling combining N-grams and Stochastic Context-free Grammars"; *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing* 2001

[Huang et al 1993] X. Huang, F. Alleva, H. Hon, M. Hwang, K. Lee and R. Rosenfeld, "The SPHINX-II Speech Recognition System: An Overview"; *Computer Speech and Language* 1993, vol. 2 pp. 137-148

[Iyer 1994] R. Iyer, "Language Modeling with Sentence-level Mixtures"; *M.Sc Thesis, University of Bombay* 1994

[Iyer and Ostendorf 1996] R. Iyer and M. Ostendorf, "Modelling Long Distance Dependence in Language: Topic Mixtures vs. Dynamic Cache Models"; *Proceedings of the International Conference on Spoken Language Processing* 1996, vol. 1 pp. 236-239

[Iyer et al 1994] R. Iyer, M. Ostendorf and J.R. Rohlicek, "Language Modelling with Sentence-Level Mixtures"; *Proceedings of the ARPA Workshop on Human Language Technology* 1994, pp. 82-87

[Jang and Hauptmann 1998] P.J. Jang and A.G. Hauptmann,"Hierarchical Cluster Language Modeling With Statistical Rule Extraction for Rescoring N-Best Hypotheses During Speech Decoding"; *Proceedings of the International Conference on Spoken Language Processing* 1998 vol. 6 pp. 2423-2426

[Jardino 1994] M. Jardino, "A Class Bigram Model for Very Large Corpus"; *Proceedings of the International Conference on Spoken Language Processing* 1994, vol. 2 pp. 867-70

[Jardino 1996] M. Jardino, "Multilingual Stochastic N-Gram Class Language Models"; *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing* 1996

[Jardino and Adda 1993/Eurospeech] M. Jardino and G. Adda, "Language Modelling for CSR of Large Corpus using Automatic Classification of Words"; *Proceedings of the European Conference on Speech Communication and Technology* 1993 vol. 2 pp. 1191-4

[Jardino and Adda 1993/ICASSP] M. Jardino and G. Adda, "Automatic Word Classification using Simulated Annealing"; *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing* 1993, vol. 2 pp. 41-44

[Jelinek 1976] F. Jelinek, "Continuous Speech Recognition by Statistical Methods"; *Proceedings of the IEEE* 1976, vol. 64(4) pp. 532-556

[Jelinek 1977] F. Jelinek, R. Mercer, L.R. Bahl and J.K. Baker, "Perplexity – a measure of the difficulty of speech recognition tasks"; *Proceedings of the 94th Meeting of the Acoustical Society of America* 1977, 62:S63 supplement no. 1

[Jelinek 1990] F. Jelinek, "Self-Organized Language Modeling for Speech Recognition"; *in "Readings in Speech Recognition", Morgan Kaufmann Publishers* 1990, pp. 450-506

[Jelinek et al 1991] F. Jelinek, B. Merialdo, S. Roukos and M. Strauss, "A Dynamic Language Model for Speech Recognition"; *Proceedings of the DARPA Workshop on Speech and Natural Language Understanding* 1991, pp. 293-295

[Joachims 1996] T. Joachims, "A Probabilistic Analysis of the Rocchio Algorithm with TFIDF for Text Categorization"; *Carnegie Mellon University technical report CMU-CS-96-118*, March 1996

[Johansson et al 1986] S. Johansson, R. Atwell, R. Garside and G. Leech, "The Tagged LOB Corpus User's Manual"; *Norwegian Computing Centre for the Humanities, Bergen* 1986

[Katz 1987] S.M. Katz, "Estimation of Probabilities from Sparse Data for the Language Model Component of a Speech Recognizer"; *IEEE Transactions on Acoustic, Speech and Signal Processing* 1987, vol. 35 no. 3 pp. 400-401

[Klakow 1998] D. Klakow, "Log-Linear Interpolation of Language Models"; *Proceedings of the International Conference on Spoken Language Processing* 1998, vol. 5 pp. 1695-1698

