Statistical Language Modelling for Automatic Speech Recognition of Russian and English

Edward W. D. Whittaker

Trinity College, University of Cambridge

and

Cambridge University Engineering Department.

Dissertation submitted to the University of Cambridge
for the degree of Doctor of Philosophy

Abstract

This dissertation concerns the development of statistical language models for use in automatic speech recognition systems. Natural language, which is a complex and variable phenomenon, has been shown to be modelled best using statistical language models. Large training corpora (comprising around one hundred million words) are employed as examples of real language usage to determine the parameters of the statistical models. The models are then used to assign probabilities to word sequences according to how likely each sequence is.

The focus of the experimental work in the dissertation is on techniques for modelling the Russian and English languages. An analysis is also made of the characteristics of a well-known English corpus and a specially collected and processed Russian corpus. Much research has been done on statistical modelling of English and the techniques are already well developed, however, when the same techniques are applied to Russian, they fare less well. The large number of unique words in the Russian language, which is a result of the high number of inflected word forms, is identified as the primary reason for this performance degradation. This characteristic of Russian (and many other languages besides) significantly increases the effects of data sparsity and means that model parameters are generally less well estimated. Nevertheless, data sparsity issues are not only confined to modelling of highly-inflected languages hence this dissertation also examines the use of modelling techniques for English.

The well established word N-gram language modelling technique uses N-tuples of words to capture local syntactic and semantic dependencies in a language. Such a model, however, suffers greatly from the effects of data sparsity. After examining the performance of word N-gram models on both corpora, the N-gram framework is extended to use classes of words as the modelling units. The ability of the class model to generalise to unseen word sequences is shown to reduce the effects of data sparsity and improve the robustness of models. The 'conventional' two-sided class model is examined together with the one-sided class model which has received little mention in the literature. Both class models employ data-driven clustering algorithms to determine the word classes automatically. The clustering algorithm which is developed for the one-sided class model performs the clustering operation significantly faster than the algorithm for the two-sided class model. The performance of both types of class N-gram models on the Russian data is found to be better than on the English data and also better than the Russian word N-gram models. The one-sided class model is shown to offer a competitive alternative to both the two-sided class model and the word model, especially where the classification of a large vocabulary into a large number of classes is required.

A novel variation on the N-gram modelling theme is also examined in the dissertation where, instead of using words or classes, the modelling units are chosen to be sub-word units. In this dissertation, these sub-word units are referred to as *particles*. The particle N-gram modelling scheme is motivated by the morphology of words which is more evident in the orthography of Russian words than in English words. A linguistics-based affix stripping algorithm is investigated together with two data-driven algorithms. The data-driven algorithms optimise the set of particles and word decompositions automatically by maximising the likelihood of the training

data using a particle bigram model. The resulting particles from each algorithm are employed in particle N-gram models where the probability of a word is given by the product of the word's component particle N-gram probabilities. The performance of the particle N-gram models is evaluated for both languages, and combinations of word and particle N-gram models are shown to give improved performance for both English and Russian. The improvements obtained with the particle modelling approach for English are found to be comparable to those obtained with the one-sided class models.

Declaration

This dissertation is the result of my own work, and where it draws on the work of others this is acknowledged at the appropriate points in the text. Some of the work has been published previously in conference proceedings (Niesler et al., 1998; Whittaker and Woodland, 1998). The length of this dissertation including appendices and footnotes is approximately 44,000 words and there are no more than 20 figures.

Acknowledgements

Unreserved thanks are due to Phil Woodland who has supervised me through three years of research. His insight and encouragement were invaluable.

My gratitude to several comrades-in-work for their thought provoking discussions and advice is no less deserving. In particular, James Christie, Philip Clarkson, Gunnar Evermann, Thomas Hain, Gareth Moore, Thomas Niesler and Harriet Nock merit special mention. Thanks also to Patrick Gosling, who, working behind the scenes, ensured that the excellent computing facilities in the speech, vision and robotics group were always maintained to a high standard. The loan of a computer from Tony Robinson was also much appreciated.

The research would not have been possible without the financial support of the Engineering and Physical Sciences Research Council. Additional funding from Trinity college and the Cambridge University Engineering department enabled me to attend several international conferences. The support of all these bodies is gratefully acknowledged.

For their contribution to my Russian education, I must thank Boris Korotkov, Tatiana Voloshinova and Nick Konoplev, without all of whom, this research would probably never have even begun.

And finally, parents, brothers and unnamed friends... thank you all for your patience and support.

Notation

$\mathcal{E}$	An event (e.g. a word trigram or word in some context)
$N(\mathcal{E})$	The number of times event $\mathcal{E}$ occurs in a sample
$P(\mathcal{E})$	The actual probability of event $\mathcal{E}$
$\hat{P}(\mathcal{E})$	An estimate of $P(\mathcal{E})$
d_r	Discount coefficient applied to an event that occurs r times
n_r	Number of events that occur r times
$\alpha(w_{i-1})$	Back-off weight for the context w_{i-1}
w	A word
$w_1, \dots, w_n$	A word sequence starting with word w_1 and ending with word w_n
h_i	The history of words $(w_1, \dots, w_{i-1})$ preceding word w_i
V	The set of vocabulary words
N_V	The vocabulary size
N_W	Number of words in training corpus
B	Number of unique word bigrams in training corpus for fixed vocabulary
c	A word equivalence class
C	Class mapping (word classification) function
N_C	The number of word equivalence classes
s	State or history equivalence class
S	State mapping or history classification function
u	A particle
U	Word decomposition function
N_P	The total number of particles in training corpus
Ψ	The set of particles
$L(w)$	The number of components (generally particles) in a word w
$l(w)$	The number of characters in a word w
H	Entropy
PP	Perplexity
LL	Log-likelihood
λ	Interpolation or thresholding parameter

Contents

1 Introduction **9**
1.1 Language modelling 9
1.2 Statistical speech recognition 10
1.2.1 Acoustic preprocessing 11
1.2.2 The acoustic model 11
1.2.3 The language model 11
1.3 Russian vs. English 12
1.4 Scope of the dissertation 13
1.4.1 Modelling N-tuples of word classes 14
1.4.2 Modelling N-tuples of particles 14
1.5 Organisation of the dissertation 14

2 Statistical language modelling **16**
2.1 Language model evaluation 16
2.1.1 Entropy and perplexity 16
2.2 Statistical language models 18
2.2.1 Markov chains 19
2.2.2 History equivalence classes 20
2.2.3 Word-based N-gram models 20
2.2.4 Class-based N-gram models 21
2.2.5 Morphological models 23
2.2.6 Higher-order N-gram models 24
2.2.7 Word-pair association models 26
2.2.8 Mixture-based models 28
2.2.9 Probabilistic context-free grammars 29
2.3 Robust parameter estimation 30
2.3.1 Probability estimation techniques 30
2.3.2 Frameworks for probability smoothing 33

2.3.3 Cross-validation 35

2.4 Methods of language model pruning 36

2.4.1 Cutoffs 36

2.4.2 Variable length N-gram models 37

2.4.3 Entropy-based pruning 38

2.5 Methods of model combination 39

2.5.1 Linear interpolation 39

2.5.2 Backing-off 40

2.5.3 Log-linear interpolation 40

2.5.4 Maximum entropy 40

2.6 Summary 41

3 The corpora and word-based language modelling 42

3.1 Language modelling tools 42

3.2 Corpora selection and preprocessing 43

3.2.1 The Russian corpus 44

3.2.2 The British National Corpus 44

3.3 Corpora characteristics 45

3.3.1 Vocabulary growth and corpus size 46

3.3.2 Coverage and vocabulary size 46

3.3.3 Russian word N-gram models 48

3.3.4 English word N-gram models 51

3.4 Permutated-history N-gram language model 53

3.5 Summary 53

4 Class-based language modelling 55

4.1 Introduction 55

4.2 Survey of existing techniques 57

4.2.1 Nuclear parts of speech 57

4.2.2 Hierarchical clustering 58

4.2.3 The exchange algorithm 59

4.2.4 Simulated annealing 60

4.3 Automatic word clustering experiments 61

4.3.1 Exchange algorithm for the two-sided class model 62

4.3.2 Experimental procedure 64

4.3.3 Results 65

4.3.4 Analysis of model performance 67

4.3.5 Discussion 71

4.4 Word recognition experiments 72

4.4.1 Results 73

4.4.2 Discussion 74

4.5 Summary 75

5 One-sided class-based language modelling 76

5.1 Introduction 76

5.2 Automatic word clustering experiments 77

5.2.1 Exchange algorithm for the one-sided class model 77

5.2.2 Position dependent word classification 80

5.2.3 Experimental procedure 80

5.2.4 Results 82

5.2.5 Analysis of model performance 85

5.2.6 Discussion 89

5.3 Summary 91

6 Particle-based language modelling 92

6.1 Introduction 92

6.2 Overview of existing techniques 94

6.2.1 Suffix-stripping 95

6.2.2 Morph decomposition in MITalk 95

6.2.3 Statistical-combinatorial modelling 96

6.2.4 Sticky pairs 96

6.2.5 Multigrams 97

6.3 Automatic methods for decomposing words 98

6.3.1 Linguistic affixes 100

6.3.2 Greedy word decomposition algorithm 103

6.3.3 Greedy particle selection algorithm 111

6.3.4 Effect of Normalization 121

6.3.5 Discussion 122

6.4 Summary 124

7 Conclusions and further work 126

7.1 Review of experimental work 126

7.1.1 Two-sided class language models 126

7.1.2 One-sided class language models 127

7.1.3 Particle language models 128

7.2 Suggested further work 129

7.2.1 Recognition performance of models 129

7.2.2 Improved automatic particle selection 129

7.2.3 Particle trigger models 130

7.3 Final summary 131

A English and Russian affixes 132

1

Introduction

The work in this dissertation concerns statistical language modelling of Russian and English for automatic speech recognition. In this introduction, the field of language modelling is first briefly described followed by an overview of statistical speech recognition. The relationship of the language model to the other components in a speech recognition system is also described. The reasons for choosing to investigate Russian and English are then given together with an examination of the fundamental differences between the two languages. The final section outlines the scope and the language modelling approaches that are developed throughout the dissertation.

1.1 Language modelling

Two obvious ways in which humans communicate with one another are through writing and speaking. Both activities involve the production and recognition of sequences of words and language modelling concerns the development of techniques and structures to model these word sequences.

Language models have found application in various fields including speech recognition, machine translation and spelling correction. Depending on the application, the task of the language model may be to determine whether a particular sequence of words is correct or valid in some sense: if a sequence satisfies a set of rules then it is allowed and if it does not, it is rejected. Such rule-based approaches have had limited use, especially where large vocabularies are involved, since rules are necessarily inflexible and the sequences of words produced by humans are often ungrammatical. The task of the language model may also be to determine how *likely* a particular sequence of words is and assign it some probability. This is the concern of the work in this dissertation together with the development and evaluation of statistical, as opposed to rule-based, models of language.

The development of statistical language models requires a vast quantity of example word sequences to make the probability estimates of sequences representative of the language in general. Fortunately, machine readable *text corpora* containing hundreds of millions of words are now readily available for many languages. How representative each corpus is of actual language usage, will depend on the domain from which the data was collected. For example, business newspapers will be representative of the style and content of written business journalism and are unlikely to be representative of conversational speech. It is clear however that real examples of language usage will be more useful and representative than a set of rules which describe some examples and inevitably fail for many other examples.

1.2 Statistical speech recognition

Speech recognition has been compared to the action of a "voice" typewriter (Jelinek, 1997) i.e. the automatic (unaided) conversion of spoken speech into written text. This process, which is performed with relative ease by humans, represents a formidable task to the designers of such systems. Large vocabulary[1], connected speech recognition systems generally aim to determine the most likely word sequence $\hat{W}$, from the huge number of possible sequence hypotheses W given a sequence of observed acoustic features A. Acoustic preprocessing is used to extract the features A from the speech waveform. The acoustic model of a recogniser matches word sequences with similar acoustic properties to the observed input while the task of the language model is to determine the likelihood of each hypothesised sequence. One way in which the recognition process can be represented is by finding that word sequence $\hat{W}$ which maximises the probability of the sequence given the acoustics A

$$\hat{W} = \arg\max_{W} P(W \mid A). \tag{1.1}$$

The conditional probability of the word sequence given the acoustics may be rearranged using Bayes' rule to give

$$P(W \mid A) = P(W)\frac{P(A \mid W)}{P(A)}. \tag{1.2}$$

Since $P(A)$ is independent of W the most likely word sequence may then be determined by

$$\hat{W} = \arg\max_{W} P(W)P(A \mid W). \tag{1.3}$$

The acoustic model of the speech recogniser computes $P(A \mid W)$ while $P(W)$ is handled by the language model.

Recognition involves a search process to determine the hypothesis which maximises the function given by Equation (1.1). The search process combines information from several different knowledge sources: the language model, the acoustic model and a pronunciation dictionary. A number of decoding strategies (Ney and Ortmanns, 1999) are currently popular including Viterbi beam search, stack decoding and multi-pass strategies that refine hypotheses using progressively more complex models for each pass. Since the possible search space is so vast, all decoding methods are forced to restrict the number of active hypotheses allowed at any time.

[1]Currently, 65,000 words is considered a "large" vocabulary.

This inevitably makes the search sub-optimal but the trade-off between recognition accuracy, speed and the computer memory requirements of the recogniser is always present.

1.2.1 Acoustic preprocessing

The signal processing *front-end* of a speech recogniser is responsible for the extraction of the features from the speech waveform. The acoustic signal is converted into a parametric representation which is generally at a lower information rate than the original signal. Further processing and pattern matching schemes can then be applied to the encoded representation. Most feature extraction methods involve an analysis of the short term spectral characteristics of the waveform. A commonly used set of features for representing the local spectral properties of the signal is the cepstral coefficients and their temporal derivatives.

1.2.2 The acoustic model

In a large vocabulary recogniser the acoustic models usually represent phone units. Word models may then be generated using a dictionary of word pronunciations. Hidden Markov models (Rabiner, 1989) represent the most widely used technique in contemporary speech recognition systems for building acoustic models, although other techniques, for example recurrent neural networks (Robinson and Fallside, 1991), have also been used successfully.

For building systems based on hidden Markov models, the training data usually comprises a set of parameterised acoustic sequences for which there is generally only a word-level transcription. A pronunciation dictionary may then be used to obtain a phonetic transcription from the word-level transcription. The standard, maximum likelihood training process involves the optimisation of the component phone models' parameters $P(A \mid \mathcal{M})$ so as to increase the likelihood of the acoustic training data. Alternative sub-word models to phone models include demisyllables and syllables. Word models are obtained in a similar manner, in each case through concatenation of component sub-word models as specified by the pronunciation dictionary. Context-dependent models both for word-internal and cross-word contexts for modelling coarticulation effects have also proved necessary to achieve good recognition performance.

1.2.3 The language model

The task of the language model in a large vocabulary recogniser is to estimate the probability of each hypothesised sequence of words: $P(W)$. This probability is used to guide the search towards linguistically probable sequences of words by assigning them higher probabilities than are assigned to unlikely sequences of words. The language model must ensure that no sequence of words is impossible, but that some sequences are more likely than others. The language model helps to differentiate between acoustically similar words or word sequences according to how likely the competing sequences are.

During recognition, requests are made to the language model for the probability of a word given a hypothesised sequence, or path, of previous words. The language model score is combined with the acoustic score for the current word to determine how probable the hypothesised sequence of words is. Some search strategies require that the context of previous words used in predicting the current word be restricted. Separate paths must be maintained for distinct hypotheses hence pruning of paths is essential to prevent the search becoming computationally unmanageable.

Before going on to consider the types of structural dependencies between words that will be used in the experiments in this dissertation, the motivation behind the modelling of the Russian and English languages will be examined.

1.3 Russian vs. English

No other language has received quite as much attention as English has in terms of the total research effort expended by the speech recognition community. Over the past decade, however, more attention has started to be paid to the recognition of other languages although commercial factors have generally influenced the direction of this research. Russian has received comparatively little attention and the first experimental large vocabulary speech recogniser for Russian was developed only recently at IBM (Kanevsky et al., 1996). Russian is one of several languages in the Slavic family and possesses significant properties which make it stand out as a potentially problematic language for recognition. These characteristics are shared to varying degrees by many other languages, not just the Slavic ones. It is therefore hoped, that methods which are applied successfully to Russian could also serve well for other languages.

It is worth beginning by making some preliminary observations about the nature of the Russian language. Russian words often exhibit clearer morphological patterns than can be found in English words. If a simplified model of a Russian verb is examined (Caflisch, 1995), the presence of several constituent parts can be determined: a *root* which can be thought of as responsible for the nuclear meaning of the verb, attached to which may be zero or more *derivational prefix(es)* and zero or one *suffix*, which together form a *stem*. The *stem* often acquires an entirely new lexical meaning with the presence of these affixes. An *inflection (inflectional suffix)*, which is appended to the stem, determines the grammatical case, gender, number etc. of the *word*. All the points in the word where the constituent parts are joined can be considered *morpheme boundaries*. Obviously, this is an idealised example, although the "synthetic" nature of Russian is also clearly visible in Russian nouns, adjectives and participles. An example of the constituents of a typical adjective are shown in Figure 1.1 for the Russian word "*suitable*".

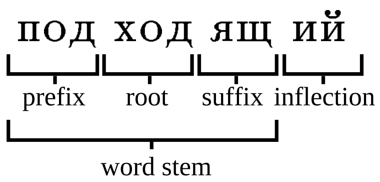


Figure 1.1 *Constituents of the Russian adjective meaning "suitable".*

Naturally, many English words also have a clear morphological structure although it is safe to say that it is much less productive than in Russian. In general, it is limited to the plural of nouns and third-person of verbs which are often formed by appending an "*s*" to the word stem. The prefixing of the morpheme "*un*" to word stems, while common, could not be said to be very productive. English compensates for having less grammatical information encoded within a word, by imposing strict constraints on the relative order of words in a sentence. In the sentence, "*The boy kicks the ball.*", it is only clear who is doing what to whom from the order in which the words are written. In Russian, on the other hand, the subject and object of the sentence can only be determined by each word's inflection and by agreement with the verb, not from the order of the words themselves. In fact, the above sentence translated into Russian, could be expressed with six[2] different permutations of the words without loss of meaning:

boy kicks ball
ball kicks boy
kicks boy ball
kicks ball boy
boy ball kicks
ball boy kicks

However, it is the case that some word orderings are preferred stylistically to others, and different emphasis is placed on words depending on their position in the sentence. In (Yokoyama, 1985), it is hypothesised that Russian word ordering is sensitive to: the fundamental ordering of words (if it exists at all), discourse factors (e.g. anaphora), the lexical and semantic complexity of words, and intonation.

A final remark on punctuation is necessary. Most speech recognition systems do not include punctuation in their output unless it is spoken explicitly (referred to as verbalised punctuation). However, in Russian, the presence or absence of commas could well be a case of life or death, as in the following example, attributed to Nicholas II, of a decree on the petition of a criminal sentenced to death: "*kaznit'(,) nelzya(,) pomilovat'!*[3]". The condemned man is sentenced to death if the comma appears after the first word, and pardoned if it is placed after the second word. The intention of a speaker may be made clearer from pauses and intonation in speech but is ambiguous in the absence of punctuation when written. Nonetheless, in the experiments contained in this dissertation, all punctuation, except for sentence boundary information, is removed from texts and is not used for modelling purposes.

1.4 Scope of the dissertation

All the models used in the experimental work in this dissertation centre around capturing language dependencies which occur in N-tuples of some particular modelling unit. Commonly, the modelling unit is the *word* and dependencies between words are modelled by N-tuples of words

[2]There is no definite or indefinite article per se in Russian.

[3]"казнить(,) нелзя(,) помиловать!" translates as: "*Execute(,) impossible(,) reprieve!*" (Ginzburg, 1989).

as in the ubiquitous word N-gram model. The work in this dissertation also considers dependencies between N-tuples of word *classes* and N-tuples of *sub-word* units which are referred to as N-tuples of *particles* in this dissertation. In language modelling, data sparsity is a recurrent theme—there is never enough data with which to estimate the parameters of a model sufficiently well. The aim of using different modelling units is to reduce the effects of data sparsity while preserving the usefulness of units in capturing language dependencies.

1.4.1 Modelling N-tuples of word classes

The use of classes of words to capture language dependencies is an example of combining units which have a similar functional use so as to make the probability estimate of an event more robust. There are generally fewer word equivalence classes than vocabulary words and so the probability estimate of the class-based events should be better estimated. Importantly, word N-tuples that did not occur in the training data, but that do appear in new data, may be well modelled by a class N-tuple. The class model is then said to be able to *generalise* better to previously unseen sequences. The ability to generalise is obtained with a sacrifice in the model's ability to predict a particular event precisely i.e. there is a corresponding loss of specificity. Chapters 4 and 5 consider class-based language modelling in detail and investigate two different forms of class models. In particular, algorithms for generating the word classes automatically are examined.

1.4.2 Modelling N-tuples of particles

The highly productive morphology of the Russian language suggests the use of modelling Russian using units which are constituents of the words themselves. In this dissertation, such units are referred to as particles. The contention here is that modelling dependencies in Russian at the word level is not optimal since much of the information about a word's role in a sentence is locked up within the word itself and not determined by its position relative to other words. By decomposing words into smaller units, the aim is to unlock some of this information and to use the units to capture more informative language dependencies. Chapter 6 considers several methods for decomposing words automatically into particles. Probabilities for word events are then obtained using particles as the units for capturing dependencies in the language. Although this technique is motivated by characteristics of the Russian language, the particle modelling approach is applied to both Russian and English in this dissertation.

1.5 Organisation of the dissertation

This introduction has outlined how the language model fits in with the other components of a speech recogniser and has explained the motivation behind investigating the language modelling of Russian and English. Chapter 2 presents a detailed introduction to statistical language modelling with special emphasis on the techniques relevant to the experimental work in the dissertation. An investigation of the salient characteristics of the Russian and English language corpora that are used for the modelling experiments in this dissertation is made in Chapter 3. Techniques for capturing language dependencies using word classes are examined in Chapters 4 and 5, and modelling of language using sub-word, particle units is then considered in Chapter 6. Conclusions and projected further work are presented in Chapter 7.

2

Statistical language modelling

The need for a language model in large vocabulary speech recognisers has already been outlined, together with the relationship of the language model to the other components of a contemporary speech recognition system. In this chapter, a comprehensive introduction to the theory of statistical language modelling is given. The first section describes the metric used to assess language model performance and the second section gives an overview of the characteristics and effectiveness of several language models that have been employed in speech recognition systems. The third and fourth sections look at the problems encountered in robustly estimating model parameters when there is insufficient training data, and examine several techniques that are used to overcome these problems. In the final section several frameworks are described for combining different language models.

2.1 Language model evaluation

The defining test of language model performance is to incorporate it in a speech recognition system and to perform a full recognition experiment. Only during recognition will the interaction of the language model with the acoustic model become apparent. Although the performance of the language model in the speech recogniser is ultimately crucial, it is generally impractical to evaluate language models in this manner due to the large amount of computation required. Moreover, the complexity of the recognition task often does not provide a sufficiently objective measure of language model performance. Some means is required, therefore, of evaluating language models as quickly and as objectively as possible. This should be done in isolation from the acoustics, in such a way that it will correspond to the performance of the language model when it is used in the speech recogniser. An information theoretic measure of language model performance which satisfies these requirements will be discussed next.

2.1.1 Entropy and perplexity

The probability that a language model assigns to some test text, which did not form part of the data used to train the model, gives an indication of how well that model predicts or models unseen data. The higher the average probability that a language model assigns to some test text, the better that model can be said to be at predicting the text. Ideally, the language model would capture the underlying probability distributions between the dependencies in a language, if indeed such distributions can be said to exist. In practice, the use of different model formulations which capture different language dependencies, and the use of different data to train a model will generate different probability estimates for the text.

From the point of view of information theory, the language model can be treated as an information source. This information source produces sequences of tokens $w_1, \ldots, w_n$ with probability $P(w_1, \ldots, w_n)$. When the source outputs a token it removes the uncertainty about the identity of the token. For this discussion, the source is a language model which imparts information about the language for use in the recognition search. The greater the uncertainty about the next token, the greater the information that the language model is said to have.

The tokens that are produced may, for example, be words, word sequences, collections of words (word classes) or even parts of words (sub-word units). Since the tokens are most often words, this discussion will refer to a source that only produces words. Each sequence of words $w_1, \ldots, w_n$ can be thought of as the realisation of a random process W, where each w_i is chosen from a vocabulary V of size N_V. The per-word entropy H of this source is defined as the average expected value of the logarithm[1] of the probability that the realisation of W is $w_1, \ldots, w_n$,

$$\begin{aligned} H &= -\lim_{n\to\infty} \frac{1}{n} E\{\log_2 P(w_1, \ldots, w_n)\} &(2.1)\\ &= -\lim_{n\to\infty} \frac{1}{n} \sum_{w_1,\ldots,w_n} P(w_1, \ldots, w_n) \log_2 P(w_1, \ldots, w_n), &(2.2) \end{aligned}$$

The summation is over all possible sequences of words, however, if the source is ergodic[2] then the entropy of the source is equivalent to

$$H = -\lim_{n\to\infty} \frac{1}{n} \log_2 P(w_1, \ldots, w_n), \tag{2.3}$$

and the entropy can be computed using an infinitely long sequence of words that is generated by the source. Obviously, the computation cannot be performed for an infinitely long sequence of words so n will be finite but should be sufficiently long. The estimated per-word entropy is then

$$\hat{H} = -\frac{1}{n} \log_2 P(w_1, \ldots, w_n). \tag{2.4}$$

The per-word entropy of the language model is a measure of the average difficulty or uncertainty that the speech recogniser experiences in determining a word from the same source. The

[1] The binary logarithm is used here for consistency with the following definitions and entropy is therefore measured in *bits*.

[2] A source is said to be *ergodic* if its statistical characteristics can be determined over a sufficiently long sequence *temporally*, instead of from an *ensemble* of sequences.

underlying structure of a language and its statistics are never usually known, hence estimates of the probabilities of the word sequences $\hat{P}(w_1, \ldots, w_n)$ are used instead. The estimated entropy $\hat{H}$ must then be greater than or equal to the entropy H,

$$H \leq \hat{H}, \tag{2.5}$$

because the probability estimates of the sequences can never be better than the actual probabilities of the word sequences. Another interpretation (Jelinek, 1997) is that the entropy of the source is a measure of the misestimation of the underlying probabilities $P(\cdot)$, by the language model probabilities $\hat{P}(\cdot)$

$$H \leq -\frac{1}{n} \sum_{w_1,\ldots,w_n} P(w_1, \ldots, w_n) \log_2 \hat{P}(w_1, \ldots, w_n). \tag{2.6}$$

The perplexity (Bahl et al., 1983) PP of a language model, which is directly related to the entropy, is defined as

$$PP = 2^{\hat{H}} = \hat{P}(w_1, \ldots, w_n)^{-\frac{1}{n}}, \tag{2.7}$$

which can be thought of as the average branching factor of the language model—the equivalent average number of equally probable words that follow any given word. The difficulty of the recognition task can now be interpreted as that of a source which chooses words independently of each other and with equal probability from a vocabulary of size PP.

It should be noted that neither entropy nor perplexity take into account the acoustic difficulty of recognising a word. It is possible that a language model that differentiates between acoustically similar words (homophones) or word sequences, may result in a lower word error rate than a model that has a lower perplexity. Moreover, perplexity does not highlight variations across a corpus and it has been suggested that a more informative method would use a histogram over local probabilities (Ney et al., 1994). An in-depth analysis of the relationship between recognition accuracy and perplexity is given in (Clarkson, 1999).

2.2 Statistical language models

In the previous section it was shown that the language model is evaluated according to the probability that it assigns to the sequence of words, $P(w_1, \ldots, w_n)$. There are many different ways in which the probability of a sequence of words can be generated, however it should be clear that it is impractical to generate, let alone store, the probabilities for all possible sequences of any number of words.

The chain rule of probabilities can be used to decompose the joint probability of the word sequence into a product of conditional probabilities

$$P(w_1, \ldots, w_n) = P(w_1) \cdot \prod_{i=2}^{n} P(w_i \mid w_1, \ldots, w_{i-1}). \tag{2.8}$$

The probability of each successive word is conditioned on all the preceding words, which will be referred to as the *word history*. The sequential manner in which words are predicted is particularly appropriate to the way in which the search for the most probable word sequence is performed in the speech recogniser. However, the number of parameters which are required to be estimated in such a model is still excessive. Some means of reducing the number of parameters is necessary through an appropriate choice of the type of language "events" which are to be modelled.

It must be made clear that the purpose of the language model in a speech recogniser is to apportion probabilities among all possible future events given a history of hypothesised words. It is not the aim of the language model to extract meaning from word sequences or uncover the underlying structure of language and inevitably it is only the "surface" probability distributions that may be modelled. Despite it seeming that the language models presented in this section are divorced from a rigorous linguistic interpretation, grammatical concepts have often guided their design.

Before examining how the types of conditioning events may be selected, the concept of Markov chains and its relationship to language modelling will first be introduced.

2.2.1 Markov chains

Markov chains[3] and hidden Markov models (HMMs) have already been mentioned in Section 1.2.2 in relation to the acoustic model of a speech recogniser. They also provide a useful mathematical tool for interpreting the action of language models and as such, will be referred to in several places in this dissertation. Only the notation and relevance of the techniques to language modelling will be considered here. For a detailed description of the mathematical properties of Markov models, parameter estimation techniques and search algorithms, the reader is referred to (Rabiner, 1989; Jelinek, 1990).

Many language models can be fully described using a Markov chain interpretation of their operation. The Markov chain can be thought of as a finite state process in which each state of the model corresponds to an observable event. The transition from one state to the next in a first-order Markov chain is only dependent on the previous state and not on any of the states prior to the previous state. The transition between states $s \rightarrow s'$ is specified by the state transition function $p(s' \mid s)$, where each state s corresponds to an observable event. The observations can therefore be related directly to the state sequence.

The hidden Markov model is an extension of the Markov chain in which the observation is a probabilistic function of the state. The underlying stochastic process determining the state

[3]It is somewhat appropriate that A.A.Markov, the Russian who developed the concept, used it to model sequences of vowels and consonants in Pushkin's poem "Eugene Onegin" (Jelinek, 1990).

sequence is hidden from the observer and can only be observed through another set of stochastic processes that produce the observation sequence (Rabiner, 1989). Hidden Markov models are useful for describing language models in which more than one state may be active at any time.

In a language model the observable outputs are usually words chosen from some vocabulary. A state in the Markov chain interpretation of language models typically corresponds to a word history or a mapping of the word history.

2.2.2 History equivalence classes

The difficulty of obtaining probability estimates for every conceivable word history in Equation (2.8) has already been mentioned. One solution is to classify word histories into equivalence classes. Given a word history $h_i = (w_1, \ldots, w_{i-1})$ for word w_i at position i, the task of language modelling is described in (Jelinek, 1997) as one of finding the best history classification function $S(w_1, \ldots, w_{i-1})$

$$S : w_1, \ldots, w_{i-1} \rightarrow s_i = S(w_1, \ldots, w_{i-1}). \tag{2.9}$$

The classification of a word history should be sufficiently refined so as to be a useful predictor of the next word, and the classes s should occur frequently enough to allow them to be reliably estimated.

There are many ways in which the history classification function S can be defined, the most obvious of which is a simple truncation of the word history to the last $(N-1)$ words. This corresponds to the very effective and widely used N-gram model. The definition of equivalence here is that all histories which end in the same $(N-1)$ words are identical from a language modelling point of view. Thus, two word histories $x_1, \ldots, x_m$ and $y_1, \ldots, y_n$ are equivalent in a word N-gram model if $x_{m-N+1}, \ldots, x_m = y_{n-N+1}, \ldots, y_n$. This idea can be applied to N-grams of word classes in which each word in the history is mapped to a word class. Then, the definition of equivalence is that all histories which end in the same $(N-1)$ classes are equivalent.

A more general interpretation of Equation (2.9) involves the mapping of whole word histories into equivalence classes which will be examined further in Section 2.2.6.1. The history classification function would then be chosen so as to maximise the number of words in the conditioning word history while ensuring that the classification is productive enough to produce reliable probability estimates. Such a model may be interpreted as the tying of word histories and pooling of their combined statistics. This model, however, has the drawback of needing to store a potentially large history classification function to map word sequences into history equivalence classes.

2.2.3 Word-based N-gram models

As discussed above, if the history classification function maps an entire word history into the last $(N-1)$ words, the model is referred to as a word-based N-gram language model (shortened to word N-gram model). Truncating the word history in this way reduces the number of parameters so they can be more reliably estimated, while still preserving the usefulness for predicting the current word. However, even for the trigram model ($N = 3$) the number of free parameters in the model is enormous—with a vocabulary of 65,000 words, there are potentially 2.7×10^{14} parameters to estimate. There will never be enough data to estimate all these parameters even if the resulting model could ever be stored or the probability estimates retrieved. Methods for estimating these parameters reliably are considered in Section 2.3.

The trigram model may lead to unintuitive equivalence classes and may not disambiguate sequences which are grammatically incorrect. The following two word histories

TWO TALL YOUNG STUDENT WENT YESTERDAY TO ...
TWO TALL YOUNG STUDENT WENT TO ...

are considered entirely different by the word trigram model, even though from the point of view of predicting the next word, the two histories are very similar. In addition, each triplet of words in the above sentences is entirely plausible and each trigram estimate is probably quite high. Hence the partial sentences, which are obviously both ungrammatical, are unlikely to be assigned appropriately small probabilities by the trigram model.

Despite these obvious drawbacks, bigram and trigram language models are still the most effective and widely used language models in contemporary speech recognition systems. Their effectiveness lies in the way in which the model parameters can be efficiently estimated and stored, and the simplicity with which they can be incorporated into the search process. The range of the N-gram model is, by definition, restricted; only local linguistic constraints are captured by the model. However, these relationships are generally powerful enough to guide the recognition search better than many more complicated models.

2.2.4 Class-based N-gram models

A class-based N-gram language model uses N-tuples of word equivalence classes to capture language dependencies in a text[4]. Since words are grouped into a number of classes which is smaller than the size of the vocabulary, there are fewer free parameters to estimate. Class-based models tend to be more compact than word models and the parameters more robustly estimated. The ability to generalise to unseen word sequences in the training data is generally better, however this is obtained with a tradeoff in the class model's ability to predict words precisely.

A deterministic word equivalence class mapping C (hereafter referred to simply as a deterministic class mapping) can be defined as follows:

$$C : w \to C(w). \tag{2.10}$$

[4]An equivalence class of all unknown words is often used in word N-gram modelling: words which have no vocabulary entry are mapped to an unknown word symbol. The unknown word class is therefore the complement of all the words that are in the vocabulary, and is used to model the collective statistics of all unknown words.

Given the above class mapping, several different class-based N-gram models can be constructed for computing $P(w_i \mid w_1, \dots, w_{i-1})$, for example

$$P_0(w_i \mid C(w_i)) \cdot P_1(C(w_i) \mid C(w_{i-N+1}), \dots, C(w_{i-1})), \tag{2.11}$$

$$P_0(w_i \mid C(w_i)) \cdot P_1(C(w_i) \mid w_{i-N+1}, \dots, w_{i-1}), \tag{2.12}$$

$$P(w_i \mid C(w_{i-N+1}), \dots, C(w_{i-1})). \tag{2.13}$$

A state mapping (history equivalence class) similar to that given by Equation (2.9) may also be incorporated into the class modelling scheme

$$P_0(w_i \mid C(w_i)) \cdot P_1(C(w_i) \mid S(w_{i-N+1}, \dots, w_{i-1})). \tag{2.14}$$

A probabilistic class mapping maps words into a number of classes each with a certain probability. Such mappings can take into account the fact that some words have multiple parts of speech, for example the word *light* can have up to four different parts of speech: *noun, verb, adjective* or *adverb*. Such a model results in the generation of multiple histories due to all the different realisations which result from the multiple class mappings of a word. In accordance with Bayes' formula, when a non-deterministic class mapping function is used, the prediction of the current word w_i requires a summation over the probabilities of all the possible realisations of the word history producing w_i (Ney et al., 1994)

$$P(w_i \mid w_1, \dots, w_{i-1}) = \sum_c P_0(w_i \mid c)\{\sum_s P_1(c \mid s) P_2(s \mid w_{i-N+1}, \dots, w_{i-1})\}. \tag{2.15}$$

It should be noted that Equation (2.15) reduces to Equation (2.14) in the case of deterministic class and state mappings.

Several methods have been used successfully in the literature for determining the class mapping function. These methods can be divided into two types: linguistic and data-driven. An obvious linguistic approach assigns words to classes according to the word's grammatical part of speech (POS) (Ney et al., 1994; Niesler and Woodland, 1996a). Such a model is only able to capture syntactic relationships between words since there is no semantic information encoded in parts-of-speech definitions. Another variation used in (Placeway et al., 1993) assigns words with similar semantic properties to the same class, for example, names of months form one class and names of ships another. The data-driven approach has been used successfully in (Kneser and Ney, 1993) where words are assigned to classes using a greedy algorithm that clusters words so as to minimise the training set perplexity. In contrast to the POS classification, both semantic and syntactic relationships are captured by these classes. A variation on this theme, presented in (Jardino and Adda, 1994), uses simulated annealing to minimise the training set perplexity. These approaches will be considered in more detail in Chapter 4.

Class models are often combined with word N-gram models using linear interpolation. Such combinations exploit the class model's ability to generalise, with the more specific predictive characteristics of the word model. Non-linear combinations of word and class N-gram models have been proposed in (Niesler and Woodland, 1996b) and (Blasig, 1999) by incorporating the class model into the backing-off scheme which will be described in Section 2.3.2.1.

A class model based on word classifications obtained using latent semantic analysis is presented in (Bellegarda et al., 1996). Relationships between the occurrence of words in individual documents are first obtained and then clustered according to some similarity criterion. As a consequence, the dependencies in this model are mainly semantic in nature and longer in range than the above class N-gram models.

2.2.5 Morphological models

The application of morphology to the statistical language modelling of French is presented in (El-Beze and Derouault, 1990). The motivation behind incorporating morphology in the language models for highly-inflected languages like French is to compensate for the increased data sparsity arising from the large vocabulary sizes that are necessary—word N-gram models tend to be less well trained if the vocabulary is very large.

The model in (El-Beze and Derouault, 1990) combines a class trigram component that uses a probabilistic class mapping with a trigram of word lemmas[5] component. The classes include 30 classes for content words (nouns, adjectives etc.) combining a mixture of syntactic and semantic definitions, and 72 classes for function words (pronouns, prepositions, conjunctions etc.). The model is described by the following:

$$P(w_i \mid w_1, \ldots, w_{i-1}) = \sum_{c_i} P(c_i \mid c_{i-2}, c_{i-1})\{\lambda_{c_i} G_i + (1-\lambda_{c_i}) M_i\} \tag{2.16}$$

with

$$\begin{aligned} G_i &= P(w_i \mid c_i) \\ M_i &= \sum_{L_i} P(L_i \mid L_{i-k}, L_{i-j}) P(w_i \mid c_i, L_i), \end{aligned} \tag{2.17}$$

where c_i is a possible POS of w_i and L_i is a possible lemma of w_i. In addition, (c_{i-2}, c_{i-1}) are the POSs of the last two observed words, whereas (L_{i-k}, L_{i-j}) are the lemmas of the last two *content* words that were encountered, which may not necessarily have been the last two observed words. The interpolation coefficients λ_{c_i} are dependent on c_i and are optimised for POSs of content words while $\lambda_{c_i} = 1$ for POSs of function words. The summations in the above formula account for words belonging to multiple POS classes and possessing multiple lemmas. The model is a variation on the conventional word N-gram model in which the events are instead defined as N-grams of classes and lemmas so the model parameters can be estimated in an identical manner to word N-gram models.

Variations on the above model, including different model combinations, are presented in (Cerf-Dannon and El-Beze, 1991) where recognition results are also given. A treatment of the

[5]A *lemma* refers to the *root-form* of a word e.g. the infinitival form of a verb.

combination of a word N-gram and the morphological model is given in (Maltese and Mancini, 1992).

2.2.6 Higher-order N-gram models

Two models are presented here which involve an alternative approach to defining the history classification function. The first method automatically determines the classification of N-gram contexts and the second method uses a binary tree growing procedure to classify contexts by asking questions about attributes of the word history.

2.2.6.1 Context clustering

In (Ueberla, 1995) the clustering of word histories (contexts) is considered. In the terminology of Markov models this is equivalent to a state tying operation in which the states are $(N-1)$-tuples of words; Ueberla refers to this model as having context-equivalent states. The clustered histories can be employed in the following model:

$$P(w_i \mid w_1, \ldots, w_{i-1}) = P_0(w_i \mid C(w_i)) \cdot P_1(C(w_i) \mid S(w_{i-N}, \ldots, w_{i-1})). \tag{2.18}$$

This model is referred to as two-sided (non-symmetric) (Ney et al., 1994): a state classification is used as the conditioning event for predicting the current word's classification, from which the current word is then predicted. It is non-symmetric because the state and word classification functions are different.

The clustering algorithm that Ueberla uses is very similar to the greedy algorithm mentioned in Section 2.2.4 for word clustering and which will be described in more detail in Chapter 4. The two main differences are that Ueberla's algorithm simultaneously clusters words into word classes and contexts into context-equivalent classes. An heuristic to improve the clustering speed is also applied. The set of words to be clustered is the vocabulary V and the set of contexts V^n where n is the number of words in the context.

The algorithm proceeds by moving each word in turn (each context in turn) to a set of candidate word classes (context classes) and leaves the word (context) in the class (context class) for which the increase in likelihood of the training data is the greatest. The heuristic speed improvement that is applied computes a list of K candidate classes into which a word w is most likely to be clustered. This means that the best class might not be among the candidate classes, however, the results show that the increase in clustering speed far outweighs the small loss of accuracy. The heuristic speedup for a unigram context is as follows: for each context class $s \in S$, a list of the K context classes s_i which most frequently co-occur with s is maintained i.e. the K highest entries in $N(s, s_i)$ (a symmetric situation applies for C). For a word w which is being moved, a list of the K classes which most frequently co-occur with w is constructed in a similar manner. The number of classes which appear in both these lists gives a score indicating how similar the distributions of w and s are. The candidate classes are chosen to be the K highest scoring classes. Certain other heuristic improvements are also made, for example the K co-occurring classes are only updated after a certain number of words have been moved.

Experimental results are reported in (Ueberla, 1995) for bigram and trigram clustering, i.e. single words and bigrams are clustered using S. Only the most frequent 500,000 bigrams are clustered into 7000 context classes for the trigram model, and the 20k vocabulary is clustered into 1000 classes. The clustered bigram model is shown to consistently outperform a back-off word bigram model when there are less than two million words of training data available. The clustered trigram model only outperforms the clustered bigram model when trained on the largest, forty million word training set. In (Ueberla and Gransden, 1996) results are reported for a variation of the context clustering algorithm which initially moves groups of contexts simultaneously and progressively refines the groups that are clustered until individual contexts are moved. The groups of contexts share common sub-contexts and are moved, as before, into the class for which the increase in training set likelihood is greatest.

2.2.6.2 Tree-based models

In (Bahl et al., 1989), an intuitively attractive approach is presented to the clustering of word histories using binary decision trees. At each node in the tree, a *yes/no* question is posed about a particular word in the history. This results in one of two branches successively being taken until a leaf node is reached. When a leaf node is reached, the context has been classified. Each leaf node is associated with a probability distribution with which to compute the probability of a word in that context.

It is readily accepted that the definition of N-gram equivalence classes is naive since histories which end in the same $(N-1)$ words are deemed equivalent. Word $(N-1)$-gram contexts which differ, may in fact be functionally identical from a language modelling point of view as was highlighted in Section 2.2.3. This results in unnecessary data fragmentation and less reliable estimation of probabilities. Using a binary decision tree to classify word histories is an attempt to circumvent these problems by making a more effective classification of the word history and by incorporating more information about the history into that classification.

Tree construction begins at the root node and requires a "good" question to be determined about some aspect of the history. When the question has been chosen, the data is split into two parts according to whether the answer is yes or no for the particular piece of data. These two branches result in two more nodes for which questions have to be determined, and so the tree growing process continues until some stopping criterion is met. Any form of question is allowed about any of the words in the history, consequently the search space is potentially huge. Questions may, for example, take the form of: *"Is the last (second to last) word such-and-such?"*, in which case an N-gram model could be built in the tree framework given enough questions about the identity of the preceding $(N-1)$ words. Trees could also be constructed by very patient linguisticians, however a data driven, greedy approach is instead employed at each node to find the optimal question from a set of questions.

Determining the optimal question at a particular node does not take subsequent nodes into consideration i.e. question selection is greedy and only locally optimal. Bahl et al. describe a method for allowing the use of composite binary questions using a pylon of questions (successive *yes/no* questions-and-answers). Allowing this results in reduced data fragmentation and also permits questions of the form: *"is the previous word an adjective and this one an adverb?"*. However, the additional computation involved significantly increases the complexity and construction time of the trees. Data sparsity still remains an issue at the leaves of the tree, and the distributions must be smoothed with lower-order distributions for which the parameters are chosen by maximising the likelihood of some held-out data.

Bahl et al. report results for a tree that was constructed to predict the 21st word from a history of 20 words. The tree, which was incompletely grown, performed better than a word trigram model. An even greater reduction in perplexity was obtained by interpolating the tree model with the trigram model. This indicates that the different history classifications in the tree model capture different dependencies between words which complement those of the word trigram model. It is also pointed out that, if sufficient numbers of parallel processors are available, tree-growing need not take an excessive amount of time since the optimal question to be determined for each node is independent of the data and questions at other nodes.

2.2.7 Word-pair association models

The essence of modelling word-pair relationships is the incorporation of longer-term dependencies than are provided for in the conventional N-gram model. It was noted above that N-gram models for English tend to capture only *local* syntactic and semantic constraints. In addition, there is no dependence on the location of an event in the corpus for an N-gram model, since the probability distribution of words for a given context is constant irrespective of where that context occurs in the corpus. Models of word-pair associations extend the range over which dependencies can be captured. The dependencies tend to be semantic in nature and allow changes in topic or style to be tracked dynamically and for the probability distributions to be adjusted accordingly. A simple M-gram word-pair model can be described by

$$P(w_i \mid w_{i-M+1}, \dots, w_{i-1}) = \sum_{m=1}^{M-1} \lambda_m \alpha(w_i \mid w_{i-m}), \tag{2.19}$$

where $\sum_{m=1}^{M-1} \lambda_m = 1$ and $\sum_w \alpha(w \mid v) = 1$ and $\alpha(w \mid v) \geq 0 \quad \forall v$. This model[6] forms the basis of the word triggers and cache models presented next.