[Kneser 1996] R. Kneser, "Statistical Language Modeling Using a Variable Context Length"; *Proceedings of the International Conference on Spoken Language Processing* 1996

[Kneser and Ney 1993] R. Kneser and H. Ney, "Improved Clustering Techniques for Class-Based Statistical Language Modelling"; *Proceedings of the European Conference on Speech Communication and Technology* 1993, pp. 973-976

[Kneser and Steinbiss 1993] R. Kneser and V. Steinbiss, "On the Dynamic Adaptation of Stochastic Language Models"; *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing* 1993, pp. 586-589

[Kobayashi and Kobayashi 1999] N. Kobayashi and T. Kobayashi, "Class-combined Word *N*-Gram for Robust Language Modeling"; *Proceedings of the European Conference on Speech Communication and Technology* 1999, vol. 4 pp. 1599-1602

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

[Kuhn and De Mori 1992] R. Kuhn and R. De Mori, "Corrections to 'A Cache-Based Natural Language Model for Speech Recognition' "; *IEEE Transactions on Pattern Analysis and Machine Intelligence* 1992, vol. 14 pp. 691-692

[Kuo and Reichl 1999] H.J. Kuo, W. Reichl, "Phrase-based Language Models for Speech Recognition"; *Proceedings of the European Conference on Speech Communication and Technology* 1999 vol. 4 pp. 1595-1598

[Lafferty and Suhm 1996] J. Lafferty and B. Suhm, "Cluster Expansions and Iterative Scaling for Maximum Entropy Language Models"; *in "Maximum Entropy and Bayesian Methods", Kluwer Academic Publishers* 1996

[Maltese et al 2001] G. Maltese, P. Bravetti, H. Crépy, B.J. Grainger, M. Herzog and F. Palou, "Combining word- and class-based language models: A comparative study in several languages using automatic and manual word-clustering techniques"; *Proceedings of the European Conference on Speech Communication and Technology* 2001, vol. 1 pp. 21-24

[Marcus et al 1995] M. Marcus, B. Santorini and M. Marcinkiewicz, "Building a Large Annotated Corpus of English: The Penn Treebank"; *Computational Linguistics* vol. 19 pp.313-330

[Martin et al 1999] S. Martin, C. Hamacher, J. Liermann, F. Wessel and H. Ney, "Assessment of Smoothing Methods and Complex Stochastic Language Modeling"; *Proceedings of the European Conference on Speech Communication and Technology* 1999, pp. 1939-1942

[Martin, Liermann and Ney 1998] S. Martin, J. Liermann and H. Ney, "Algorithms for bigram and trigram word clustering"; *Speech Communication 24* 1998, pp. 19-37

[Meteer and Rholicek 1993] M. Meteer and J. R. Rohlicek, "Statistical Language Modelling combining N-gram and Context-free Grammars"; *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing* 1993, vol. 2 pp. 37-40

[Miller and Alleva 1996] J.W. Miller and F. Alleva, "Evaluation of a Language Model using a Clustered Model Backoff"; *Proceedings of the International Conference on Spoken Language Processing* 1996

[Moore and Young 2000] G.L. Moore and S.J. Young, "Class-based language model adaptation using mixtures of word-class weights"; *Proceedings of the International Conference on Spoken Language Processing* 2000

[Mori et al 1998] S. Mori, M. Nishimura and N. Itoh, "Word Clustering for a Word Bigram Model"; *Proceedings of the International Conference on Spoken Language Processing* 1998, vol. 6 pp. 2467-2470

[Nadas 1985] A. Nadas, "On Turing's Formula for Word Probabilities"; *IEEE Transactions on Acoustics, Speech and Signal Processing* 1985, vol. 33(6) pp. 1414-1416

[Ney and Essen 1993] H. Ney and U. Essen, "Estimating 'Small' Probabilities by Leaving-One-Out"; *Proceedings of the European Conference on Speech Communication and Technology* 1993, vol. 3 pp. 2239-2242