2.2.7.1 Word trigger models

The *word triggers* approach to language modelling develops associations between pairs of words in which a *trigger* word boosts the probability of a set of *target* words with which it is associated (Lau et al., 1993). The weights λ_m in Equation (2.19) may be chosen to make the contribution of each predecessor word dependent on the distance from w_i.

The method used to select useful pairs is of great importance since the search space of possibilities is huge (a large proportion of N_V^2). The average mutual information measure has been

[6] A special case of the above model in which only the (w_{i-2}, w_i) dependency is used has been referred to as a *distance-2 bigram* model in the literature.

used successfully in (Rosenfeld, 1994) and (GuoDong and KimTeng, 1999) to select trigger pairs $(A_0 \rightarrow B)$ according to the expected benefit A_0 provides in predicting B,

$$AMI(A_0 : B) = P(A_0, B) \log \frac{P(A_0, B)}{P(A_0)P(B)} + P(A_0, \overline{B}) \log \frac{P(A_0, \overline{B})}{P(A_0)P(\overline{B})} + P(\overline{A}_0, B) \log \frac{P(\overline{A}_0, B)}{P(\overline{A}_0)P(B)} + P(\overline{A}_0, \overline{B}) \log \frac{P(\overline{A}_0, \overline{B})}{P(\overline{A}_0)P(\overline{B})}. \quad (2.20)$$

Self-triggers, in which a word triggers itself, have been shown to be particularly powerful and robust, while triggers with the same root have also proved useful. In particular, strong associations were found between a noun and its possessive form and the singular and plural forms of a noun (Rosenfeld, 1994).

Class triggers have also been investigated in (Rosenfeld, 1994). Classes are obtained using a linguistic tool to map words to their lemma in an attempt to combine the effects of same-root triggers into one class. In (Niesler and Woodland, 1997) a class N-gram model is used and the class membership probability of a target word is adapted according to the previous occurrence of its associated word and class triggers in the text.

The above model form is not the only way in which trigger-target dependencies can be incorporated. Trigger models have also been built using the maximum entropy approach (see Section 2.5.4) to combine both word trigger and word N-gram constraints into one model.

2.2.7.2 Cache-based models

The *self-triggers* mentioned above share several similarities with the *cache* model first developed in (Kuhn and De Mori, 1990) where the name was coined from its use in computer science. Cache language models boost the future probabilities of events (commonly unigrams, bigrams etc.) according to how many times they have been previously observed in the history. It has been shown that there is a tendency for some words to be repeated locally in a text more often than would be predicted by a conventional N-gram model. It is this phenomenon that is captured by the cache model. Cache models may also be considered as a method of model adaptation since the probabilities of words are adjusted in favour of words that have already appeared in the current discourse.

In experiments conducted in (Kuhn and De Mori, 1990; Kuhn and De Mori, 1992) 200-word caches were maintained for 19 part-of-speech classes and used to adjust the class membership probability component in the multiple-class membership model of Equation (2.15)

$$P(w_i \mid w_1, \dots, w_{i-1}) = \sum_c P_0(w_i \mid c) \cdot P_1(c \mid C(w_{i-2}), C(w_{i-1})) \quad (2.21)$$

where

$$P_0(w_i \mid c) = \lambda_c P_{cache}(w_i \mid c) + (1 - \lambda_c) P_{class}(w_i \mid c). \quad (2.22)$$

The cache model probabilities can be computed directly from the occurrence frequency of words in the K words maintained by the cache for class c

$$P_{cache}(w_i \mid c) = \frac{N(w, c)}{K}. \tag{2.23}$$

There is no summation over possible realisations of classes of words in the history since the model relies on assigning the most probable word class to each word. The word is then inserted into the appropriate cache for that class. The above model can easily be combined with a conventional word N-gram model using linear interpolation: the interpolation weights λ_c are chosen for each class so as to optimise some held-out text and are not themselves adjusted dynamically during recognition.

Cache models have been shown to significantly reduce the perplexity on a text especially where the general language model is itself poorly trained. However, the improvement in perplexity is not consistently reproduced in terms of recognition results (Woodland et al., 1998). This may be explained by the fact that a word will only increase the probability of itself occurring again in the future if it was correctly recognised in the first place by the general model. If it was correctly recognised, then it is likely that it will be correctly recognised again in the future and thus the cache is unlikely to help.

Since the cache is generally restricted to only a few hundred words there is very little data with which to estimate N-gram probabilities for $N > 1$ so unigram caches are common although higher-order N-grams have also been evaluated in (Jelinek et al., 1991). A cache model in which the effect of the predecessor word decreases exponentially with the distance from the current word is treated in (Clarkson and Robinson, 1997).

2.2.8 Mixture-based models

Mixture-based models combine several component language models, each of which is specific to a particular style or topic. The simplest method of combining M models $\mathcal{M}_j$ is to interpolate them linearly

$$P(w_i \mid w_1, \dots, w_{i-1}) = \sum_{j=1}^{M} \lambda_j P_{\mathcal{M}_j}(w_i \mid w_1, \dots, w_{i-1}), \tag{2.24}$$

and to adjust the λ_j dynamically, according to which component language models correspond best to the current discourse.

The M components can be determined in a data-driven manner using a clustering algorithm to classify texts which share a similar style or topic. In some cases a corpus may already have topic tags defined. Texts may also be permitted to belong to more than one mixture component which can reduce the effects of data fragmentation. One of the M models is usually a conventional word N-gram which is built using all the available data; word N-gram models can then be built individually for the other $(M - 1)$ topic components.

2.2.9 Probabilistic context-free grammars

A context free grammar assigns one or more syntactic structures to a sentence. The grammar comprises a set of rewrite rules which map a non-terminal symbol onto one or more terminal or non-terminal symbols. Non-terminal symbols usually correspond to grammatical definitions such as parts of speech and terminal symbols are often the words of a language. The context-free aspect of these grammars means that long-range dependencies (within-sentence) can be captured and so they are not constrained to the immediately preceding context, as is the case for N-gram models. An example of a structure applied to the sentence *"the boy kicked the ball"* is given in Figure 2.1.

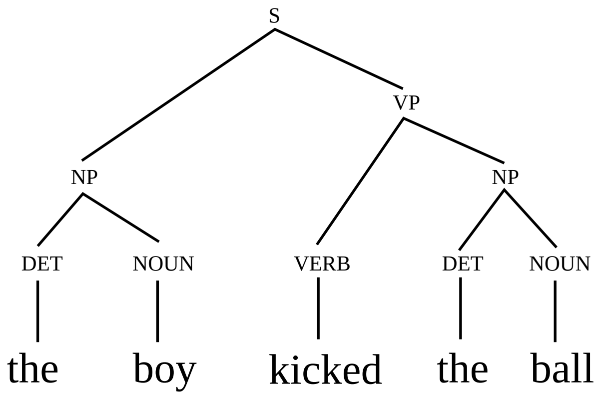


Figure 2.1 *Syntactic structure of the sentence "the boy kicked the ball".*

The non-terminal symbols are: sentence (S), noun-phrase (NP), verb-phrase (VP), determiner (DET), noun and verb. The terminals are "*the*", "*boy*" "*kicked*" and "*ball*".

A probabilistic context-free grammar (PCFG) assigns probabilities to each of the rewrite rules and thus a probability can be generated for each parse of a sentence. Summing over all possible parses of a sentence gives a probability for the sentence. In this way a PCFG can be employed as a language model. In cases where a sentence has multiple parses (referred to as syntactic ambiguity) the parse with the highest probability can be chosen. This may also be useful for extracting meaning from a sentence.

Probabilities for each rewrite rule can be estimated from a training text so as to optimise the training text likelihood. Training algorithms exist for estimating the rule probabilities, however they tend to be computationally very demanding (Baker, 1979). The application of PCFGs to language modelling has in general been restricted to small corpora and small vocabulary sizes (Wright et al., 1992; Lloyd-Thomas et al., 1995). The incorporation of PCFGs into the recognition process represents a further problem and most attempts have concentrated on n-best rescoring of the recogniser output. A comprehensive treatment of the estimation algorithms and PCFGs is given in (Lari and Young, 1990; Charniak, 1993; Stolcke, 1995).

The *link grammar* developed in (Sleator and Temperley, 1991) is a highly lexical context-free grammar which produces a parse of a sentence by forming direct links between pairs of words in the sentence. In addition, the link grammar has sufficient flexibility to encompass N-gram models by incorporating links between consecutive N-tuples of words. A probabilistic model of link grammar has also been developed for which efficient training and parsing algorithms also exist (Lafferty et al., 1992).

Another relatively recent contribution is the structured language model described in (Chelba and Jelinek, 1999a). The structured model develops syntactic-like structure in a sentence incrementally by generating the probability for a word using all possible parses of the partial sentence up to that word, given its part of speech and the exposed heads from the two previous parses. Improvements both in terms of perplexity and word error rate (up to 1% absolute) were obtained over conventional word trigram models on the Wall Street Journal and Switchboard tasks (Chelba and Jelinek, 1999a; Chelba and Jelinek, 1999b; Jelinek and Chelba, 1999).

2.3 Robust parameter estimation

Having described a broad range of language modelling dependencies in the previous section, the discussion will now turn to how the parameters which model these language events can be estimated robustly. An *event*, in language modelling terms, refers to any of the kinds of dependencies which are captured by a model e.g. for a word bigram model the events are all the possible word pairs. When conditional bigram probabilities are discussed, the events are the individual words that may follow a fixed predecessor word context.

Many of the possible events, for which a probability estimate is required, are not observed during training. This is particularly true for word N-gram models where the training data contains a very small fraction of the events for which probabilities must be determined. It is imperative that no event be assigned a zero probability since this would result in infinite perplexity and would mean that the event could never be predicted at recognition time.

Several techniques have been developed for tackling the problem of estimating probabilities for unseen events and these will be discussed in relation to word N-gram models. Firstly, the Good-Turing probability estimate is described followed by a range of discounting techniques which reduce the count of observed events by some factor and assign the leftover probability mass to unseen events. Secondly, two frameworks are described for combining the probability estimates of events from different distributions. The final part of this section considers a method which uses the available data both for model training and validation.

2.3.1 Probability estimation techniques

The maximum likelihood probability estimate for an event $\mathcal{E}$ that occurs r times in a sample of size R is the same as the relative frequency

$$P(\mathcal{E}) = \frac{r}{R}. \tag{2.25}$$

Events which do not appear in the sample are therefore assigned a zero probability by the maximum likelihood estimator. Observed events are biased high and unobserved events are biased low. To correct this bias and assign probability to unseen events, it is suggested in (Katz, 1987) that the count of observed events should be discounted by a multiplicative factor d_r, referred to as a *discount coefficient*

$$r^* = r \cdot d_r, \tag{2.26}$$

which results in a new probability estimate for an event that occurs r times

$$q_r = \frac{r^*}{R}. \tag{2.27}$$

The Good-Turing probability estimate was first proposed in (Good, 1953) and two alternative derivations of the estimate may be found in (Nadas, 1985). The basis of the estimate is the *symmetry requirement* which states that two events which occur the same number of times in a sample must have the same probability. Before beginning, it is therefore useful to define the concept of *count-counts* n_r for the number of events which occur r times in a sample. The sample size is then given by $R = \sum_r rn_r$. In particular, the number of zeroton events n_0 and the number of singleton events n_1 play an important role in many discounting schemes.

The Good-Turing estimate modifies the count of an event occurring r times to be

$$r^* = (r+1)\frac{n_{r+1}}{n_r}. \tag{2.28}$$

The estimate for the total probability of all unseen events is then

$$q_0 \cdot n_0 = \frac{n_1}{R}. \tag{2.29}$$

The number of singletons in a sample is thus used as an estimate of the number of unseen events. To satisfy the above estimate for the total probability of unseen events different discount coefficients may then be defined

$$q_0 \cdot n_0 = 1 - \frac{1}{R}\sum_{r>0} n_r \cdot r \cdot d_r. \tag{2.30}$$

2.3.1.1 Good-Turing discounting

The Good-Turing estimate, that was introduced above, can also be used to modify the count of an event that occurs r times, in which case the corresponding discount coefficient is

$$d_r = (r+1)\frac{n_{r+1}}{r \cdot n_r}. \tag{2.31}$$

However, if a data sample is particularly sparse, the Good-Turing estimate will break down for r in cases where $n_r = 0$. Since the counts can be expected to be more sparse for large r, Katz suggests that only events which occur k times or less should be discounted, where typically $k = 5$. Events that occur more than k times are assumed to be reliably estimated by their relative frequencies, hence the discount coefficient is defined as

$$d_r = \begin{cases} \frac{(r+1)\frac{n_{r+1}}{r \cdot n_r} - (k+1)\frac{n_{k+1}}{n_1}}{1-(k+1)\frac{n_{k+1}}{n_1}} & 1 \leq r \leq k \\ 1 & r > k \end{cases} \tag{2.32}$$

It is evident from Turing's estimate that the relative values of the count-counts must satisfy the relationship

$$n_1 \geq 2n_2 \geq 3n_3 \ldots \tag{2.33}$$

otherwise the resulting discounted counts will not be consistent with each other. In particular, d_r will be negative if $\frac{(k+1)n_{k+1}}{n_1} > 1$ and it is also necessary to ensure that $d_r > 0$ for all r. For most naturally occurring data these constraints are satisfied, however problems do occur for example in class N-gram models when small numbers of classes are used.

2.3.1.2 Linear discounting

The idea behind linear discounting is that event counts should be discounted in proportion to the count, hence d_r is constant for all r (Ney et al., 1995). If the Good-Turing estimate for unseen events is required to be satisfied the discount coefficient is set to

$$d_r = 1 - \frac{n_1}{R}. \tag{2.34}$$

This has also been shown to optimise the leaving-one-out probability which will be discussed in Section 2.3.3.

2.3.1.3 Absolute discounting

Absolute discounting subtracts a constant value b from each count, so its effect on higher counts is less than on lower counts. The discount coefficient is defined as

$$d_r = \frac{r-b}{r}. \tag{2.35}$$

To satisfy the Good-Turing estimate for unseen event probabilities, the value of b is set to

$$b = \frac{n_1}{M}, \tag{2.36}$$

where $M = \sum_r n_r$, the number of different events in the sample.

It has been shown in (Ney et al., 1994) that in order to maximise the leaving-one-out likelihood of a text, an upper bound on the optimal value for b is

$$b = \frac{n_1}{n_1 + 2n_2}. \tag{2.37}$$

2.3.1.4 Witten-Bell discounting

The Witten-Bell discounting scheme was developed in (Witten and Bell, 1991) and first applied to language modelling in (Placeway et al., 1993). The three discounting methods described above subtract probability mass from the observed events in a particular context whereas the Witten-Bell scheme adds a small factor to the conditioning context to account for unseen words in that context. The conditional probability of a word w in a particular context h is thus defined as

$$\hat{P}(w \mid h) = \frac{N(h,w)}{N(h)+t}, \tag{2.38}$$

where t is the number of distinct events that occur in context h, i.e. the number of distinct words that follow h in the corpus. It follows that the probability mass of previously unseen words in the context h is given by:

$$\hat{P}(0 \mid h) = \frac{t}{N(h)+t} \tag{2.39}$$

which is distributed among the unseen events according to a lower-order distribution in a similar manner to that used in the other discounting schemes.

A discount coefficient may also be defined for consistency with the other methods, in which the dependency is now on the number of distinct events t in a particular context

$$d_r(t) = \frac{N(h)}{N(h)+t}, \tag{2.40}$$

The technique has been shown to perform as well as the Good-Turing method and is particularly robust to abnormalities in training data.

2.3.2 Frameworks for probability smoothing

All the discounting schemes that have been described effectively reduce the counts of observed events in the sample and implicitly assign small non-integer counts to unobserved events. The frameworks presented now for smoothing event probabilities affect the conditional probabilities of events. The first scheme relies on the implementation of one of the above discounting schemes while the second scheme does not require event counts to be discounted and instead combines the relative frequency estimates of different events.

2.3.2.1 Backing-off

Katz introduced the *backing-off* scheme in conjunction with Good-Turing discounting in (Katz, 1987). In general, backing-off refers to using a probability estimate proportional to one from a more general distribution when the estimate from the specific distribution is unreliable or perhaps non-existent. Backing-off results in a decision being made about which distribution to use and which not to use.

In the case of an N-gram model, if the estimate for the N-gram is deemed unreliable, the more general $(N-1)$-gram distribution can be referred to. For example, the trigram offers a greater refinement in predictive power over a bigram or unigram and the trigram distribution would be used in cases where the trigram events have occurred frequently enough for their estimates to be reliable. If this estimate is unreliable, the bigram distribution is used instead (*backed-off to*) and if this is also unreliable then the unigram distribution will be used. Usually, an estimate is deemed unreliable if it is not stored in the language model i.e. if the event count is zero or if the event has been discarded (see Section 2.4).

The backing-off scheme will now be illustrated using the word trigram as an example. Given an estimate for the probability of observed events $\hat{P}$ the total probability of unseen events $\hat{P}_u$ occurring in the context (w_{i-2}, w_{i-1}) is given by

$$\hat{P}_u(w_{i-2}, w_{i-1}) = 1 - \sum_{w:N(w_{i-2},w_{i-1},w)>0} \hat{P}(w \mid w_{i-2}, w_{i-1}). \tag{2.41}$$

The backing-off scheme distributes this probability mass among unobserved events according to the more general distribution $\hat{P}(w_i \mid w_{i-1})$

$$\hat{P}(w_i \mid w_{i-2}, w_{i-1}) = \frac{\hat{P}_u(w_{i-2}, w_{i-1})}{\sum\limits_{w:N(w_{i-2},w_{i-1},w)=0} \hat{P}(w \mid w_{i-1})} \cdot \hat{P}(w_i \mid w_{i-1}), \tag{2.42}$$

where the denominator ensures that the probabilities sum to one. For a scheme in which events occurring more than k times are not discounted the three cases for generating the trigram probability estimate are given below:

$$\hat{P}(w_i \mid w_{i-2}, w_{i-1}) = \begin{cases} \frac{N(w_{i-2},w_{i-1},w_i)}{N(w_{i-2},w_{i-1})} & N(w_{i-2}, w_{i-1}, w_i) > k \\ d_{N(w_{i-2},w_{i-1},w_i)} \cdot \frac{N(w_{i-2},w_{i-1},w_i)}{N(w_{i-2},w_{i-1})} & 1 \le N(w_{i-2}, w_{i-1}, w_i) \le k \\ \alpha(w_{i-2}, w_{i-1}) \cdot \hat{P}(w_i \mid w_{i-1}) & N(w_{i-2}, w_{i-1}, w_i) = 0 \end{cases} \tag{2.43}$$

where $\alpha(w_{i-2}, w_{i-1})$ is termed a *back-off weight* which is defined as

$$\alpha(w_{i-2}, w_{i-1}) = \frac{1 - \sum\limits_{w:N(w_{i-2},w_{i-1},w)>0} \hat{P}(w \mid w_{i-2}, w_{i-1})}{1 - \sum\limits_{w:N(w_{i-2},w_{i-1},w)>0} \hat{P}(w \mid w_{i-1})}, \tag{2.44}$$

and ensures that the constraint $\sum_w \hat{P}(w \mid w_{i-2}, w_{i-1}) = 1$ is satisfied.

In Equation (2.44), the numerator expresses the left-over probability mass obtained from discounting the counts of observed events and the denominator is a normalising factor that expresses the total back-off probability. Smoothing is performed recursively with the unigram probabilities being smoothed first since these are required for bigram smoothing and so on.

The advantages of the backing-off method lie in its computational efficiency since all the back-off weights can be precomputed, and in the ease with which it can be incorporated into the decoding process.

2.3.2.2 Deleted interpolation

Another method for smoothing the relative frequency estimate of a conditioned event is to combine the maximum likelihood estimates of different types of events linearly. In contrast to backing-off where a decision is made about which distribution to use, this method combines all the available distributions. For example, the trigram relative frequency estimates may be smoothed by interpolating trigram, bigram and unigram relative frequency estimates

$$P(w_i \mid w_{i-2}, w_{i-1}) = \lambda_3 \frac{N(w_{i-2}, w_{i-1}, w_i)}{N(w_{i-2}, w_{i-1})} + \lambda_2 \frac{N(w_{i-1}, w_i)}{N(w_{i-1})} + \lambda_1 \frac{N(w_i)}{N} \tag{2.45}$$

where λ_j are non-negative weights which satisfy $\lambda_1 + \lambda_2 + \lambda_3 = 1$. The λ_j should be chosen so as to maximise the probability of events in some held-out data (cross-validation set) which was not used to obtain the event counts $N(\cdot)$[7].

The method of deleted interpolation makes use of a hidden Markov model interpretation of language models, to describe the above combination of estimates (Jelinek and Mercer, 1980). The λ_j $j \in 1, 2, \ldots, N$ represent the probabilities of a transition into one of N states, each of which is associated with a 1-gram, 2-gram, ..., or N-gram output distribution for estimating the probability of w_i. The λ_j are defined globally which is inevitably a sub-optimal arrangement, and are not dependent on the conditioning events since the search space would be too large. The optimal λ_j can be estimated using the forward-backward algorithm (Baum, 1972) or with the help of calculus (Jelinek, 1997; Bahl et al., 1991).

2.3.3 Cross-validation

A common problem in any modelling task is the over-fitting of model parameters to the training data. Data that was not used to train the model parameters can be used to validate the model and minimise over-fitting. Cross validation for language modelling involves, in its simplest form, the training of model parameters on some *retained* text and the optimisation of these parameters on some generally smaller *held-out* text. Optimisation may take the form of making the probability estimates more robust or perhaps making the model more compact without a loss of accuracy.

Event counts are obtained from the retained part and the optimal held-out probability estimate $\hat{\mu}_r$ of *all* events which occur exactly r times in the retained part is

$$\hat{\mu}_r = \frac{Y_r}{\sum\limits_r Y_r} \tag{2.46}$$

where Y_r is the number of occurrences in the held-out data of all events that appear exactly r times in the retained part.

A rotation form of cross-validation partitions the text into H disjoint parts and successively rotates the held-out portion of the data. The deleted estimate is then defined as

[7] If the same data is used for computing λ_j the trivial solution of $\lambda_1 = \lambda_2 = 0$ and $\lambda_3 = 1$ is obtained.

$$\mu_r = \frac{\sum\limits_{h=1}^{H} Y_r^h}{\sum\limits_{h=1}^{H} \sum\limits_{k=1}^{K} Y_k^h}, \tag{2.47}$$

where Y_r^h $(h = 1, \ldots, H)$ is Y_r when the hth subset is the held-out (deleted) part.

The *leaving-one-out* method of cross-validation is a special case of deleted estimation in which the text is partitioned into $H = N_W$ parts, where N_W is the size of the corpus and each held-out sample therefore constitutes only one word. Leaving-one-out makes maximum use of the available data since all samples are used as both training and test data. The Good-Turing probability estimate is also obtained as a result within the leaving-one-out framework (Nadas, 1985).

2.4 Methods of language model pruning

The discussion so far has concentrated on methods for obtaining reliable probability estimates of all the possible events which the language model captures. In particular, estimates are needed for those events which did not occur during training. Probabilities for these events are estimated using only the statistics for events which have been observed. Although the statistics for all events that occurred might possibly be stored in the language model, it is often impractical with large training sets to store the statistics for each event explicitly due to the size of the resulting language model and the time required to search for specific events. This section will consider three methods, which are motivated by different goals, for removing the explicit statistics of observed events from a language model to produce more compact models.

2.4.1 Cutoffs

The simplest and most widely used method for making language models more compact removes all those events which occur the same number of times or less than a specified cutoff value. This method is also referred to as absolute frequency thresholding. For example, in the case of a word trigram model, word bigrams and trigrams which occur only once are often discarded. This generally results in a significant reduction in the size of the language model and only a small degradation in the performance of the model. Singleton events often account for the largest part of observed events in language modelling and even if the explicit statistics of events are removed, the statistics of singleton events as a whole are still required by many of the discounting schemes described earlier.

In general, when the explicit statistics of events are removed from a model, the performance of the model worsens. However, depending on the way in which a corpus is partitioned into training and test data, removing all singleton events has been shown to improve model performance (see Section 3.3.4). This is particularly true if the data is "noisy" (e.g. if a corpus has been inadequately preprocessed) or is not sufficiently homogeneous between training and test sets. Noisy events should ideally be infrequent since they contribute no useful information at best, and at worst, only inaccurate information.

2.4.2 Variable length N-gram models

The idea behind the variable length N-gram model is to successively extend an existing context if it is shown that doing so will improve the performance of the language model. In (Niesler and Woodland, 1996a), the leaving-one-out log-likelihood of the model on the training data is used as the performance criterion,

$$LL_{loo} = \sum_{i=1}^{n} \log P(w_i \mid h) \tag{2.48}$$

where $P(w_i \mid h)$ is the leaving-one-out probability estimate which is calculated by removing one instance of the event (h, w_i) from the training data. Alternatively, this can be represented as the sum of the likelihood contributions from all the events in each context in the model,

$$LL_{loo} = \sum_{n=1}^{N_H} \sum_{i=1}^{N_V} N(h_n, w_i) \cdot \log P(w_i \mid h_n) \tag{2.49}$$

where N_H is the number of unique contexts stored in the language model, $N(h_n, w_i)$ is the number of times event w_i occurs in context h_n and $P(w_i \mid h_n)$ is the leaving-one-out probability of the event w_i.

If the extension of a context results in more than a specified increase in the leaving-one-out likelihood, then the context is extended; if not, that particular context is not extended and cannot subsequently be extended any further. Whenever a context is extended there is a change in the overall likelihood of the data. This is due to the additional contribution from the extended context $LL_{h_n^+}$ and the changed contribution of the original unextended context now that its context has been extended LL'_{h_n}. Since the original contribution to the likelihood was that of the unextended context LL_{h_n}, the change in likelihood ΔLL_{loo} is simply,

$$\Delta LL_{loo} = LL'_{h_n} + LL_{h_n^+} - LL_{h_n} \tag{2.50}$$

and an acceptance criterion with threshold parameter λ_{ct} can thus be defined as

$$\Delta LL_{loo} > -\lambda_{ct} \cdot LL_{loo}. \tag{2.51}$$

Storing the contribution of each extended context that is retained in the model removes the need to recompute its contribution if it is considered subsequently for further extension. In addition, the contribution of the original context when it has been extended can be computed efficiently by adjusting only the contributions of those events which occur in both the original context and the extended context.

2.4.3 Entropy-based pruning

The operation of the entropy-based pruning of language models (Stolcke, 1998) is similar to a method described in (Seymore and Rosenfeld, 1996) and since the former method was shown to give slightly better results it will be described here. Explicit probability estimates are removed from the language model if it is shown that doing so results in an improvement of the language model perplexity or a degradation that is deemed acceptable. For each context, every N-gram event stored in an $(N-1)$-gram context has its explicit probability estimate tentatively replaced by the implicit (backed-off) $(N-1)$-gram estimate,

$$P'(w \mid h) = \alpha'(h)P(w \mid h') \tag{2.52}$$

where h' is the last $(N-2)$ words in h.

The pruning algorithm aims to minimise the divergence between the original distribution $P(\cdot \mid \cdot)$ and the pruned distribution $P'(\cdot \mid \cdot)$. Assuming that each N-gram has an independent effect on the divergence, the relative entropy can be used to quantify this change

$$D(P \mid\mid P') = -\sum_{w_i} P(w_i, h)[\log P'(w_i \mid h) - \log P(w_i \mid h)]. \tag{2.53}$$

The removal of an explicit N-gram event (h, w) changes the back-off weight for that context and therefore affects the contribution from all backed-off estimates,

$$\begin{aligned} D(P \mid\mid P') = -P(h)\{&P(w \mid h)[\log P'(w \mid h) - \log P(w \mid h)] + \\ &\sum_{\forall w_i : N(h,w_i)=0} P(w_i \mid h)[\log P'(w_i \mid h) - \log P(w_i \mid h)]\}. \end{aligned} \tag{2.54}$$

Insertion of the backed-off estimates into the above equation removes the need for a summation over all vocabulary words and allows the relative entropy to be computed efficiently,

$$\begin{aligned} D(P \mid\mid P') = -P(h)\{&P(w \mid h)[\log P(w \mid h') + \log \alpha'(h) - \log P(w \mid h)] + \\ &[\log \alpha(h') - \log \alpha(h)] \sum_{\forall w_i : N(h,w_i)=0} P(w_i \mid h)\}. \end{aligned} \tag{2.55}$$

The summation in the above equation is simply the probability mass of unseen events given by Equation (2.41). The marginal history probabilities $P(h)$ can be obtained by multiplying together the appropriate conditional probabilities $P(w_{i-N+1}) \cdot P(w_{i-N+2} \mid w_{i-N+1}) \cdots$ and the updated back-off weight $\alpha'(h)$ is obtained by omitting the contribution from the pruned N-gram in the numerator and denominator of Equation (2.44).

Since relative entropy is directly related to the intrinsic perplexity of the language model $PP = e^{-\sum_{h,w} P(h,w) \log P(w|h)}$ the change in perplexity between the original and the pruned model is given by $e^{D(P||P')-1}$. Consequently a selection criterion can be defined so that explicit N-gram estimates are retained which, if they were to be removed, would increase the perplexity by more than some threshold value.

2.5 Methods of model combination

Many of the language models discussed above capture different language dependencies and the question often arises of how these different information sources about the language may best be combined. Four methods are considered here, the first two of which, linear interpolation and backing-off, have already been mentioned in the context of probability smoothing. The other two methods, log-linear interpolation and maximum entropy, have recently gained in popularity with the availability of cheaper and more powerful computer processors.

2.5.1 Linear interpolation

Linear interpolation was described earlier in conjunction with smoothing probability distributions with more general distributions. The method can be used to combine probability distributions from M component models as was discussed for mixture-models and the deleted interpolation method

$$P(w_i \mid w_1, \dots, w_{i-1}) = \sum_{j=1}^{M} \lambda_j P_{\mathcal{M}_j}(w_i \mid w_1, \dots, w_{i-1}) \tag{2.56}$$

where $\sum_{j=1}^{M} \lambda_j = 1$. In general, the weights λ_j are defined globally which restricts the number of additional parameters that must be estimated. The weights can easily be determined using an expectation-maximisation (E-M) algorithm (Dempster et al., 1977) to optimise the perplexity on some held-out data.

The interpolated model is guaranteed to be no worse (in terms of perplexity) than any of its component models for the data on which the weights are optimised. Component models which are not useful are assigned a relatively small weight. Linear interpolation is a simple and often effective method of model combination but tends not to make the best use of the strengths of individual models. The globally weighted average of all the models means that the contribution of each model is fixed, even though in a particular situation one model may be far more robust at predicting a specific event than another model is. For example, in a trigram model if the trigram has been observed many times during training it could be expected to be a good predictor and its weight should be correspondingly high. However, if the trigram was only seen once or twice, more emphasis should be given to the bigram model if that were to be better estimated.

The interpolation weights might ideally be made dependent on the context, however, the extra number of parameters to be estimated would be a serious drawback. One alternative is to form separate *bins* of weights which vary the contribution of different models depending on how reliable the component probability estimates are. Contexts may be classified according to how well estimated their component distributions are, and optimal interpolation weights can then be determined for each class of contexts.

2.5.2 Backing-off

Backing-off involves the selection of one distribution in preference to another. The method is frequently employed in N-gram models, (see section 2.3.2.1) where by default the N-gram distribution is questioned first and if this is not well estimated, the $(N-1)$-gram distribution is referred to and so on. More generally, information sources are ranked in order of the detail or refinement of prediction they provide about an event. The most detailed information source is questioned first and if the source provides insufficient information about the event the next model in line is questioned. Ultimately, a specific choice is made about which distribution to use from those available.

Backing-off has been applied successfully to combine models which capture different dependencies, for example, in (Niesler and Woodland, 1996b) and (Blasig, 1999) a class N-gram model is combined with a word N-gram model: if the word N-gram estimate is unreliable, a back-off is performed to the class model and the class N-gram estimate used instead. Back-off models are generally simple to generate and are compact.

2.5.3 Log-linear interpolation

Log-linear interpolation (Klakow, 1998) has been shown to be a more effective way of combining models than linear interpolation while still maintaining the same number of parameters that must be estimated. M component models are combined in the following manner:

$$P(w \mid h) = \frac{1}{Z(h)} \prod_{i=1}^{M} P_i(w \mid h)^{\lambda_i}, \tag{2.57}$$

where $Z(h)$ is a normalising constant that ensures $\sum_w P(w \mid h) = 1$ for each h. There are no explicit constraints on the interpolation parameters λ_i which can be optimised on some held-out data using an algorithm for multidimensional optimisation. Both the generalised iterative scaling algorithm (Darroch and Ratcliff, 1972) and the downhill simplex method (Press et al., 1992) have been used to obtain the λ_i.

Log-linear interpolation has been applied to domain adaptation and combining bigram and distance bigram models in (Klakow, 1998) where it was shown to give significant improvements both in terms of perplexity and word error rate compared to models combined by linear interpolation.

2.5.4 Maximum entropy

Maximum entropy models have been applied successfully to language modelling in (Rosenfeld, 1994; Berger et al., 1996). The maximum entropy approach to language modelling combines language dependencies in a fundamentally different way to the previously described methods: instead of combining the models themselves, features from different models or information sources are combined into one model. Each dependency (h, w) is associated with a feature $f(h, w)$ which may be a real-valued or binary-valued function of (h, w). For example, if (h, w)

were a word bigram and the single-word context h was followed by w in the training data then $f(h,w) = 1$, and 0 otherwise. Different features are required to agree on the average with a certain statistic of the training data. The expectation of each $f(h,w)$ under the desired distribution $P(h,w)$ (the expected value of the feature in the model) is constrained to be equal to the desired expectation K (generally its expected value in the training data) using the constraint:

$$\sum_{h,w}[P(h,w)f(h,w)] = K. \tag{2.58}$$

Hence, for a set of constraint functions $\{f_i(h,w)\}_{i=1,2,...}$ with corresponding desired expected values $\{K\}_{i=1,2,...}$, the aim is to determine the optimal probability distribution $P(h,w)$ or, more often, the conditional probability distribution $P(w \mid h)$. Applying the maximum entropy principle involves choosing the probability distribution with the highest entropy from among all possible probability distributions which satisfy the constraints. If the constraints are consistent[8] (which should be the case if they are all obtained from the same data) a unique solution is guaranteed to exist and be of the following form (Berger et al., 1996)

$$P(w \mid h) = \frac{1}{Z(h)} \exp(\sum_i \lambda_i f_i(h,w)), \tag{2.59}$$

where the λ_i are unknown constants and $Z(h) = \sum_w \exp(\sum_i \lambda_i f_i(h,w))$ ensures that $P(w \mid h)$ is a probability distribution. To search for the λ_i that make $P(w \mid h)$ satisfy all the constraints, the generalised iterative scaling algorithm (Darroch and Ratcliff, 1972) is used, and is guaranteed to converge to the solution.

The maximum entropy approach to language modelling is attractive in that many disparate knowledge sources can all be combined effectively into one model. However, the main reason why maximum entropy models have not become widespread is the computational burden of the search for the maximum entropy distribution that satisfies the modelling constraints.

2.6 Summary

This chapter has outlined the field of statistical language modelling with particular reference to its relationship to automatic speech recognition. The perplexity measure was defined as a means of evaluating the performance of a language model and this was followed by an overview of the major types of statistical language models that are currently in use. A description was then given of several techniques that have been proposed for ensuring that the parameters of language models are reliably estimated. Finally, a range of techniques were discussed for pruning language models and combining different language models.

The remainder of this dissertation will concentrate on the development and evaluation of various techniques for modelling Russian and English.

[8] If constraints are derived from different data sources or a discounting scheme is used, the constraints may no longer be consistent.

3

The corpora and word-based language modelling

This chapter begins with a brief outline of the requirements of the language modelling tools that were needed to model the language dependencies used in the experimental work. The origin and preprocessing of the Russian and English corpora are then described and as a first step in modelling these languages, conventional word-based techniques are applied to the two corpora. The failings of these techniques are analysed in some detail and provide the motivation for the new techniques that are proposed in subsequent chapters.

The original contributions contained in this chapter include the collation and preprocessing of a Russian text corpus for language modelling and an analysis of its characteristics. In particular, word N-gram models are built for various vocabulary sizes of up to one million words for the Russian corpus and a back-off word N-gram model using permutated word histories is also described.

3.1 Language modelling tools

A collection of language modelling software tools was developed along the lines of the CMU-Cambridge Language Modelling Toolkit (Clarkson and Rosenfeld, 1997). It was foreseen that certain basic requirements were required of the tools for the experiments in this dissertation which were not available in the CMU-Cambridge Toolkit. Consequently, these tools needed to be significantly extended to provide the required functionality. The most important difference was allowing vocabulary sizes greater than sixty-five thousand words—this will be seen to be essential for experiments on the Russian corpus. Another major difference was to permit words in the history to be mapped into different classes according to their position i in the history,

$$P(w_i \mid w_{i-N+1}, ..., w_{i-1}) = P(w_i \mid C_{N-1}(w_{i-N+1}), ..., C_1(w_{i-1})). \tag{3.1}$$

This particular facility of the tools is used in Chapter 5 for one-sided class-based language models in which the class mapping is made to depend on the position of the word in the history. It should be noted that a conventional word N-gram model is a particular case of the class model where each word is mapped to its own unique class and where this same mapping is used for all $(N-1)$ words in the history.

Essentially, only counts and back-off weights are stored in the model and the probability for each requested N-gram is generated on-the-fly. Where the number of parameters in a model is referred to in this dissertation, this is taken to be the total number of N-grams stored i.e. for a trigram model it is the total number of unigrams, bigrams and trigrams stored in the model. It should be noted that the actual physical storage requirements for the above model formulation are greater than when words in the history are not mapped into classes. For the latter case, one tree structure would be sufficient. Instead, each set of N-gram counts must be stored in separate count tables since the context for each N-gram is now inconsistent across N-gram tables. However, the same definition for the number of parameters will be used since this is valid and consistent with other models in terms of the number of parameters required to generate a probability and the search effort required to retrieve each parameter.

A set of *context cues* may be specified during language model construction to define words that are themselves not to be predicted, but which are used in the context for predicting other words. In all perplexity calculations in this dissertation, the unknown word "<UNK>" and the sentence-start "<s>" symbols are specified as context cues and their probability of occurrence is never predicted. However, where these two symbols or the sentence-end symbol "</s>" occur, they are used as the context for predicting a following word i.e. no back-off is performed to exclude them from the context.

In addition, all three model pruning methods described in Section 2.4 were implemented and where possible the results were verified against those published in the literature. The accuracy of the newly developed language modelling tools was tested, using a similar, standard word trigram model and identical vocabulary, against the Entropic HTK language modelling tools (Young et al., 1997) and CMU-Cambridge Toolkit (Clarkson and Rosenfeld, 1997). The results in each case were shown to be identical.

3.2 Corpora selection and preprocessing

Two language modelling corpora, one for Russian and one for English, were required for the language modelling experiments in this dissertation. Both corpora needed to be sufficiently similar in terms of composition and size, so that their characteristics and the experimental results could be compared sensibly. Two similar corpora were eventually located, each of which contained around one hundred million tokens. Given the large size of the corpora it was expected that small amounts of "noisy" text would not manifest themselves in the experiments, however, certain precautions were nonetheless taken. The composition and characteristics of the two corpora are detailed below together with a description of the sanity checks that were made to ensure the reliability and integrity of the two corpora.

3.2.1 The Russian corpus

Currently, there are no commercially available Russian language text corpora. However, a large source of Russian text material was eventually located in Russia, and this source was used as the basis for all the Russian language modelling experiments contained in this dissertation. This corpus of Russian texts is very varied in content, ranging from classical literature and translations of popular foreign novels to lists of anecdotes and jokes. In its original form it was completely unusable since texts were stored using inconsistent character mappings[1] and also contained a large amount of text formatting characters which had to be removed prior to use. After the corpus had been cleaned and the character mappings normalised, sentence boundary information (sentence-start and sentence-end markers) was added which replaced various punctuation markers such as full-stops, ellipsis and exclamation marks. All other punctuation was removed. Finally, an important procedure in corpus preparation was also executed—the removal of repeated "chunks" of text from the corpus, including whole articles or excerpts from other texts where necessary. If such repetitions were to occur both in the training set and the test set then spurious results (most likely to be much better results) would be obtained. It is important therefore that none of the test data appears in the training data. An heuristic method of determining repetitions of fifty-word sequences was developed and repeated segments up to and including the nearest sentence-end marker were removed from the corpus. The final stage of corpus preprocessing mapped all numerical digits to a "<NUMBER>" symbol[2].

3.2.2 The British National Corpus

Since it was desired to conduct similar experiments on English to those on Russian, the choice of English language corpus, of which there were many, was dictated by the characteristics of the Russian corpus, over which there was little control. It was decided that the most similar, readily available English language corpus in terms of its composition and size, was the British National Corpus (BNC corpus) (Burnard, 1995). The BNC corpus is a collection of English language texts ranging from *belles lettres* and entire novels to transcriptions of spoken language. The words in the BNC corpus are tagged with various attributes such as parts of speech, and each article in the corpus also has header information pertaining to topic classifications and publication information. This extra information is surplus to the needs of the language modelling experiments in this dissertation and all of it was removed. As with the Russian corpus, there were large sections of repeated data in the BNC corpus. The same criterion of removing repeated 50-word sequences was used to produce a "cleaner" corpus. Finally, to maintain some consistency with the processing of the Russian corpus, numerical digits were all mapped to a "<NUMBER>" symbol, however the occurrence of these numerical digits was significantly lower than in the Russian corpus.

[1] Some letters occur in both the Cyrillic and Roman alphabets and there were many cases of Russian words that had been represented using a mixture of the two. Such inconsistencies needed to be removed otherwise the same words might be treated as completely different ones.

[2] There is no simple method for converting numerical digits into word representations in Russian, since each number changes its grammatical case, gender and number depending on its role in the sentence.

3.3 Corpora characteristics

Once the corpora had been cleaned and normalised, the next step was to partition the corpus into training and test sets. Two test sets were required: one development set (`dev-test`) with which to *optimise* the parameters of the model (which are estimated using the training set), and one evaluation set (`eval-test`) with which to evaluate the performance of the language model. The larger portion of the data is assigned to the training data set, since in general there is never enough data with which to estimate all the free parameters of a model and maximum use of the available data should be made wherever possible. Both corpora were finally partitioned in the ratio of 98:1:1 for `training:dev-test:eval-test` sets sizes.

The method in which a corpus is partitioned can significantly affect the experimental results. Two common methods result in what shall be referred to here as *homogeneous* and *heterogeneous* data partitioning. Homogeneous data sets can be created by selecting every nth sentence (or "chunk" of sentences) from the corpus to form the test sets, with the left-over data forming the training set. Heterogeneous data sets can be formed by selecting entire test sets on an article-by-article basis such that the number of words in each partition are in the desired ratios. For the experiments in this dissertation only the homogeneous data partitions for each corpus are used, however in this chapter comparative results will be given for the heterogeneous partition of the BNC corpus where this is instructive. The characteristics of the homogeneous partitions for both corpora are given in Tables 3.1 and 3.2.

	`training`	`dev-test`	`eval-test`
Total words	98,663,154	1,007,350	1,004,880
Total sentences	8,349,349	85,195	85,194

Table 3.1 *Homogeneous Russian corpus partitions.*

	`training`	`dev-test`	`eval-test`
Total words	112,492,860	1,142,137	1,142,867
Total sentences	6,064,508	61,880	61,885

Table 3.2 *Homogeneous English corpus partitions.*

It must be remembered from section 2.1.1 that for the perplexity calculation to be reliable, the length of the sequence used for testing should tend to infinity. This is, of course, impossible in practice but the corpus partition sizes were justified by the perplexity results on different sizes of test data of a baseline word trigram model built using the training data for each corpus. The perplexity on half the `eval-test` set of around half a million words was 7.4% lower for the Russian corpus (10.3% for the English corpus) than on the entire one million-word `eval-test` set. However, the perplexity on the combination of both the `eval-test` and `dev-test` sets of around two million words was only 0.1% higher for the Russian corpus (0.5% for the English corpus) than on the `eval-test` set alone. Data comprising one million words was therefore considered adequate for testing purposes while retaining the maximum amount of available data for training.

3.3.1 Vocabulary growth and corpus size

It is evident from a basic knowledge of Russian and English that the number of different words encountered in the two languages will differ significantly. What is unknown is to what extent this difference manifests itself and the consequences it will have on existing language modelling techniques. In Figure 3.1 the growth in the number of unique tokens (essentially the vocabulary size) is plotted against corpus size for the two corpora.

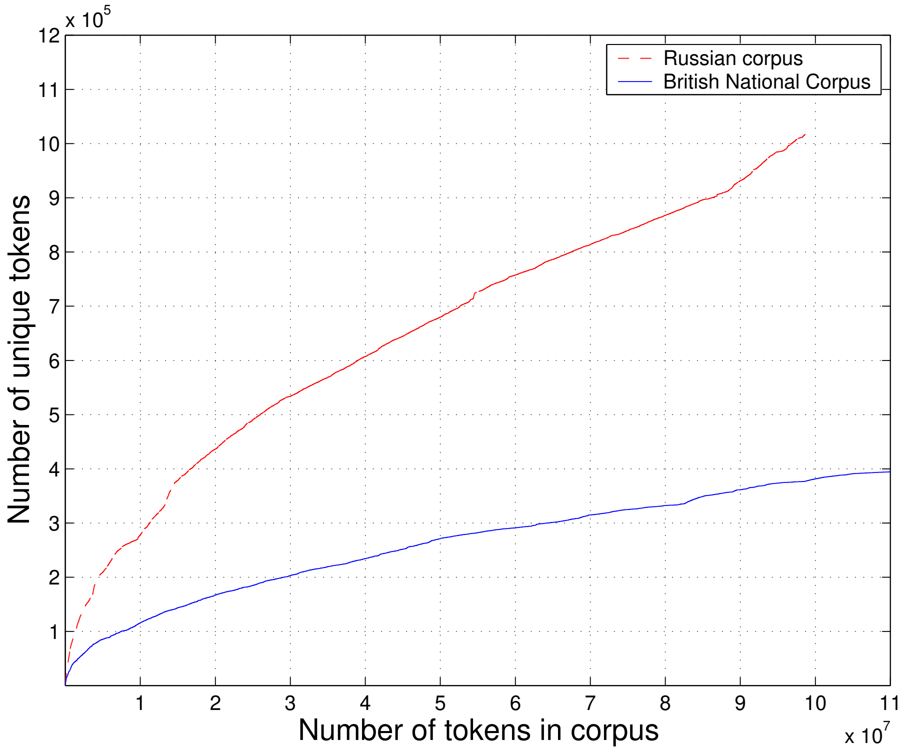


Figure 3.1 *Growth in vocabulary size against corpus size.*

It is clear from Figure 3.1 that the rate of growth of the vocabulary for Russian is approximately two and a half times greater than that for English. In fact, this is perhaps surprisingly low, since Russian words generally have many more inflected forms than words in English. This might be explained by the fact that not all inflected forms of a Russian word are used with the same frequency. The other interesting observation from this graph is that the vocabulary size is nowhere near saturating with the increasing corpus size.