[Ney et al 1994] H. Ney, U. Essen and R. Kneser, "On Structuring Probabilistic Dependencies in Stochastic Language Modelling"; *Computer, Speech and Language* 1994, vol. 8:1 pp. 1-38

[Niesler 1997] T. Niesler, "Category-based Statistical Language Models"; *Ph.D Thesis, University of Cambridge* 1997

[Niesler and Woodland 1996/ICASSP] T.R. Niesler and P.C. Woodland, "A Variable-Length Category-Based *N*-Gram Language Model"; *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing* 1996, vol. 1 pp. 164-167

[Niesler and Woodland 1996/ICSLP] T.R. Niesler and P.C. Woodland, "Combination of Word-based and Category-based Language Models"; *Proceedings of the International Conference on Spoken Language Processing* 1996, vol. 1 pp. 220-223

[Niesler and Woodland 1997] T.R. Niesler and P.C. Woodland, "Modelling Word-Pair Relations in a Category-Based Language Model"; *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing* 1997, vol. 2 pp. 795-798

[Niesler et al 1998] T.R. Niesler, E.W.D. Whittaker and P.C. Woodland, "Comparison of part-of-speech and automatically derived category-based language models for speech recognition"; *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing* 1998

[Odell et al 1996] J.J. Odell and T.R. Niesler, "Lattice and Language Modelling Toolkit V2.0 – Reference Manual"; *Entropic Cambridge Research Laboratories Inc.* 1996

[Ogata and Ariki 1998] J. Ogata and Y. Ariki, "Indexing and Classification of TV News Articles based on Speech Dictation using Word Bigram"; *Proceedings of the International Conference on Spoken Language Processing* 1998, vol. 7 pp. 3265-3268

[Pietra et al 1995] S. Della Pietra, V. Della Pietra and J. Lafferty, "Inducing Features of Random Fields"; *Carnegie Mellon University Technical Report* 1995, CMU-CS-95-144

[Press et al 1989] W.H. Press, B.P. Flannery, S.A. Teukolsky and W.T. Vetterling, "Numerical Recipes"; *Cambridge University Press* 1989

[Printz and Olsen 2000] H. Printz and P. Olsen, "Theory and Practice of Acoustic Confusability"; *ISCA Tutorial and Research Workshop on Automatic Speech Recognition: Challenges for the New Millennium* 2000, pp. 77-84

[Ries et al 1995] K. Ries, F.D. Buø and Y. Wang, "Improved Language Modeling by Unsupervised Acquisition of Structure"; *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing* 1995

[Ries et al 1996] K. Ries, F.D. Buø and A. Waibel, "Class Phrase Models for Language Modeling"; *Proceedings of the International Conference on Spoken Language Processing* 1996

[Rosenfeld 1994] R. Rosenfeld, "Adaptive Statistical Language Modeling: A Maximum Entropy Approach"; *Ph.D Thesis, Carnegie Mellon University* 1994, CMU-CS-94-138

[Rosenfeld 1999] R. Rosenfeld, "Whole-Sentence Maximum Entropy Language Models"; *Public presentation, Department of Engineering, University of Cambridge*, 21st April 1999

[Samuelsson and Reichl 1999] C. Samuelsson and W. Reichl, "A Class-based Language Model for Large-Vocabulary Speech Recognition Extracted from Part-of-Speech Statistics"; *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing* 1999

[Samuelsson and Voutilainen 1997] C. Samuelsson and A. Voutilainen, "Comparing a Linguistic and a Stochastic Tagger"; *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics* 1997, pp. 246-253

[Sanker et al 1998] A. Sankar, F. Weng, Z. Rivlin, A. Stolcke and R.R. Gadde, "Development of SRI's 1997 Broadcast News Transcription System"; *DARPA Broadcast News Transcription and Understanding Workshop* 1998, pp. 91-96

[Schwartz and Chow 1990] R. Schwartz and Y. Chow, "The N-Best Algorithm: An Efficient and Exact Procedure for Finding the N Most Likely Sentences"; *Proceedings*

of the IEEE International Conference on Acoustics, Speech and Signal Processing 1990, pp. 81-84