3.3.2 Coverage and vocabulary size

A useful measure of the coverage of a vocabulary is the percentage of words that are encountered in some held-out text which are *out-of-vocabulary* (OOV). The OOV-rate is thus defined as the number of tokens which are not in the vocabulary divided by the number of tokens in the held-out text. The significance of a particular vocabulary's OOV-rate on the recognition performance of a speech recogniser cannot be understated. If a word is not in the vocabulary, it cannot be recognised. However, the wrong word that is hypothesised to have occurred in it place will affect the recognition of subsequent words since it will be used as the context for predicting the current word by the language model. It has been shown that, on average, for the Wall Street Journal Task, every OOV word that occurs in the test data, results in approximately 1.6 word-errors (Woodland et al., 1994). OOV words tend to be fairly rare, so even if they are subsequently included in the vocabulary, their acoustic and linguistic properties tend not to be well modelled. For each OOV word in the test data that is subsequently added to the vocabulary, on average approximately 1 word error (from the previous 1.6) is recovered during recognition.

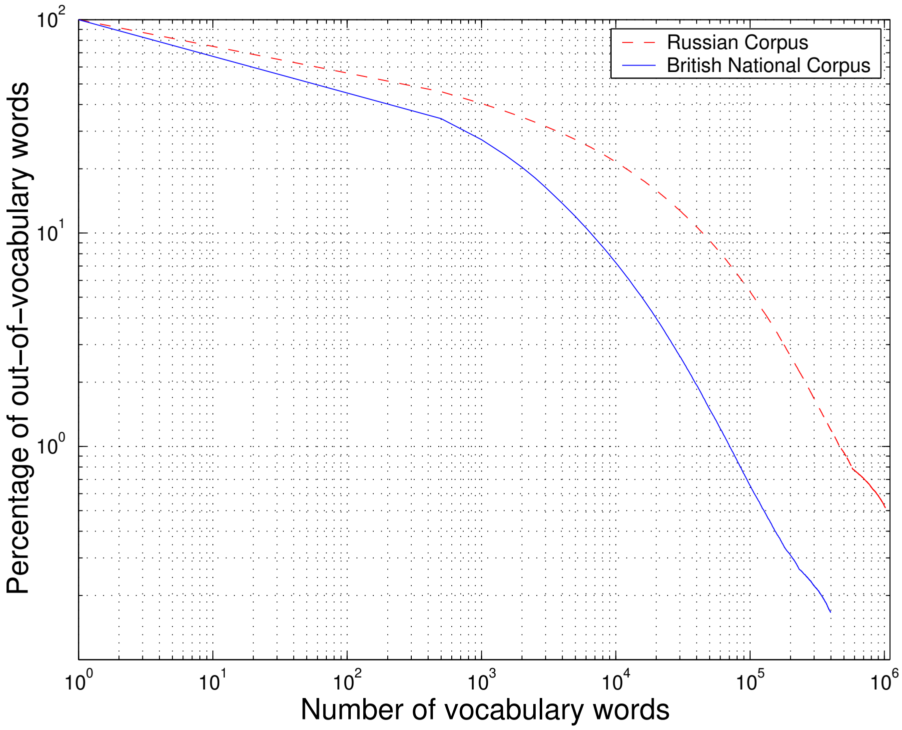


Figure 3.2 *Variation of OOV-rate against* (log) *vocabulary size.*

A vocabulary of size N_V is defined by taking the most frequent N_V words from the `training` set of each corpus. The OOV-rate is computed with respect to the `eval-test` set of each corpus. The results, displayed in Figure 3.2, highlight the significant difference between the two languages which the size of the vocabulary has on the OOV-rate.

Currently the vocabulary size of a *large vocabulary speech recogniser* is around 65,000 (65k) words since this is close to the limit of the number of identifiers that can be represented in a computer by a two-byte integer. For the English corpus, such a vocabulary provides almost 99% coverage on the `eval-test` set. The most significant observation with regard to the Russian corpus, and one that inevitably dictates the course of subsequent work, is that the coverage of the appreciably large (65k) vocabulary is only 92.4%. Alternatively, the OOV-rate of 7.6% is seven times greater than for the English corpus 65k vocabulary.

From Figure 3.2 it is clear that as the size of the vocabulary is increased, the coverage on the English corpus increases almost ten times faster than the coverage on the Russian corpus. However, for both corpora and for the size of vocabularies of interest (>65k), each doubling of the vocabulary size approximately halves the OOV-rate. Note that there are always OOV words in some held-out text, even when the vocabulary is defined to contain all the words which occur in some much larger partition of the corpus.

All subsequent experiments will be performed with a vocabulary of the most frequent 65k words that occur in the `training` set for both the Russian and English corpora. In addition, experiments for a vocabulary of the most frequent 430,000 (430k) words in the `training` set will be used for the Russian corpus. This vocabulary size was chosen to provide an OOV-rate identical to that on the English corpus with a 65k vocabulary.

3.3.3 Russian word N-gram models

Word trigram models employing Good-Turing discounting and Katz back-off were built for the Russian corpus with all singleton N-grams ($N > 1$) discarded. A range of vocabulary sizes were investigated to assess the variation in perplexity and coverage on the `eval-test` set and the results are shown in Table 3.3 and perplexity is plotted against vocabulary size in Figure 3.3.

Vocabulary size	Perplexity on `eval-test`	OOV rate (%)
65k	413.3	7.60
100k	481.0	5.31
200k	586.8	2.65
300k	639.8	1.67
400k	670.9	1.19
430k	677.0	1.10
500k	689.9	0.93
600k	702.5	0.77
700k	708.0	0.71
800k	713.8	0.64
900k	718.8	0.59
1M	724.3	0.53
1.01M(ALL)	724.9	0.52

Table 3.3 *Perplexities on Russian corpus `eval-test` data of back-off word trigram models.*

As is expected, the perplexity of the word models increases as the vocabulary size is increased. This is obvious from the interpretation of perplexity given in Section 2.1.1 where for a

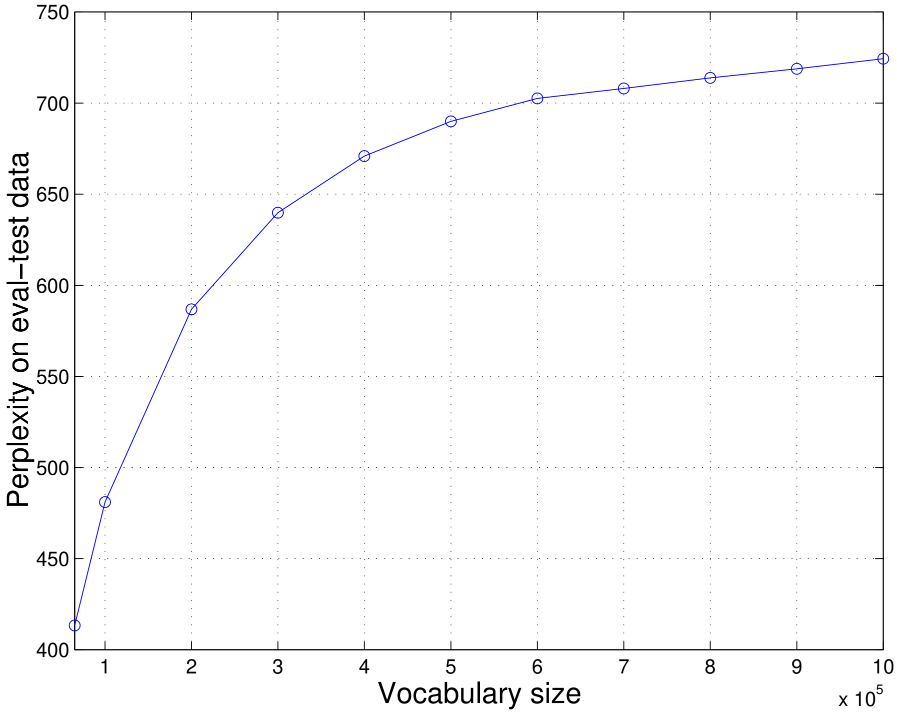


Figure 3.3 *Perplexity on Russian corpus* `eval-test` *data of word trigram models with various vocabulary sizes.*

fixed size training corpus, the greater the number of different words to be predicted, the greater the uncertainty with which words are predicted and hence the greater the perplexity. Since vocabulary words are chosen according to their frequency of occurrence in the corpus, increasing the vocabulary size means increasing the number of lower-frequency words in the vocabulary. The statistics of additional low-frequency words are unlikely to be as well estimated as words in a smaller vocabulary and so the perplexity for the larger vocabulary increases. It is interesting to observe that the increase in perplexity has nothing to do with an increase in data sparsity which might be expected when a larger vocabulary is used. For example, using a trigram word model built with the 430k Russian vocabulary, the equivalent perplexity of a 65k vocabulary is in fact 0.3% lower than the 65k word trigram model.

Several, more detailed perplexity experiments were conducted on the Russian `eval-test` data for the 65k and 430k vocabularies. The results are shown in Tables 3.4 and 3.5, and give the perplexity and N-gram hit-rates for trigram and 4-gram back-off word language models for the cases where singleton N-grams ($N > 1$) are discarded and where they are retained. The N-gram hit-rates show the percentage of N-gram requests that were found in the model and what percentage of requests required backed-off estimates. It was not possible to build the 4-gram language model for the 430k vocabulary while retaining all N-grams, due to computer memory limitations. The results show that retaining N-grams is useful in reducing the perplexity for both trigram and 4-gram models and for both vocabulary sizes. For the 65k trigram model the perplexity improvement is 6.3% while for the 430k trigram model it is 8.8%. Models in which all N-grams are retained have significantly more parameters (five to six times more) than models which discard singleton N-grams. However, these are significant improvements[3] and suggest that perhaps the Good-Turing discounting scheme in conjunction with backing-off is underestimating low-frequency events. The improvements when N-grams are retained correlate with an increase in trigram (4-gram) hits and further imply the usefulness of the singleton events in prediction. However, this is generally only true for training and test data that are homogeneous.

	Trigram model		4-gram model	
Cutoffs (bi,tri,4g)	1, 1	0, 0	1, 1, 1	0, 0, 0
Perplexity	413.3	387.4	398.9	385.5
4-gram hits (%)	—	—	28.2	35.1
3-gram hits (%)	54.0	61.4	25.8	26.2
2-gram hits (%)	32.0	28.4	32.0	28.4
1-gram hits (%)	14.0	10.3	14.0	10.3

Table 3.4 *Homogeneous Russian corpus partition: perplexities and hit-rates with 65k vocabulary and different cutoffs.*

	Trigram model		4-gram model	
Cutoffs (bi,tri,4g)	1, 1	0, 0	1, 1, 1	0, 0, 0
Perplexity	677.0	617.4	656.9	N/K
4-gram hits (%)	—	—	22.3	N/K
3-gram hits (%)	44.8	52.0	22.5	N/K
2-gram hits (%)	33.8	31.6	33.8	N/K
1-gram hits (%)	21.4	16.4	21.4	N/K

Table 3.5 *Homogeneous Russian corpus partition: perplexities and hit-rates with 430k vocabulary and different cutoffs.*

The 4-gram models outperform the trigram models by around 3% which is not a large improvement. This may be a consequence of the sparsity of the Russian corpus, the effect of which becomes more significant as N is increased. It may also be speculated that the free word-ordering in Russian means that there is little to be gained by increasing the context for making predictions, since sequences of longer length are more likely to appear in the future as hitherto unobserved permutations.

[3]It is worthwhile repeating that large perplexity improvements are sometimes obtained when all N-grams are retained, if the test data appears, even partially, in the training data. A procedure to remove repetitions in the corpus was performed as part of the corpus preparation, which suggests that the results reflect a characteristic of Russian rather than a peculiarity of the corpus.

3.3.4 English word N-gram models

An equivalent set of back-off word trigram language models with various vocabulary sizes was built for the homogeneous partition of the English corpus. All singleton bigrams and trigrams were discarded and the perplexity computed on the `eval-test` data. The results are shown in Table 3.6 and perplexity is plotted against vocabulary size in Figure 3.4.

For reasons similar to those already given, increasing the vocabulary size increases the perplexity due to the addition of more low-frequency words to make larger vocabularies. The increase observed with English is similar to that for Russian when examined over a comparable range of OOV-rates.

Vocabulary size	Perplexity on `eval-test`	OOV rate (%)
65k	216.1	1.10
100k	224.5	0.65
200k	232.4	0.31
300k	235.0	0.22
399k(ALL)	236.8	0.17

Table 3.6 *Perplexities on homogeneous English corpus partition* `eval-test` *data of back-off word trigram models.*

An interesting variation on the more detailed perplexity experiments is now considered for the English corpus. It was mentioned earlier that two different types of partitions (homogeneous and heterogeneous) had been constructed for the English corpus. The effect of cutoffs on back-off trigram and 4-gram language models will now be examined for the two corpus partitions using 65k vocabularies. Table 3.7 shows the perplexity and N-gram hit-rates of trigram and 4-gram word models on the heterogeneous corpus partition. The vocabulary is chosen to be the most frequent 65k words from the heterogeneous corpus training partition and hence differs from the vocabulary used in the homogeneous corpus experiments. The OOV-rate for the 65k vocabulary on the `eval-test` set is 1.5%.

	Trigram model		4-gram model	
Cutoffs (bi,tri)	1, 1	0, 0	1, 1, 1	0, 0, 0
Perplexity	240.9	236.8	225.4	229.0
4-gram hits (%)	—	—	30.7	37.7
3-gram hits (%)	60.3	67.1	29.6	29.4
2-gram hits (%)	30.2	25.9	30.2	25.9
1-gram hits (%)	9.5	7.0	9.5	7.0

Table 3.7 *Heterogeneous English corpus partition: perplexities and hit-rates with 65k vocabulary and different cutoffs.*

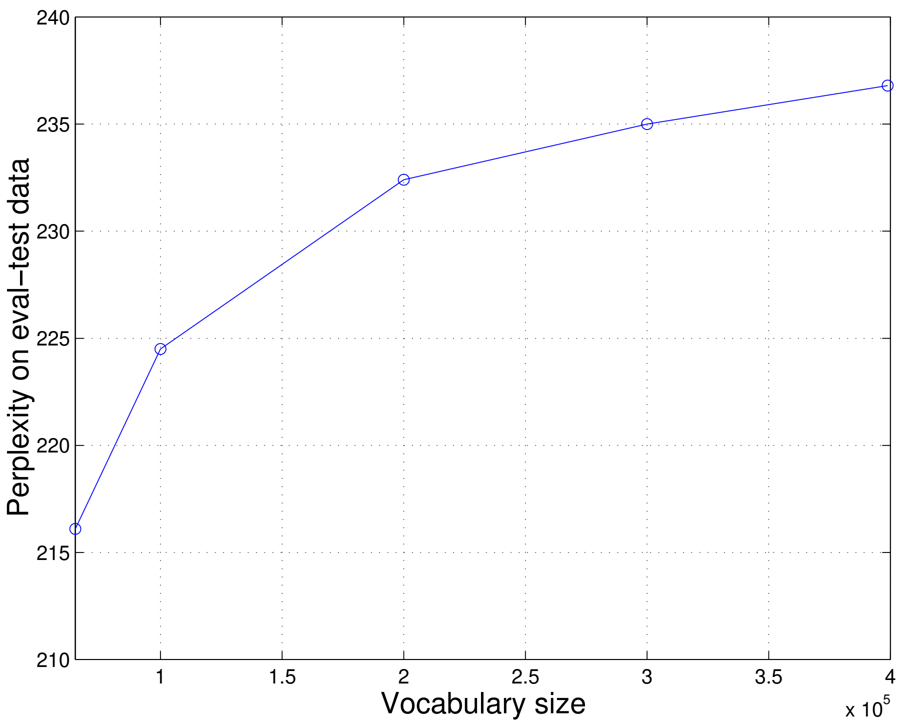


Figure 3.4 *Perplexity on English corpus* **`eval-test`** *data of word trigram models with various vocabulary sizes.*

The most striking result is for the 4-gram model when all N-grams are retained, for which the perplexity is 1.6% *higher* than when all singleton N-grams have been discarded. This is a direct consequence of the way in which the corpus has been partitioned. The implication is that the information provided by the singleton N-grams is less useful, indeed more harmful, than the probabilities that the back-off and Good-Turing discounting schemes assign to singleton events. Another interpretation is that the singletons represent "noise" in the corpus which it is desirable to remove in any case. It is also interesting to note that the hit-rates increase substantially when singleton events are retained but that the perplexity does not change much. This indicates that the backed-off estimates of these probabilities are almost as good (better in the case of the 4-gram model) as the discounted estimates when singleton events are retained.

The results obtained with the homogeneous corpus partition are shown in Table 3.8. Since there is more similarity in composition (homogeneity) between the `training` and `eval-test` sets, all the perplexity values are lower than for the heterogeneous case. In addition, the OOV-rate is now only 1.1% compared to 1.5% although the N-gram hit-rates are very similar in each case. The difference between the perplexities of models where N-grams are retained and where they have been discarded is markedly different to the trend that was observed for the heterogeneous corpus partition. When all N-grams are retained in the 4-gram model there is a small but appreciable *reduction* in perplexity compared to the 4-gram model from which singletons have been discarded. The improvement in the trigram model is greater than for the homogeneous partition indicating that the singleton events now provide more useful estimates

	Trigram model		4-gram model	
Cutoffs (bi,tri)	1, 1	0, 0	1, 1, 1	0, 0, 0
Perplexity	216.1	208.4	200.6	199.1
4-gram hits (%)	—	—	31.5	39.3
3-gram hits (%)	61.5	68.9	30.0	29.6
2-gram hits (%)	29.8	25.0	29.8	25.0
1-gram hits (%)	8.7	6.1	8.7	6.1

Table 3.8 *Homogeneous English corpus partition: perplexities and hit-rates with 65k vocabulary and different cutoffs.*

than the backed-off estimates.

3.4 Permutated-history N-gram language model

A series of experiments was conducted to investigate the assertion that due to the perceived potential for free-word ordering, N-gram models were unsuited to modelling the Russian language. The aim of the permutated-history N-gram language model was to incorporate an additional knowledge source into the standard back-off scheme which would exploit any occurrence of free-word ordering that occurred. If the N-gram, say (A, B, w) that is requested is not found in the model, instead of backing-off to the bigram (B, w) the N-gram with its history permutated (B, A, w) is first sought. Only if this permutated-history N-gram is not found, will the model then back-off to the $(N-1)$-gram distribution. A permutated-history trigram model was built and evaluated on the Russian corpus. The performance, however, was shown to be 5.8% worse than the conventional trigram model. This degradation in performance was attributed to the additional back-off stage which, if the permutated-history trigram was not found, reduced the probability of every final backed-off bigram or unigram estimate by the appropriate back-off weights. When the model did back-off from the trigram distribution, very few ($\approx$ 0.7%) of the permutated-history trigrams were in fact found. Consequently, many probability estimates were reduced unnecessarily which resulted in a higher perplexity.

The conclusion drawn from this experiment was that the potential for free-word ordering does not seriously reduce the effectiveness of the N-gram framework for modelling Russian. This conclusion is at least valid over the relatively short span of the three words in a word trigram model.

3.5 Summary

In this chapter, the composition of the Russian and English language corpora that are used for the experiments in this dissertation and the preprocessing that was necessary, have been described in detail. The characteristics of the two corpora in the context of language modelling were then examined. Back-off word N-gram language models were built for both corpora and the variation of perplexity and OOV-rate for different vocabulary sizes was investigated. It was shown that much larger vocabulary sizes are required for Russian in order to achieve the same coverage as for English. The requirement for larger vocabularies increased the already serious effects of sparsity in the Russian corpus.

The experiments in the next three chapters look at several methods for reducing the effects of data sparsity. The first methods that are considered involve the pooling of event statistics in class-based language models to produce more reliable probability estimates. The second set of methods involve reducing the vocabulary size using sub-word units (particles) to capture different dependencies in the language.

4

Class-based language modelling

In this chapter an investigation of class-based language modelling techniques is made using a two-sided class-based language model. After an introduction to the two-sided class-based model, an overview is given of several techniques described in the literature for automatic word classification. The results of a series of clustering experiments on the Russian and English corpora are then presented and class-based language models built for different numbers of word classes. Perplexity results for these models are reported and discussed. The last section in this chapter compares the performance of class-based language models that use automatically generated word classes and linguistic parts of speech. Both perplexity and word recognition results are used for this comparison.

4.1 Introduction

All class-based language models (hereafter referred to simply as class models) employ some component that uses word equivalence classes (often abbreviated to word classes) to capture dependencies in the training text. A deterministic word classification function (or class mapping function) of the form

$$C : w \rightarrow C(w), \tag{4.1}$$

assigns each word to one class only since the class mapping function is many-to-one. Having determined C, class N-gram language models can be built in a similar manner to word N-gram models in which the dependencies are captured by N-tuples of word classes rather than by N-tuples of words. The class N-gram model referred to in the following literature review and the work in this chapter is of the following form:

$$P_0(w_i \mid C(w_i)) \cdot P_1(C(w_i) \mid C(w_{i-N+1}), ..., C(w_{i-1})). \tag{4.2}$$

Ney refers to this model as the two-sided symmetric class model (Ney et al., 1994) since the same word classification function C is used to map both the current word and the predecessor words.

The model comprises two independent probability distributions: a unigram class membership component $P_0(\cdot)$ and a class N-gram component $P_1(\cdot)$ which is used to predict the current word's class from its predecessor word classes.

Generally, word classes are groups of words which are deemed to be similar in some way. If linguistic parts of speech are used to define the classes, all nouns might be grouped together in one of the classes, for example, and all adjectives in another. Alternatively, some statistical criterion of similarity can be used to determine the word classes. The latter is the focus of the work in this chapter. Both independent probability distributions in Equation (4.2) are optimised simultaneously using the maximum likelihood relative frequency estimates from the training data for the component conditional probabilities

$$P_0(w_i \mid C(w_i)) = \frac{N(C(w_i), w_i)}{N(C(w_i))} = \frac{N(w_i)}{N(C(w_i))}, \tag{4.3}$$

and

$$P_1(C(w_i) \mid C(w_{i-N+1}), ..., C(w_{i-1})) = \frac{N(C(w_{i-N+1}), ..., C(w_{i-1}), C(w_i))}{N(C(w_{i-N+1}), ..., C(w_{i-1}))}. \tag{4.4}$$

It is apparent that the optimal classification is likely to lie somewhere between the two extreme classifications that would be obtained if the two components were optimised separately. Consequently, there is a trade-off between the resolution of the unigram component, where each word would tend to prefer its own unique class, and the predictive ability of the N-gram component where all words would tend to be grouped in one and the same class.