[Sekine 1994] S. Sekine, "Automatic Sublanguage Identification for a New Text"; *Second Annual Workshop on Very Large Corpora* 1994, pp. 109-120

[Sekine 1998] S. Sekine, "Corpus based Parsing and Sublanguage Studies"; *Ph.D Thesis, New York University* 1998

[Seymore and Rosenfeld 1996] K. Seymore and R. Rosenfeld; *Proceedings of the International Conference on Spoken Language Processing*, 1996

[Seymore and Rosenfeld 1997/CMU] K. Seymore and R. Rosenfeld, "Large-scale Topic Detecion and Language Model Adaptation"; *Carnegie Mellon University technical report CMU-CS-97-152*, June 1997

[Seymore and Rosenfeld 1997/Eurospeech] K. Seymore and R. Rosenfeld, "Using Story Topics for Language Model Adaptation"; *Proceedings of the European Conference on Speech Communication and Technology* 1997, Vol. 4 pp. 1987-1990

[Seymore et al 1997] K. Seymore, S. Chen, M. Eskenazi and R. Rosenfeld, "Language and Pronunciation Modeling in the CMU 1996 Hub 4 Evaluation"; *DARPA Speech Recognition Workship* 1997, pp. 141-146

[Seymore et al 1998/BN] K. Seymore, S. Chen, S. Doh, M. Eskenazi, E. Gouvêa, B. Raj, M. Ravishankar, R. Rosenfeld, M. Siegler, R. Stern and E. Thayer, "The 1997 CMU Sphinx-3 English Broadcast News Transcription System"; *DARPA Broadcast News Transcription and Understanding Workshop* 1998, pp. 55-59

[Seymore et al 1998/ICASSP] K. Seymore, S. Chen and R. Rosenfeld, "Topic Adaptation for Language Modelling Using Unnormalized Exponential Models"; *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing* 1998

[Seymore et al 1998/ICSLP] K. Seymore, S. Chen and R. Rosenfeld, "Nonlinear Interpolation of Topic Models for Language Model Adaptation"; *Proceedings of the International Conference on Spoken Language Processing* 1998, Vol. 6 pp. 2503-2506

[Shannon 1948] C.E. Shannon, "A Mathematical Theory of Communication"; *The Bell System Technical Journal* 1948, vol. 27 pp. 379-423, 623-656. Available online at `http://galaxy.ucsd.edu/new/external/shannon.pdf`

[Siu and Ostendorf 1996] M. Siu and M. Ostendorf, "Modeling Disfluencies in Conversational Speech"; *Proceedings of the International Conference on Spoken Language Processing* 1996, pp. 386-389

[Siu and Ostendorf 1997] M. Siu and M. Ostendorf, "Variable N-Gram Language Modeling and Extensions for Conversational Speech"; *Proceedings of the European Conference on Speech Communication and Technology* 1997 pp. 2739-2742

[Siu and Ostendorf 2000] M. Siu and M. Ostendorf, "Integrating a Context-dependent Phrase Grammar in the Variable N-Gram Framework"; *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing* 2000

[Smaïli et al 1999] K. Smaïli, I. Zitouni and J.P. Haton, "Towards a Better Collaboration between a N-Class and a N-Gram Language Model"; *Proceedings of the International Workshop on Speech and Communication* 1999

[Sparck-Jones 1973] K. Sparck-Jones, "Index Term Weighting"; *Information Storage and Retrieval* 1973, vol. 9 pp. 619-633

[Stolcke 1998] A. Stolcke, "Entropy-based Pruning of Backoff Language Models"; *DARPA Broadcast News Transcription and Understanding Workshop* 1998, pp. 270-274

[Stolcke and Shriberg 1996] A. Stolcke and E. Shriberg, "Statistical Language Modeling for Speech Disfluency", *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing* 1996, pp. 405-408