The majority of the schemes presented below determine C by optimising the log-likelihood of a bigram class model LL_{bi} on the training data, using the relative frequency estimates given above

$$LL_{bi}(C) = \sum_{i=1}^{N_W} \log \frac{N(w_i)}{N(C(w_i))} \cdot \frac{N(C(w_{i-1}), C(w_i))}{N(C(w_{i-1}))}, \tag{4.5}$$

where N_W is the total number of words in the training data. A further simplification is obtained by grouping together similar terms

$$LL_{bi}(C) = \sum_{i=1}^{N_C} \sum_{j=1}^{N_C} N(c_i, c_j) \cdot \log \frac{N(c_i, c_j)}{N(c_i) \cdot N(c_j)} + \sum_{i=1}^{N_V} N(w_i) \cdot \log N(w_i), \tag{4.6}$$

where N_C is the number of word equivalence classes.

The above formula appears to be composed of two elements which may be interpreted as the average mutual information I between classes and the entropy H of the words in the training data (Brown et al., 1992)

$$LL_{bi}(C) = \frac{1}{N_W} \cdot (I(c_i : c_j) - H(w)). \tag{4.7}$$

The classification function has no effect on the entropy component which is constant for a fixed text, hence the average mutual information between classes can instead be used as an equivalent optimisation criterion which can be further simplified for implementation to

$$LL_{bi}(C) = \sum_{i=1}^{N_C} \sum_{j=1}^{N_C} N(c_i, c_j) \cdot \log N(c_i, c_j) - 2 \cdot \sum_{i=1}^{N_C} N(c_i) \cdot \log N(c_i) + \text{const.} \tag{4.8}$$

Once C has been determined, it is a relatively simple matter to construct the class model. Although most algorithmic methods determine the classification function using a bigram class model, the assumption is often made that the classification is suitable for mapping all N words in the N-gram. Both the probability distributions in Equation (4.2) can be modelled separately and combined by multiplication. There are far fewer free parameters to estimate in a class N-gram model than in a word N-gram model since in general $N_C \ll N_V$. There are $(N_C \cdot (N_C - 1) + N_V - N_C)$ parameters to estimate for a class bigram model compared to $N_V \cdot (N_V - 1)$ for a word bigram model. However, in many cases it is still necessary to apply some form of probability smoothing scheme. For the class N-gram distribution this can be performed in an identical manner as for word N-gram models (see Section 2.2.3). If necessary, the unigram distribution can be smoothed by setting a minimum unigram count on all words.

Several different algorithms for determining the classification function C will now be considered.

4.2 Survey of existing techniques

The way in which words are assigned to word classes is dictated by the way in which they will be used in the language model or speech recogniser. An intuitive linguistic assignment of words to classes may put all words that possess a common part of speech together in one class. Since English words generally possess more than one part of speech, this classification function would necessarily be many-to-many. All the data-driven approaches described in this section, except for the first method, permit a word to belong to only one class and the classification function is therefore many-to-one. This has the effect of significantly reducing the complexity of both the clustering algorithm and the resulting class model. The following sections describe in a more or less chronological order the developments in word clustering algorithms.

4.2.1 Nuclear parts of speech

In (Jelinek, 1990) a method which had its origins in an experiment conducted in (Cave and Neuwirth, 1980) is described for finding word classes through the application of an expectation maximisation (E-M) algorithm (Dempster et al., 1977). A hidden markov model interpretation of the class language model given by Equation (2.15) is used where words can belong to multiple classes and the model parameters are estimated through repeated application of the forward-backward algorithm (Baum, 1972). The class N-gram component is represented by a probability distribution $q(\cdot \mid \cdot)$ for the transitions between states $(C(w_{i+N-1}), ..., C(w_{i-1}))$ and $(C(w_{i+N}), ..., C(w_i))$ in the model. The unigram component is modelled by an emission probability distribution $p(\cdot \mid \cdot)$ associated with each state transition. This specifies the probability with which a word from the vocabulary is emitted during a particular state transition.

The nuclear part of speech approach recognises the shortcomings of linguistic part of speech assignments and is based on the assumption that a relatively small number M of words exhibit significantly different grammatical behaviour from each other. Given an initial nuclear set $\mathcal{M}$ the desire is to classify the remaining vocabulary words (words are allowed to belong to more than one class) into the M available classes. The initial nuclear set of $M = 200$ is chosen heuristically to be a mixture of the most frequent words in the vocabulary and those words which exhibit important grammatical characteristics which are not otherwise represented. A class trigram model is used and the two distributions are initialised as follows

$$q(c_3 \mid c_1, c_2) = \frac{1}{200}, \tag{4.9}$$

and

$$p(w \mid c) = \begin{cases} K(c)N(w) & \text{if } w \text{ is in } \mathcal{M}, \text{ and } c \text{ is the class of } w \\ 0 & \text{if } w \text{ is in } \mathcal{M}, \text{ and } c \text{ is not the class associated with } w \\ \frac{K(c)N(w)}{200} & \text{if } w \text{ is not in } \mathcal{M}, \end{cases} \tag{4.10}$$

where $K(c)$ is chosen to ensure that $p(w \mid c)$ adds up to one, when summed over all w for each c. After training, the class membership of a word is restricted to the three most probable classes and then a further few iterations of the forward-backward algorithm are executed.

Allowing multiple class membership in the final model requires the class model given by Equation (2.15). However if class membership is further restricted to only one class, using the most probable class for each word, then the model in Equation (4.2) can be used.

4.2.2 Hierarchical clustering

In (Brown et al., 1992) a greedy algorithm[1] is employed to cluster words into classes. The optimisation criterion that is used is the log-likelihood of the bigram class model on the training data $LL_{bi}(C)$, which was shown above to be equivalent to maximising Equation (4.8), the average mutual information between classes.

[1]Greedy algorithms consider each possible configuration one at a time. The effect of each configuration is evaluated without regard to configurations which are subsequently selected.

There are two main stages to this agglomerative algorithm which clusters words in a hierarchical bottom-up fashion. After initialisation, in which each vocabulary word is placed in its own unique class, the first stage of the algorithm consists in successively merging each pair of classes for which the reduction in $LL_{bi}(C)$ is the smallest. Since the initial state is the one for which $LL_{bi}(C)$ is a maximum, each merge of a pair of classes results in a reduction in $LL_{bi}(C)$. When a pair of word classes is merged, it is not necessary to recompute $LL_{bi}(C)$ entirely by performing the each summation again, since only those components in Equation (4.8) which are affected by the move need to be updated. A sensible arrangement of the update equations thus results in an algorithm of approximately order $\mathcal{O}(N_V^2)$. The algorithm continues to merge pairs of classes until the desired number of classes N_C has been obtained. (Note that if the algorithm were to run to completion, all words would eventually end up in one class.)

The second stage of the algorithm does not allow N_C to change and instead moves each word from its current class to the class for which the increase in $LL_{bi}(C)$ is the greatest. Once again, only those terms which are affected by the movement of a word from one class to another need to be updated and this can be performed in $\mathcal{O}(N_C^2)$ time.

Initial vocabulary partitions for vocabularies larger than 5000 words are obtained by assigning the N_C most frequent words to their own classes. The $(N_C + 1)$th most probable word is then assigned to its own class and those two classes (from the resulting $(N_C + 1)$), for which the reduction in $LL_{bi}(C)$ is the smallest, are merged as before. This brings the number of classes back to N_C again. This process is executed for the $N_V - N_C$ unclassified words and then the second stage of the algorithm is performed as before.

4.2.3 The exchange algorithm

In work that appears to have been conducted at around the same time as the hierarchical clustering method, another greedy approach to the clustering of words was being developed. In fact, the algorithm is essentially identical to the second stage in the hierarchical clustering method described above. Work on the *exchange algorithm*[2] for word clustering which has been reported over the last decade is more extensive than for the hierarchical clustering method.

In their first paper, Kneser and Ney use the leaving-one-out log-likelihood (see Section 2.3.3) as well as the maximum likelihood of a bigram class model on the training text as the optimisation criteria (Kneser and Ney, 1993). The benefit of using an approach which incorporates an element of cross-validation is that the resulting class model is prevented from over-fitting to the training data. The ability to generalise to unseen test data is therefore optimised and also enables the optimum number of classes to be found. It does appear, however, that this optimum number of classes only holds true for a stand-alone class model and the number of classes is not necessarily optimal for an interpolated class and word model.

As the name suggests, the exchange algorithm *exchanges* words between the N_C available classes so as to maximise some optimisation criterion. Each word is moved from its class to the

[2]Exchange algorithms are a conventional means of clustering a set of vectors by exchanging a vector between the available clusters (Duda and Hart, 1973).

class for which the increase in $LL_{bi}(C)$ is the greatest. Recalculating the optimisation criterion of a bigram class model each time would require $\mathcal{O}(N_C^3)$ computations, however, since the majority of the terms in Equation (4.8) are unaffected by a word movement, updating only the affected terms is of order $\mathcal{O}(N_C^2)$.

In general, the classification function, which is determined using a class bigram optimisation criterion, is used to map words in all N positions of the N-gram i.e. the classification function is independent of position. However, there have been attempts in (Martin et al., 1995) and (Martin et al., 1998) to optimise the classification function directly using the perplexity of a class *trigram* model on the training text. A comparison between trigram class models that were built and evaluated using the classification from a trigram clustering criterion was shown to give a reduction of up to 6% in perplexity over the classification obtained using a bigram criterion. This perplexity improvement was negligible however, when the class models were combined with the word model. The absolute clustering times using the trigram criterion made it prohibitive for large numbers of classes and the computational effort does not normally justify the perplexity improvements.

It has already been pointed out that greedy algorithms only have local convergence and hence the choice of initialisation is generally of great importance. In (Martin et al., 1998) three initialisation methods are investigated: a *baseline initialisation* assigns the most frequent $(N_C - 1)$ words each to their own unique class and all remaining words fill the last class; a *random initialisation* assigns words to classes randomly and evenly; and a *POS initialisation* uses 33 linguistic part of speech tags and leaves the remaining $(N_C - 33)$ classes empty. Each initialisation method is shown to produce approximately the same results indicating that the algorithm is insensitive or fairly independent of the initialisation method. It could also mean, of course, that another better method has yet to be found. Martin et al. also note that there is no basic performance difference between the hierarchical clustering method described above and the three initialisation methods that were investigated for the exchange algorithm.

4.2.4 Simulated annealing

Simulated annealing gets its name from the analogy with the physical annealing of solids that occurs in nature. The main characteristic which distinguishes this method from the previous greedy methods is that configurations which result in a temporary decrease in the objective function that is being maximised can be accepted. Configurations which result in an increase in the objective function are always accepted, while those resulting in a decrease are accepted with probability given by the Metropolis criterion

$$\exp\left(\frac{PP_i - PP_{i+1}}{c}\right), \tag{4.11}$$

where PP_i is the perplexity at iteration i and c is a *control parameter* (analogous to a *temperature* dependent variable in physical systems) which is itself governed by an *annealing schedule*.

For the purposes of word classification problems the objective function is often the perplexity (or alternatively log-likelihood) of a class model on the training data. Equation (4.11) forms the basis of the *Metropolis algorithm* and the *acceptance criterion* of accepting a new configuration at time $(i+1)$ is

$$P_c\{accept\ configuration\} = \begin{cases} 1 & \text{if } f(i+1) \geq f(i) \\ \exp\left(\frac{f(i)-f(i+1)}{c}\right) & \text{if } f(i+1) < f(i) \end{cases} . \tag{4.12}$$

Simulated annealing has been applied successfully to several tasks including a solution to the Travelling Salesman problem (Press et al., 1992), and for finding a locally optimal layout of tracks between components in VLSI chips. Simulated annealing is proven to locate a globally optimal solution (Aarts and Korst, 1990), under the rather unrealistic requirement that an infinite number of configurations are considered at each iteration, and that the control parameter is reduced logarithmically. The first application of simulated annealing to word classification problems for language modelling was presented in (Jardino and Adda, 1993a) where the class model in Equation (4.2) is used. The fundamental difference with the simulated annealing method is that new configurations are chosen randomly (Monte Carlo selection of both word and destination class) and that configurations which increase the perplexity may also be accepted. Multiple rearrangements of several words to several destination classes have also been investigated.

Experiments on small (around one hundred thousand words) French and German corpora are described in (Jardino and Adda, 1993a; Jardino and Adda, 1993b) where small perplexity improvements over the exchange algorithm approach are reported. The best results are obtained using an initialisation of all words in one class, and by moving a multiple of the number of vocabulary words for each value of the control parameter which is decreased logarithmically. In (Jardino, 1996) a class trigram model built using the classification that was obtained using a bigram criterion, is shown to have a lower perplexity on test data than each of the word trigram models trained on large ($> 30 \times 10^6$ words) English, French and German corpora. The low perplexity figures (particularly unexpected for English) were attributed to the inclusion of punctuation in the texts. In (Jardino and Adda, 1994) a refinement is described that allows words to belong to two classes and gives a 20% perplexity reduction on test data, compared to the equivalent single-class membership model. Multiple-class membership is not restricted to the simulated annealing clustering approach but the results do indicate that there is potential for further improvement if words are permitted to belong to more than one class.

4.3 Automatic word clustering experiments

In this section the operation of the exchange algorithm is examined in more detail and perplexity experiments are performed on the Russian and English corpora that were described in the previous chapter. The exchange algorithm is used to determine the class mapping function by optimising the perplexity on the training data of the two-sided, symmetric class bigram model given by Equation (4.2). The classification function is then used to build class trigram models. The main aim of this section is to examine the performance of a well-established clustering algorithm in terms of the clustering times and the performance of the class models that are built using the resulting class mapping function. The performance of the class models that are built are evaluated in terms of the number of classes and the language involved. The work in this section extends the experiments that were conducted on the Russian data and published in (Whittaker and Woodland, 1998). In addition, the analysis here will provide a useful performance benchmark for the experiments in the next chapter where a clustering algorithm for a one-sided class model is presented.

4.3.1 Exchange algorithm for the two-sided class model

A brief outline of the workings of the exchange algorithm has already been presented in Section 4.2.3. A thorough analysis of the update equations and efficient implementation details are given in (Martin et al., 1998) and in order not to repeat that analysis, only the details which are considered relevant to experiments in subsequent chapters will be given here. The essential points in the operation of the algorithm are given below:

1. **Initialisation**: `initial configuration of words to classes`
2. **Iterate** `for all words` w:
   - **Iterate** `for all classes` c:
     - `'Move' word` w `to class` c
     - `Compute change in likelihood`
     - `'Remove' word` w `from class` c
   - `Move word permanently to best class`
3. `Repeat step 2 for fixed number of iterations`

4.3.1.1 Update equations

The viability of the exchange algorithm relies on there being an efficient means of updating the optimisation criterion each time a word is moved. The set of update equations presented below allows the implementation of an algorithm of order $\mathcal{O}(N_C^2)$ by updating only those counts that are affected when a word w_i is moved from class c_j to class c_k (Martin et al., 1995)

$$\forall c \neq c_j : N(c, c_j) = N(c, c_j) - N(c, w_i), \tag{4.13}$$

$$\forall c \neq c_j : N(c_j, c) = N(c_j, c) - N(w_i, c), \tag{4.14}$$

$$N(c_j, c_j) = N(c_j, c_j) - N(c_j, w_i) - N(w_i, c_j) + N(w_i, w_i), \tag{4.15}$$

and similarly

$$\forall c \neq c_k : N(c, c_k) = N(c, c_k) + N(c, w_i), \tag{4.16}$$
$$\forall c \neq c_k : N(c_k, c) = N(c_k, c) + N(w_i, c), \tag{4.17}$$
$$N(c_k, c_k) = N(c_k, c_k) + N(c_k, w_i) + N(w_i, c_k) - N(w_i, w_i), \tag{4.18}$$

where for example

$$N(w_i, c) = \sum_{\forall w : w \in c} N(w_i, w). \tag{4.19}$$

It is pointed out in (Martin et al., 1998) that the counts $N(w_i, c)$ and $N(c, w_i)$ only have an effect if they are non-zero. Since many words are infrequent, the number of classes for which this is the case is generally much less than N_C. A significant amount of unnecessary computation can be avoided by only considering the predecessor and successor word classes for which $N(w_i, c) \neq 0$ and $N(c, w_i) \neq 0$, respectively. The implementation used in the experiments here also maintains a reversed bigram lookup table so the count $N(w, w_i)$ of words w occurring before word w_i can easily be retrieved.

4.3.1.2 Computational complexity

The update equations presented above facilitate the computation of the optimisation criterion in $\mathcal{O}(N_C)$ time each time a word is moved from one class to another class. Since, over each iteration I, each word in the vocabulary must be moved (tentatively) from its original class to all possible destination classes, there is a $\mathcal{O}(I \cdot N_V \cdot N_C^2)$ complexity to the algorithm which dominates due to the $\log$ operations in the innermost loop where the change in likelihood is computed. However, for small numbers of classes, the size of the training data N_W may also become a dominant factor. This manifests itself as the number of unique bigrams (containing only vocabulary words) in the training data B (where $B \ll N_W$ in general) which is factored into the count generation procedure. The complexity of the algorithm can therefore be shown to be

$$\mathcal{O}(I \cdot (2 \cdot B + 2 \cdot N_V \cdot N_C^2)), \tag{4.20}$$

where the $2 \cdot B$ factor originates from the generation of counts $N(w_i, c)$ and $N(c, w_i)$ and because the implementation does not involve any search in looking up the necessary bigram counts. In addition, by only considering the N_C^{pre}, predecessor and N_C^{suc}, successor word classes for which $N(w_i, c) \neq 0$ and $N(c, w_i) \neq 0$ the complexity can be reduced still further to

$$\mathcal{O}(I \cdot (2 \cdot B + N_V \cdot N_C \cdot (N_C^{pre} + N_C^{suc}))). \tag{4.21}$$

This results in a significant reduction in the complexity of the algorithm which is consequently made to be "more than linear but far less than quadratic [in the number of classes]" (Martin et al., 1998).

4.3.1.3 Model parameters

Additional comparisons will be made between models on the basis of how many parameters are stored in each model. There is not necessarily a parameter stored for every possible event that the model may have to handle. The number of parameters represents the number of observed events for which explicit parameters are stored in the model. The parameters for these events can then be used to compute probability estimates for other events which were unobserved or discarded (see Section 2.3). A model with many parameters ought to perform better than a model with fewer parameters, however, this is often not the case since some parameters may be poorly estimated or even redundant. In the experiments in this dissertation, the number of parameters in a trigram model is taken to be the sum of the number of unigrams, bigrams and trigrams that are stored in the model. Strictly speaking, there is also a backoff weight associated with each unigram and bigram context stored in the model, and various other parameters defining discounting coefficients, however these are not included in the number of parameters. In addition, the parameters in the class membership component were not factored into this number, however, since their number is small in comparison with the number in the N-gram component this is a reasonable approximation.

4.3.2 Experimental procedure

The experimental procedure employed for obtaining the word classes and subsequently building the class trigram models was identical for both corpora so as to allow as comprehensive and legitimate a comparison as possible. The clustering algorithm requires: a list of word bigram and unigram counts for all vocabulary words in the training data; the number of classes N_C into which words can be clustered to be specified; and, an initial classification of words into the N_C classes. A word bigram language model was built to serve the first requirement and the initial classification assigned the most frequent $(N_C - 1)$ vocabulary words to their own unique class and all remaining vocabulary words were placed in the N_Cth class. Further to this, four special symbols were each placed in their own unique classes and could not be moved from these classes nor could other words be moved to them during the clustering operation. These special symbols are: <s>, </s>, <NUMBER> and <UNK> (sentence-begin, sentence-end, number[3] and unknown-word symbols). The extra four classes account for the $(N_C + 4)$ classes in all the experiments below. It should be noted that all vocabulary words, other than the four mentioned above, were considered for classification into any of the N_C classes, and no minimum unigram count was placed on a word for it to be considered to be moved. Words are moved in order of decreasing frequency of occurrence in the training data and two whole iterations through the vocabulary are performed in every case.

The value of N_C was varied to determine the effect of different numbers of classes on the performance of the class model. Words were clustered into 204, 504, 1004, 2004, 3004, 4004 and 5004 classes. The vocabulary size was fixed to be the most frequent 65,000 words as used in the word N-gram experiments in chapter 3. After two iterations, all vocabulary words had been

[3] The requirement for a number-symbol was explained in Chapter 3.

moved up to two times and this final classification function was then used to construct back-off class trigram models. The class trigram component was built and smoothed in an identical manner to the back-off word N-gram models in the previous chapter: singleton bigrams and trigrams were discarded in all cases and back-off models built using Good-Turing discounting and Katz's backing-off scheme. Since singletons were discarded from the word trigram model it is reasonable from a statistical point of view not to include singletons in the class models.

Each vocabulary word occurred at least once in the training data, so the class membership component was simply estimated using the relative frequency of the word unigram count and the class unigram count in the training data. No further smoothing was necessary. The two component probabilities were simply multiplied together to obtain the word-level probability of the class model.

4.3.3 Results

The times in hours to complete one iteration of the algorithm for different numbers of classes on a 300MHz Ultra 2 Sun Sparc Workstation are given in Table 4.1 for both the Russian corpus and the English corpus.

No. of	Hours per iteration	
classes	Russian corpus	English corpus
204	0.8	0.7
504	2.1	1.9
1004	5.9	5.5
2004	17.7	16.3
3004	33.9	36.4
4004	61.4	53.7
5004	99.0	91.5

Table 4.1 *Hours per iteration for different numbers of classes on a Sun Ultra2 processor.*

The results presented in the following tables and graphs show the perplexities computed on the `eval-test` set of each corpus of the stand-alone class trigram models and of the interpolated word trigram and class trigram models. The interpolation weights were chosen so as to optimise the perplexity on the appropriate `dev-test` portion of each corpus using a tool that uses the E-M algorithm (Rosenfeld, 1994) to perform the optimisation. The relative perplexity improvement of each interpolated model over the stand-alone word trigram model is given in the fifth column of the table and the number of parameters in each stand-alone model is given in the final column. Results are reported for seven different numbers of classes in Table 4.2 for the Russian corpus and in Table 4.3 for the English corpus. To ease comparison, the same results are plotted in Figures 4.1 and 4.2 for the Russian and English corpus respectively.