[Ueberla 1995] J.P. Ueberla, "More Efficient Clustering of *N*-Grams for Statistical Language Modeling"; *Proceedings of the European Conference on Speech Communication and Technology* 1995 vol. 2 pp. 1257-1260

[Ueberla 1997] J.P. Ueberla, "Domain Adaptation with Clustered Language Models"; *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing* 1997, vol. 2 pp. 807-810

[Ueberla and Gransden 1996] J.P. Ueberla and I.R. Gransden, "Clustered Language Models with Context-Equivalent States"; *Proceedings of the International Conference on Spoken Language Processing* 1996

[Uebler and Niemann 1998] U. Uebler and H. Niemann, "Morphological Modeling of Word Classes for Language Models"; *Proceedings of the International Conference on Spoken Language Processing* 1998 vol. 5 pp. 1687-1690

[van Laarhoven and Aarts 1987] P.J.M. van Laarhoven and E.H. Aarts, "Simulated Annealing: Theory and Applications"; *D. Reidel Publishing Company* 1987

[van Mulbregt et al 1999] P. van Mulbregt, I. Carp, L. Gillick, S. Lowe and J. Yamron, "Segmentation of Automatically Transcribed Broadcast News Text"; *DARPA Broadcast News Transcription and Understanding Workship* 1999

[Wakita et al 1996] Y. Wakita, J. Kawai and H. Iida, "An Evaluation of Statistical Language Modeling for Speech Recognition using a Mixed Category of Both Words and Parts-of-Speech"; *Proceedings of the International Conference on Spoken Language Processing* 1996

[Walls et al 1999] F. Walls, H. Jin, S. Sista and R. Schwartz, "Topic Detection in Broadcast News"; *DARPA Broadcast News Transcription and Understanding Workship* 1999

[Wang et al 2000] Y. Wang, M. Mahajan, X. Huang, "A Unified Context-free Grammar and N-Gram Model for Spoken Language Processing"; *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing* 2000

[Whittaker 2000] E.W.D. Whittaker, "Statistical Language Modelling for Automatic Speech Recognition of Russian and English"; *Ph.D Thesis, University of Cambridge* 2000

[Whittaker and Woodland 2001] E.W.D. Whittaker and P.C. Woodland, "Efficient Class-based Language Modelling for Very Large Vocabularies"; *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing* 2001

[Woodland et al 1998] P.C. Woodland, T. Hain, S.E. Johnson, T.R. Niesler, A. Tuerk, E.W.D. Whittaker and S.J. Young, "The 1997 HTK Broadcast News Transcription System"; *DARPA Broadcast News Transcription and Understanding Workshop* 1998, pp. 41-48

[Wu and Khudanpur 2000] J. Wu and S. Khudanpur, "Efficient Training Methods for Maximum Entropy Language Models"; *Proceedings of the International Conference on Spoken Language Processing* 2000

[Yamron et al 1999] J.P. Yamron, I. Carp, L. Gillick, S. Lowe and P. van Mulbregt, "Topic Tracking in a News Stream"; *DARPA Broadcast News Transcription and Understanding Workshop* 1999

[Young et al 1997] S. Young, V. Valtchev and J. Odell, "The HLM Book" (for HLM version 1.0); *Unpublished - internal to Cambridge University Engineering Department / Entropic* 1997

[Young et al 1999] S. Young, D. Kershaw, J. Odell, D. Ollason, V. Valtchev and P. Woodland, "The HTK Book" (version 2.2); *Entropic* 1995-1999

[Zhu et al 1999] Z. Zhu, S.F. Chen and R. Rosenfeld, "Linguistic Features for Whole Sentence Maximum Entropy Language Models"; *Proceedings of the European Conference on Speech Communication and Technology* 1999, vol. 4 pp. 1807-1810

[Zitouni et al 1999] I. Zitouni, J.F. Mari, K. Smaïli and J.P. Haton, "Variable-Length Sequence Language Model for Large Vocabulary Continuous Dictation Machine"; *Proceedings of the European Conference on Speech Communication and Technology* 1999, vol. 4, pp. 1811-1814