No. of classes	Perplexity PP_{class}	PP_{interp}	weights $\lambda_{word}, \lambda_{class}$	% improvement over word trigram	Class model size (parameters)
204	720.1	375.5	0.75, 0.25	9.1	1,753,940
504	558.0	358.1	0.66, 0.34	13.4	4,152,540
1004	484.2	349.6	0.61, 0.39	15.4	6,341,130
2004	436.5	345.8	0.57, 0.43	16.3	8,318,650
3004	414.4	346.4	0.55, 0.45	16.2	9,159,020
4004	405.2	348.6	0.55, 0.45	15.7	9,583,230
5004	400.1	351.4	0.55, 0.45	15.0	9,830,420
65000	413.3	—	—	—	10,896,660

Table 4.2 *Russian corpus (65k): perplexity on* **eval-test** *data of stand-alone class trigram models and interpolated class and word trigram models.*

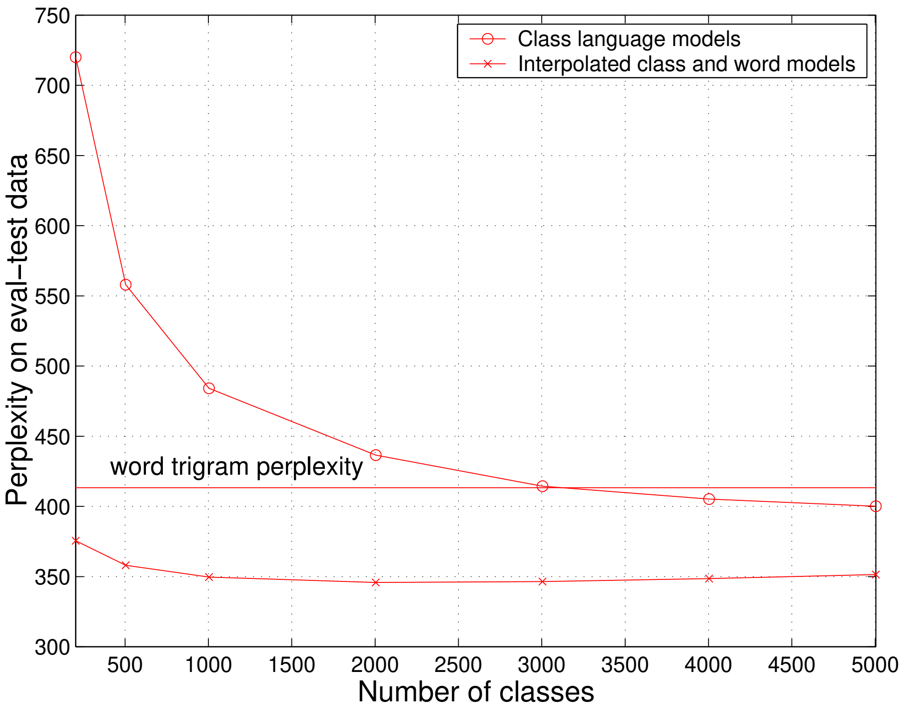


Figure 4.1 *Russian corpus (65k): perplexity on* **eval-test** *data of stand-alone class trigram models and interpolated class and word trigram models.*

No. of classes	Perplexity PP_{class}	PP_{interp}	weights λ_{word}, λ_{class}	% improvement over word trigram	Class model size (parameters)
204	383.1	205.5	0.78, 0.22	4.9	1,850,350
504	306.8	201.0	0.72, 0.28	7.0	5,424,600
1004	268.8	199.2	0.67, 0.33	7.8	7,975,660
2004	242.8	199.1	0.64, 0.36	7.9	10,110,760
3004	231.5	200.5	0.63, 0.37	7.2	10,982,810
4004	226.4	202.0	0.62, 0.38	6.5	11,384,370
5004	223.5	203.5	0.62, 0.68	5.8	11,596,280
65000	216.1	—	—	—	12,431,060

Table 4.3 *English corpus (65k): perplexity on* `eval-test` *data of stand-alone class trigram models and interpolated class and word trigram models.*

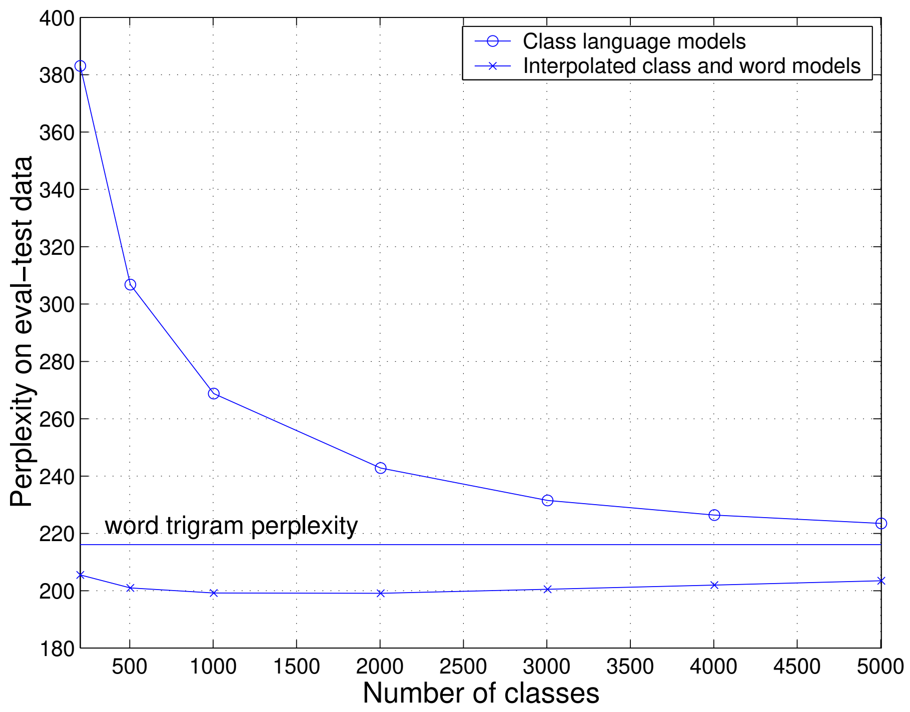
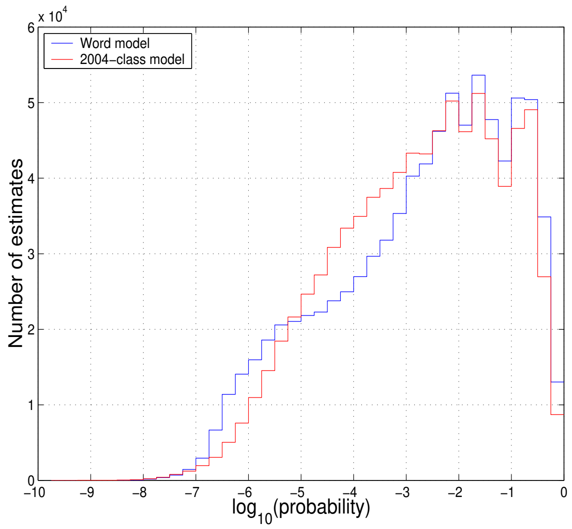


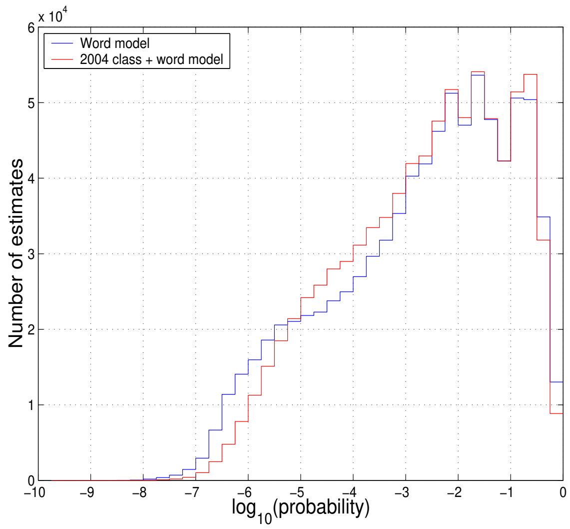
Figure 4.2 *English corpus (65k): perplexity on* `eval-test` *data of stand-alone class trigram models and interpolated class and word trigram models.*

4.3.4 Analysis of model performance

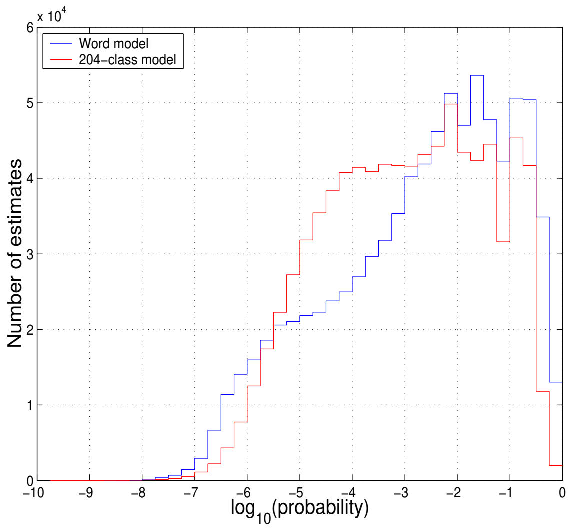
The performance of seven class models that were built for each corpus has been presented in the previous section. It is now necessary to analyse the models in more detail to determine where the deficiencies and strengths of these models lie. Since there are too many models to be able to analyse each individually, the analysis will examine only the best interpolated model and the best stand-alone class model for the Russian corpus only.



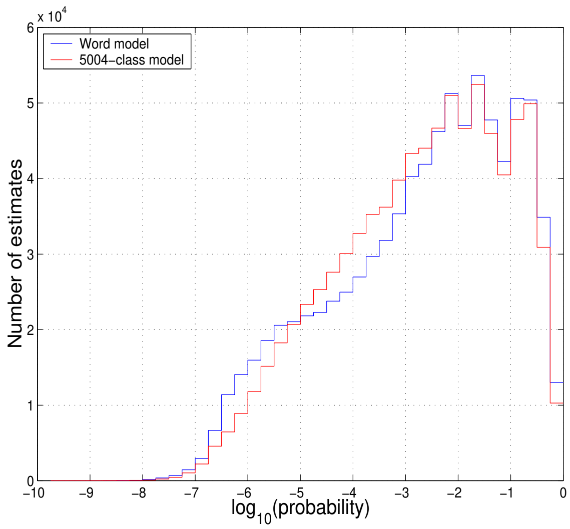
(a) 2004-class model.



(b) Interpolated word and 2004-class model.



(c) 204-class model.



(d) 5004-class model.

Figure 4.3 *Russian corpus (65k): distribution of log-probabilities for trigram word and class models.*

Figure 4.3 shows the distribution of log-probabilities assigned to words in the `eval-test` set by the different models. In each sub-figure the distribution of log-probabilities assigned by the word model is repeated to aid comparison with the distributions of the three class models and the interpolated class and word model. The plots themselves are obtained by partitioning the log-probabilities into equally spaced *bins* in the log_{10} domain. The area under each plot is approximately equal to the log-likelihood of the `eval-test` data i.e. it is directly related to the perplexity of the model.

It is immediately clear that the word model and the class model apportion probabilities quite differently. In particular, the class model appears to distribute the probabilities more evenly, with a lower proportion generally being assigned probabilities smaller than 10^{-5}. In contrast, the word model predicts more words with a high probability (> 0.1) than the class model does. The conclusion that can be made from this is that the class trigram events which capture less specific dependencies than word trigram events are more robustly estimated, hence the class model backs-off less frequently and consequently assigns fewer very low probabilities. Conversely, the high-probability estimates result from strong word relationships which the class model does not capture as well[4].

The distribution of log-probabilities in the interpolated model indicates that the best aspects of both models have been preserved: the large number of high-probability estimates of the word model and the small number of very-low probability estimates of the class model. The significant perplexity reduction that is obtained may well be attributed to the reduced number of very low probability estimates[5]. The distribution of very small probabilities ($< 10^{-7}$) in the interpolated model is particularly intriguing since there are fewer than in either of the component models. When this phenomenon was investigated more closely, it was observed that a significant proportion of the very low probabilities of one model are combined with much higher probabilities from the other model resulting in higher probability estimates on average. It is clear why the class model will often assign a higher probability than the word model to previously unseen word sequences, but not so clear the other way round. In general, very low probability estimates ($< 10^{-7}$) often involve a back-off to the model's unigram distribution[6] and further analysis showed that this was indeed the case. When very low probability estimates were produced, the word model's backed-off unigram distribution often gave a higher probability estimate to an event than the class model's backed-off unigram distribution.

The stand-alone 204-class model, which does not perform particularly well, has a distribution of log-probabilities that differs significantly from that of the word model. The distribution is much flatter for probabilities between 10^{-5} and 10^{-3} and there are relatively few very low and very high probabilities. A good model will preserve the small number of very low probabilities and have a large number of high probabilities. The interpolated model demonstrates this and to some extent so does the 5004-class model. In fact, the latter has a similar distribution of very low and very high probabilities to the word model but a greater number of probabilities between 10^{-5} and 10^{-3} (the word model has more between 10^{-7} and 10^{-5}) which helps to explain its lower perplexity.

[4]If a particular N-gram event is modelled well then the probability assigned to that N-gram event will generally be higher than a backed-off $(N-1)$-gram probability estimate.

[5]It should be remembered that here perplexity is a geometric average of the probabilities of events in the test data hence small probabilities can have a large effect on the overall probability. By interpolating two sources, many very small probabilities may only be increased by a small amount, yet have a significant effect on the overall perplexity (Rosenfeld, 1994).

[6]Note that because all bigram and trigram singletons were discarded from both class and word models, and since the class mapping is deterministic, whenever the class model backs-off, the word model must also back-off.

	Perplexity of different models			
Back-off case	204-class	2004-class	5004-class	65k word
3	196	101	85.6	71.6
3-2	1.28×10^3	820	748	655
3-2-1	2.90×10^4	2.96×10^4	3.62×10^4	1.23×10^5

Table 4.4 *Russian corpus (65k): perplexities of several models computed on sets of events that were predicted by different word model back-off cases.*

Table 4.4 shows the perplexities of three different class models and the word model for the 65k Russian vocabulary where the perplexity of each model is computed on those events that are predicted by the three[7] different backoff cases of the word model i.e. when the trigram word model finds the requested trigram (3), or backs-off to either the bigram (3-2) or unigram (3-2-1) distribution. The results clearly show the relative performance of the different models for the different backoff cases of the word model. In particular, the 204-class model has the lowest perplexity when the word model backs off to the unigram distribution but the highest perplexity when the requested trigram is found by the word model. This clearly illustrates the class model's ability to generalise to unseen word sequences. The 5004-class model, which has a 3% lower perplexity overall than the word model, has a 15-20% higher perplexity for the cases where the word model finds the trigram or only backs off to the bigram distribution. This is explained by the word model's ability to capture more specific dependencies. However, when the word model backs off completely to its unigram distribution, the 5004-class model has a 70% lower perplexity than the word model. The 2004-class model which forms the best interpolated model has perplexity characteristics which best complement the specificity of the word model.

To complement this quantitative analysis, an investigation of the contents of the word classes that were obtained showed many of them to be intuitively appealing. Judging by the contents of the ten randomly chosen classes given in Table 4.5 on the next page, it is clear that the contents of some classes are consistent in terms of their part of speech e.g. class 10, and other classes have obvious semantic similarities e.g. class 9. Also, most of the written numerals below twenty form their own class like “EIGHT” in class 4. However, it is hard to find a consistent attribute to describe some other classes e.g. class 5. Interestingly, class 6 contains what appear to be corrupted (intentionally or accidentally) forms of the word “*TO*”, together with the correct spelling of the word. Some of the classes for Russian (not shown) contain the majority of a set of word inflections, for example, all six masculine word-forms of the adjective “*superfluous*” appear in one class together. In general, a more consistent attribute (subjectively determined) can be found for the classes that were obtained on the Russian corpus than for those obtained on the English corpus.

[7]The bigram request at the beginning of the `eval-test` text has been ignored.

Class 1	Class 2	Class 3	Class 4	Class 5
EVEN EQUIVALENTLY	I'D YOU'D WE'D	HIMSELF MINDEDLY THEMSELF	EIGHT	CONCERNED WRONG HAPPENING RAINING BERSERK SNOWING GROUNDLESS NODED AGOG NEEDFUL

Class 6	Class 7	Class 8	Class 9	Class 10
TO TER TAE	AVAILABLE PAYABLE PRICED REDUNDANT TRANSMITTED UNDERWAY UNAVAILABLE REFERENCED RECOVERABLE OBTAINABLE	SOCIETY CULTURE POLITICS LITERATURE DEMOCRACY RELIGION CONSCIOUSNESS IDEOLOGY CAPITALISM CHRISTIANITY	RELATIONSHIP RELATIONSHIPS CONFLICT LINK CONVERSATION CONNECTION LINKS COMPARISON PARTNERSHIP TENSION	GONE GROWN BEGUN FALLEN SPOKEN RISEN ARISEN SPRUNG LAIN SHRUNK

Table 4.5 *All, or up to the ten most frequent, words from ten randomly chosen classes of the 1004-class English model.*

4.3.5 Discussion

For both corpora, as the number of classes is increased the perplexity of the stand-alone class model decreases, however this trend cannot continue indefinitely. If the number of classes is increased to the point where each word resides in its own class, the word model is obtained as a result. The optimum number of classes for the Russian corpus may be expected to lie somewhere between 4004 and 65000 classes. For the English corpus it is conceivable that the word model in fact represents the optimal classification of words for a stand-alone model.

When the class model is interpolated with the word model there is clearly a particular number of classes which results in optimal performance. For both the Russian corpus and the English corpus, the optimal number lies between 1004 and 3004 classes[8]. The interpolation weights also indicate that as the number of classes increases, the weighting given to the class model increases. This trend will probably continue to the point where each model is given an equal weighting by the E-M algorithm. Even when the stand-alone class model outperforms the word model, its weight is less than that assigned to the word model. This last point, that the class models are able to outperform the word models for the Russian corpus is particularly interesting. For English, class models tend always to give worse perplexity results than the word model when there is sufficient training data available. For the Russian corpus, of those models investigated, the two using 4004 and 5004 classes outperform the word model and also have fewer

[8]Referring to section 3.3 on the effect of different corpus partitioning techniques, it is interesting to note that the optimal number of classes for the heterogeneous English corpus partition is actually less than for the homogeneous partition (results not shown).

parameters than the word model.

Since both the Russian and English corpora are of a similar size it is clear that the large number of unique words in the Russian training data means that its data sparsity is likely to be greater. The class models reduce the effects of sparsity to a certain degree by pooling the statistics of contextually similar words. This makes the probability estimates more robust at the expense of making them less precise. The explanation for the variation in performance with different numbers of classes is associated with a trade-off in the model's ability to generalise to previously unseen word sequences while maintaining the accuracy with which words are predicted. If there are too few classes then over-generalisation occurs and the probabilities are not specific enough. If there are too many classes then the idiosyncrasies of the training data are captured resulting in over-specialisation and a reduction in the ability to generalise.

The clustering times for the algorithm on both corpora are similar and reflect the fact that both corpora are of a similar size and that the vocabulary sizes are the same. The size of the training data manifests itself in the number of distinct bigrams that must be retrieved during the clustering process. It is clear that this implementation of the clustering algorithm scales slightly less than quadratically in the number of classes[9]. From the analysis in section 4.3.1.2, it is clear that the algorithm should scale approximately linearly in the number of vocabulary words. Unfortunately, the long clustering times for the above models and the way in which the algorithm scales made further investigation of the algorithm for the 430k Russian vocabulary and larger numbers of classes impractical.

4.4 Word recognition experiments

The aims of the experiments in this section[10] were to evaluate the difference in performance between class models built using automatically generated word classes and those built using 152 part of speech (POS) classifications. The exchange algorithm was employed to obtain the "automatically generated" word classes. All class models were interpolated with a baseline word trigram language model that used Good-Turing discounting and had singleton bigrams and trigrams removed. Additionally, a minimum unigram count of ten was applied. The performance evaluation is both in terms of perplexity on held-out data and word recognition experiments using n-best re-scoring (Schwartz and Chow, 1990). All class N-gram models were built using the variable-length N-gram model (Niesler and Woodland, 1996a) which was described in Section 2.4.2. N-grams are progressively extended and only retained if doing so decreases the leaving-one-out training set likelihood by at least some fraction λ_{ct}. The number of parameters retained by the model can thus be varied according to the value of λ_{ct}.

Using part of speech classifications for words means that a many-to-many vocabulary map-

[9]The implementation of the clustering algorithm in (Martin et al., 1998) scales more than linearly but much less than quadratically in the number of classes, however, the corpus that was used is different and its size is a third the size of the corpora used in the experiments here.

[10]The work in this section contains experiments performed jointly by the author and Thomas Niesler in the Cambridge SVR group and are published in (Niesler et al., 1998). It should also be noted that these experiments pertain only to the English language.

ping exists. A model of the form given by Equation (2.15) was therefore required. With the automatically derived word classifications the model given by Equation (4.2) was suitable, since a word is only allowed to belong to one class.

4.4.1 Results

All experiments were performed on the Wall Street Journal (WSJ) Corpus (Paul and Baker, 1992). The language modelling data consists of 37 million words of newspaper text from the WSJ newspaper over the years 1987-89. Approximately the first 19,000 words from the language model development test data for each year were taken to form a 59,000 word test data set for perplexity calculations. Recognition experiments used the development (R-DEV) and evaluation (R-EVAL) tests which formed part of the 1994 ARPA CSR HUB-1 evaluation, the composition of which is shown in Table 4.6.

	Sentences	Speakers	Words
R-DEV	310	20	7,388
R-EVAL	316	20	8,190

Table 4.6 *Composition of the 1994 ARPA CSR HUB-1 development (R-DEV) and evaluation (R-EVAL) test sets.*

Two iterations of the exchange algorithm were executed to generate classifications for all 65k words of the vocabulary. Classifications were obtained for 150, 200, 500, 1000 and 2000 classes. Variable N-gram language models were then constructed using these classifications. Two models were built for the 150-class classification, one of which was forced to contain approximately the same number of parameters as the model that used POS classifications. This was achieved by specifying a value for λ_{ct} which determines the number of model parameters and allows a fair comparison of the performance of the two classifications to be carried out.

The perplexity figures on the test portion of the corpus for the class models and baseline word model are shown in Table 4.7. The number of parameters in each model is also shown.

Model type	N_C	Parameters	PP
POS	152	909,540	448.5
XCHNG ALG	150	1,038,770	301.1
XCHNG ALG	150	2,134,920	289.5
XCHNG ALG	200	2,697,020	265.8
XCHNG ALG	500	4,677,640	212.2
XCHNG ALG	1000	6,384,710	184.4
XCHNG ALG	2000	8,376,950	167.8
WORD	—	4,884,860	148.8

Table 4.7 *Class model characteristics and perplexity (PP) on LM-DEV.*

Lattices were generated for each R-DEV and R-EVAL test set using the HTK large-vocabulary speech recognition system (Woodland et al., 1995) with a 65k vocabulary and backoff bigram language model. These lattices were then rebuilt using the baseline word trigram model to ensure that the language models were trained on the same text. n-best ($n = 100$) lists were generated using the Entropic Lattice and Language Modelling Toolkit (Odell and Niesler, 1996) and re-scored using a linear interpolation of the class and word language models. The interpolation weights were chosen so as to minimise the word error rate on R-DEV. The recognition results together with the perplexity on LM-DEV is shown for the interpolated word and class models in Table 4.8.

		Perplexity			WER	
Model type	N_C	LM-DEV	R-DEV	R-EVAL	R-DEV	R-EVAL
POS	152	139.4	185.4	183.2	11.5	12.3
XCHNG ALG	150	142.2	189.7	187.1	11.1	12.2
XCHNG ALG	150	139.1	186.4	184.4	11.1	11.9
XCHNG ALG	200	136.9	184.5	181.9	11.0	11.9
XCHNG ALG	500	131.7	180.8	175.7	10.8	11.7
XCHNG ALG	1000	129.7	179.0	175.0	10.9	11.8
XCHNG ALG	2000	129.4	179.9	176.1	10.9	12.0
WORD	—	148.8	206.2	201.8	11.9	12.5

Table 4.8 *Perplexity and word error rates (WER) for the interpolated class and baseline word trigram models.*

4.4.2 Discussion

The perplexity results in Table 4.7 for the stand-alone class language models show that models which used automatically derived classes all had lower perplexities than the model that used POS classifications. In particular, this was true when 150 automatically derived classes were used in a model similar in size to that of the POS model. From Table 4.8 it is seen that this is also true for the word error rate of the combined class and word models even though the perplexities of the combined word and POS model are lower. In the literature, permitting multiple class membership has generally resulted in significant reductions in perplexity which shows that the POS classification is a particularly sub-optimal classification of words for this data set and vocabulary.

The automatic clustering technique allows classifications into different numbers of classes and for this task the model that used 500 classes resulted in the lowest word error rate. The relative improvement of the interpolated 500-class model over the baseline word trigram model was 7%. As the number of classes was increased, the performance began to deteriorate, since the class model begins to generalise less well to unseen word sequences. It is this generalising ability of the class model which complements the specific predictive power of the word model, and for which there is an optimum number of classes.

Analysis of the contents of the automatically derived classes revealed that words were distributed much more evenly than in the case of the POS model. Words which were themselves very frequent were often assigned a class of their own by the clustering algorithm. POS classifications group words only according to their grammatical function and with no regard to their statistical properties. Very infrequent words may be assigned their own class and frequent words may be grouped together, which is not optimal from a statistical point of view.

As an adjunct to the experiments conducted here, a two-sided class trigram model using one thousand automatically generated classes was interpolated with a word 4-gram model in the 1997 HTK Broadcast News transcription system (Woodland et al., 1998). The inclusion of the class model reduced the overall word error rate by 3% relative. The final system had the lowest word error rate overall in the 1997 DARPA Broadcast News evaluation by a statistically significant margin.

4.5 Summary

This chapter has given an overview of several techniques that have been used in the literature for determining the word classification function. Experiments on the Russian and English corpora using the exchange algorithm to obtain word classifications were then described and back-off class trigram models built using these word classifications. For the Russian corpus, two different stand-alone class models were shown to outperform the word model and also contained fewer parameters. For both corpora, the interpolated class and word model gave significant perplexity improvements over the stand-alone word model. The best improvement on the Russian corpus was twice as large as the best improvement for English, and this was attributed to the greater sparsity of the Russian corpus.

Lastly, a comparison in terms of perplexity and recognition performance was made between the performance of class models with part of speech classifications and automatically generated classes. All interpolated class and word models gave an improvement in terms of perplexity and word error rate but the best performance was obtained using the automatically generated classes.

5

One-sided class-based language modelling

The work in this chapter builds upon the experiments and the models developed in the previous chapter where a two-sided class model was considered. The work in this chapter concentrates on a particular type of one-sided class model. After an introduction to the one-sided class model that will be used in subsequent experiments, an exchange algorithm is developed for the one-sided model similar to that used for generating word classifications for the two-sided model. A series of clustering experiments is then performed on both the Russian and English corpora, and one-sided class models are built for different numbers of word classes. Particular emphasis is placed on the speed of the clustering algorithm. Finally, a comparison is made between the performance of the one-sided models and the two-sided models presented previously, in terms of the perplexity of models and the clustering times for the two different types of class model.

5.1 Introduction

The experiments in the previous chapter concentrated on the two-sided symmetric class model which combines what may be thought of as separate state transition and state emission distributions, with the same class mapping function for both the current and predecessor words. In this chapter, a one-sided class model is considered in which a state mapping is used for the predecessor words only, and the current word that is being predicted is not mapped at all. The state mapping is an $(N-1)$-tuple of word classes. The one-sided class N-gram model that will be used throughout this chapter maps words in the history as follows:

$$P(w_i \mid C_{N-1}(w_{i-N+1}), ..., C_1(w_{i-1})). \tag{5.1}$$

The probability of the current word is directly conditioned on the $(N-1)$ classes of the predecessor words. The independence assumption between the current word being predicted and the predecessor word classes that was present in the two-sided class model has effectively been removed.

It should be noted that there is no constraint on which words in the history are mapped nor how they are mapped, so long as the mappings are consistent. Combinations of classes and words can easily be incorporated into this structure and the generation of independent class mapping functions for each position in the history can also be accommodated. The word N-gram model still forms a particular case of the above class model when each word is mapped to its own unique class.

Since the one-sided model is capturing different language dependencies to the two-sided model, there is no guarantee that the performance benefits that were observed with the two-sided model will be obtained with the one-sided model. Given that the structure of the one-sided model lies somewhere between that of the word model and the two-sided class model, the characteristics of the one-sided model may also be expected to lie somewhere between those of the two models.

5.2 Automatic word clustering experiments

This section describes the details of the implementation and the experimental work using one-sided class models.

5.2.1 Exchange algorithm for the one-sided class model

The exchange algorithm that was presented in the previous chapter can be modified to determine the word classification function for the one-sided class model of Equation (5.1). The *exchange* operation is still an integral part of the algorithm and words are moved systematically among the available classes to determine the best class for each word. Similarly, when a word is moved from one class to another, the optimisation criterion can be computed efficiently by only updating contributions which are affected by the word movement. A greedy approach is likewise employed to make local optimisations by taking each word in turn and finding the best class among those available to which to move it. The best class is chosen with no regard for subsequent word classifications, hence the algorithm is not guaranteed to converge to a globally optimal solution.

In an analogous manner to the directed optimisation technique presented in the previous chapter, the optimisation criterion is defined as the log-likelihood of the training text using a one-sided class bigram model,

$$LL_{bi}(C_1) = \sum_{i=1}^{N_W} \log P(w_i \mid C_1(w_{i-1})). \tag{5.2}$$

Insertion of the maximum likelihood estimates of the conditional probabilities results in the following optimisation equation:

$$LL_{bi}(C_1) = \sum_{c_j} \sum_{w_i} N(c_j, w_i) \cdot \log \frac{N(c_j, w_i)}{N(c_j)}, \tag{5.3}$$

which can be further simplified for implementation to

$$LL_{bi}(C_1) = \sum_{c_j}\sum_{w_i} N(c_j, w_i) \cdot \log N(c_j, w_i) - \sum_{c_j} N(c_j) \cdot \log N(c_j). \tag{5.4}$$

5.2.1.1 Update equations

When a word w_i is tentatively moved from class c_j to class c_k it is only necessary to recompute the change in log-likelihood as a result of the move. Moreover, only the contribution of those counts which are affected by the movement of w_i need to be updated i.e. only those stored bigram counts in which w_i appears. The count update equations are as follows:

$$\forall w : N(c_j, w) = N(c_j, w) - N(w_i, w), \tag{5.5}$$

$$\forall w : N(c_k, w) = N(c_k, w) + N(w_i, w), \tag{5.6}$$

$$N(c_j) = N(c_j) - N(w_i), \tag{5.7}$$

$$N(c_k) = N(c_k) + N(w_i), \tag{5.8}$$

where, as before, for example

$$N(c_k, w) = \sum_{\forall i : w_i \in c_k} N(w_i, w). \tag{5.9}$$

There is no explicit search involved in computing the count updates since only those counts for words which follow w_i in the training data need to be changed and these can be indexed directly. The contribution to the log-likelihood of w_i being in class c_j need only be computed once. Thereafter, the contribution to the log-likelihood of w_i being in all remaining classes c_k can be computed. When w_i is moved, there is no “knock-on” effect[1] on class bigram counts, other than those involving c_k, hence the change in log-likelihood can effectively be computed for all classes simultaneously. Word w_i is then moved to the class c_k, for which the increase in log-likelihood is the greatest.

5.2.1.2 Computational complexity

Each iteration I of the algorithm involves finding the locally optimal class for the N_V vocabulary words by moving each word in turn to each of the N_C classes. When word w_i is moved to a tentative destination class, $(\frac{B}{N_V}+1)$ count updates involving a $\log$ operation must be performed, where $\frac{B}{N_V}$ is the average number of distinct bigrams per word i.e. all $N(w_i, w)$ in which w_i appears[2]. The complexity of the algorithm is therefore linear in the number of classes and linear in the number of words in the vocabulary:

[1] This was the case for the two-sided clustering operation since the class bigram counts $N(c_k, c)$ and $N(c, c_k)$ for all c are affected by moving a word; for the one-sided algorithm, only $N(c_k, w)$ is affected.

[2] The number of distinct bigrams B in the training data is affected by the size of the training data N_W and the size of the vocabulary N_V. As an example of the absolute values that are involved, for the 65k Russian vocabulary $B \approx 12 \times 10^6$ and for the 430k vocabulary $B \approx 20 \times 10^6$.

$$\mathcal{O}(I \cdot N_V \cdot N_C \cdot (\frac{B}{N_V} + 1)). \tag{5.10}$$

Compared to Equation (4.21) this complexity highlights a significant advantage of this algorithm over that for the two-sided model, specifically because the algorithm no longer scales quadratically in the number of classes. Correspondingly larger vocabularies may now be clustered into larger numbers of classes in a fraction of the time taken by the two-sided clustering algorithm.

5.2.1.3 Model parameters

The number of free parameters in the one-sided class bigram model is $N_V \cdot (N_C - 1)$, i.e. the number of free parameters lies between that of the two-sided class model for the same number of classes, and that of the word model.

The storage of separate N-gram ($N = 1, 2...$) tables is necessary for models of the form given by Equation (5.1)[3]. Where the number of parameters is reported in the tables below, this refers to the sum of the number of N-grams ($N = 1, 2...$) stored in each model. This value can be used as a valid comparison with the other models since it takes into account the number of parameters that are used to compute probabilities even if it does not accurately reflect the amount of storage space required by the model.

5.2.1.4 Efficient code implementation

The code for this algorithm was implemented in a different manner to that of the two-sided exchange algorithm since it was observed that a linked-list implementation would remove the necessity for any repetitive search operations and because it allowed the most efficient storage of the bigram counts $N(c, w)$. Whereas the bigram counts for the two-sided clustering algorithm were easily accommodated in a $N_C \times N_C$ array for the range of values of N_C that were investigated, for the one-sided clustering algorithm, the storage of the bigram counts would require a $N_V \times N_C$ array which is prohibitive for all but small values of N_V and N_C. A linked-list implementation therefore offered a memory efficient solution to the problem.

Separate linked lists are maintained for each class; each element in a class list c contains the count of a word w that follows class c in the training data i.e. $N(c, w)$. In addition, separate linked lists are constructed for each vocabulary word w and these are interlaced with the class lists to connect together all following word counts with the same identifier. The purpose of this is to facilitate the computation of the new optimisation criterion value when a word is tentatively moved from one class to another class. When a word w_i is tentatively moved to all possible destination classes c_k, the word list associated with each word w that follows w_i in the training data is traversed, and the count updates are computed for all classes c_k simultaneously. In this way, no searching for counts is necessary.

The drawback with this implementation concerns the removal and addition of new elements to the class lists whenever a word is taken out of its original class and placed in the best class

[3]The reason for this requirement was explained in Section 3.1 regarding the language modelling tools.

because some slow linear search operations are then necessary. However, this can be tolerated since it is only performed N_V times per iteration. No garbage collection is performed to remove "empty" elements from the linked lists which may also add a small overhead to the update calculations. Other performance issues concerning computer memory *caching* might also be of importance if the interlaced linked lists were to become significantly fragmented in memory.

5.2.2 Position dependent word classification

The one-sided class model given by Equation (5.1) suggests that different, independent classification functions are possible for each of the $(N-1)$ words in the history[4]. In the literature, classification methods which use the word bigram statistics from the training data to determine the word classification generally use the same classification for words in all positions of the history. This assumption implies that the co-occurrence statistics of adjacent words are the same as for pairs of words which are separated by any number of words. This is probably not a sensible assumption given that the class mapping function can only map words into one class. It is a plausible conjecture that words may have different parts of speech or different meanings when they appear in different relative positions to other words.

Since it has been shown that the clustering algorithm with the optimisation criterion given by Equation (5.4) is linear in the number of classes, it is still practical to perform an additional clustering operation using the distance bigram statistics from the training data. In particular, for the case of a one-sided class trigram model, the bigram counts of pairs of words which are separated by one word can be used to obtain a classification for the $(i-2)$th word in the N-gram as follows[5]:

$$LL_{bi}(C_2) = \sum_{i=1}^{N_W} \log P(w_i \mid C_2(w_{i-2})). \tag{5.11}$$

An analogous set of update equations to those developed in Section 5.2.1.1 can then be used by the exchange algorithm.

5.2.3 Experimental procedure

The experimental procedure employed for evaluating the one-sided class models is essentially identical to that used for evaluating the two-sided models in the previous chapter.

[4] A word classification function may be independently specified to map each word to its own unique class i.e. to be mapped to itself. Combinations of words and word classes may then be used in the context. This was investigated for a one-sided class trigram model for both possible combinations of words and word classes. The results, however, were worse than for the word model by itself and the beneficial properties of the class models were also absent so this combination was not pursued further.

[5] Optimising both class mapping functions simultaneously would be expected to give superior results, however, this would be at the expense of a significant increase in algorithmic complexity and clustering times.

The first step was to generate a set of word classifications for the vocabulary of each corpus for different numbers of word equivalence classes. To facilitate a comparison with the performance of the two-sided model, the 65k vocabularies for both corpora were clustered into 204, 504, 1004, 2004, 3004, 4004 and 5004 classes[6]. This range of classes allows an approximate relationship between the number of classes and the perplexity of the models to be determined. In addition, the 430k Russian vocabulary was clustered into the same range of classes, since this is now feasible due to the improved scaling properties of the one-sided clustering algorithm. The same initialisation scheme as before was used, whereby the most frequent $(N_C - 1)$ words are assigned to their own unique class and all remaining words placed in the N_Cth class.

The second step was to build one-sided class trigram models for each of the seven different numbers of classes using the `training` data from each corpus. All singleton bigrams and trigrams were discarded and Good-Turing discounting employed together with Katz's backing-off scheme. The performance of each class model was assessed by computing its perplexity on the `eval-test` portion of each corpus. Interpolated word and class models were also evaluated and the interpolation weights found using an E-M algorithm to optimise the perplexity of the model on the `dev-test` portion of each corpus.

For all the above class trigram models, a position-dependent word classification (see Section 5.2.2) was also obtained for each of the two predecessor words and class models then built and evaluated in an identical manner.

An aside: combinations of classifications

Since it was known in advance that the clustering operation for the one-sided class model would be fast, it seemed obvious to use the classifications obtained for the one-sided model in the two-sided model. The motivation behind this strategy was that since the two-sided model captures more general dependencies than the one-sided model, the advantages of both models should perhaps be combined. However, the performance of the two-sided model with the one-sided model's classifications was found to be substantially worse than both the one-sided model with one-sided classifications and the two-sided model with two-sided classifications. This was true both for the stand-alone model and when it was interpolated with the word model. Consequently, this line of experimentation was not pursued further and the results will not be given in the next section. Nonetheless, the conclusion that was drawn from this experiment was that it is important to optimise the word classifications directly for the type of model they will be used in. In particular, this is relevant to the use of part-of-speech classifications in class models since they are inherently predetermined and not optimised for the model they are used in. This observation corroborates the results that were obtained in the previous chapter in Section 4.4.

[6]As before, the additional 4 classes contain symbols which are not considered for clustering and to which other words cannot be moved. See Section 4.3.2 for a more detailed explanation.

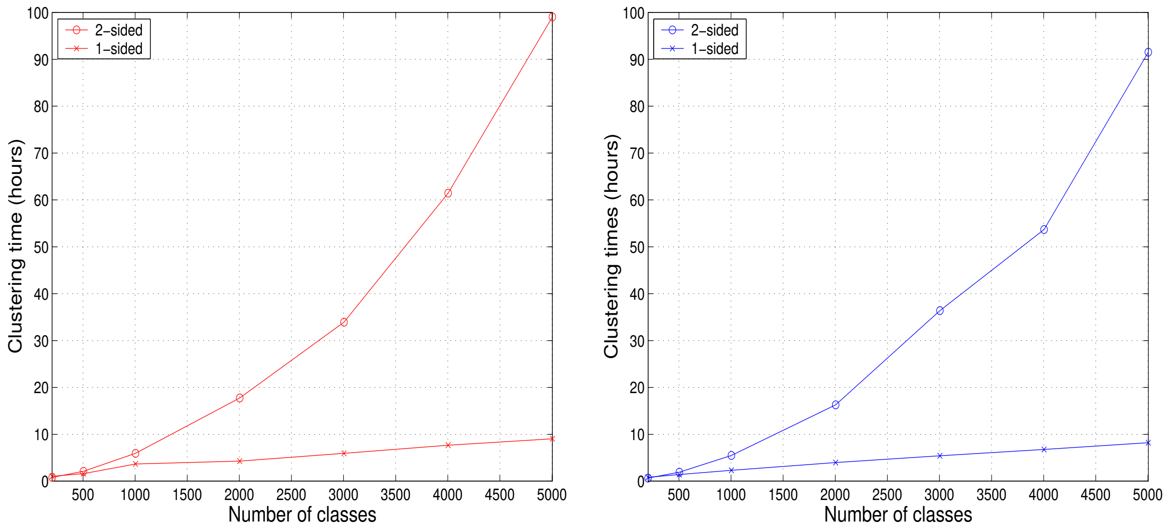
5.2.4 Results

The clustering times for the one-sided algorithm are presented first, followed by the perplexity results for one-sided class models using both the 65k and 430k Russian vocabularies and the 65k English vocabulary.

Number of classes	Hours per iteration		
	Russian corpus		English corpus
	65k	430k	65k
204	1.0	10.1	0.8
504	1.6	11.2	1.4
1004	3.7	13.7	2.3
2004	4.3	15.8	4.0
3004	5.9	18.3	5.4
4004	7.7	20.9	6.8
5004	9.1	23.3	8.2

Table 5.1 *Hours per iteration for a range of different numbers of classes and vocabulary sizes.*

The times taken to perform one iteration of the clustering algorithm using the 65k and 430k Russian and 65k English vocabularies for the seven different numbers of classes are given in Table 5.1. A comparison of the clustering times per iteration between the one-sided and two-sided clustering algorithms for the same range of classes is plotted in Figure 5.1 for the 65k vocabularies only.



(a) Russian corpus.

(b) English corpus.

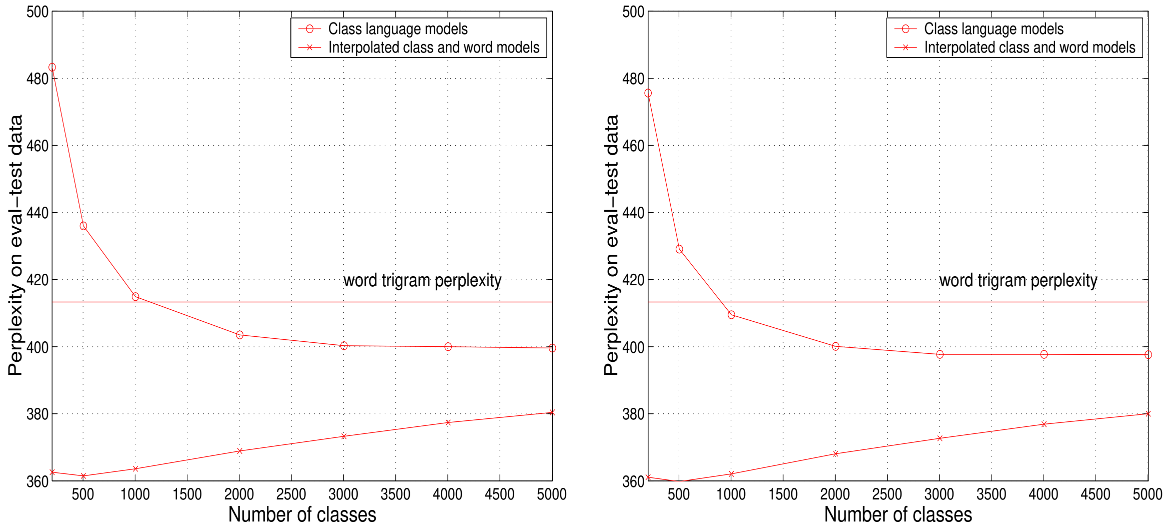
Figure 5.1 *Comparison of clustering times per iteration for 1-sided and 2-sided models with a 65k vocabulary.*

To reduce clutter in the presentation of the results, only the perplexity results for the one-sided class models with the position-independent word classifications have been displayed in the tables of results. However, to allow some comparison to be made, graphs are also provided to show the perplexity variation between models with position-dependent and position-independent word classifications.

No. of classes	Perplexity		weights	% improvement	Class model size
	PP_{class}	PP_{interp}	λ_{word}, λ_{class}	over word trigram	(parameters)
204	483.3	362.6	0.65, 0.35	12.3	7,492,300
504	436.0	361.5	0.60, 0.40	12.5	8,759,640
1004	414.9	363.6	0.58, 0.42	12.0	9,542,270
2004	403.5	368.9	0.56, 0.44	10.7	10,093,100
3004	400.3	373.3	0.55, 0.45	9.7	10,354,210
4004	400.0	377.4	0.55, 0.45	8.7	10,394,830
5004	399.6	380.4	0.54, 0.46	8.0	10,458,490
65000	413.3	—	—	—	10,896,660

Table 5.2 *Russian corpus (65k): perplexity on* `eval-test` *data of stand-alone 1-sided class trigram models and interpolated class and word trigram models (position independent classification).*

The perplexities of one-sided, position independent class models for the Russian corpus with the 65k vocabulary are shown in Table 5.2. A comparison of the perplexities obtained with the position-independent and position-dependent classifications for the same vocabulary is plotted in Figure 5.2.



(a) Position independent classification.

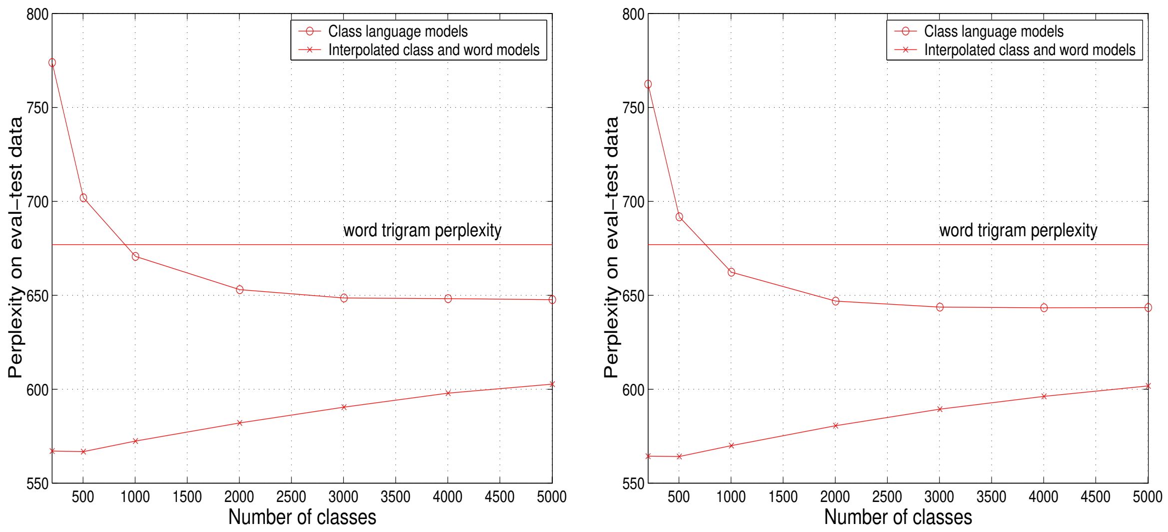
(b) Position dependent classification.

Figure 5.2 *Russian corpus (65k): perplexity on* `eval-test` *data of stand-alone 1-sided class trigram models and interpolated class and word trigram models.*

The perplexities of one-sided, position independent class models for the Russian corpus with the 430k vocabulary are shown in Table 5.3. A comparison of the perplexities obtained with the position-independent and position-dependent classifications for the same vocabulary is plotted in Figure 5.3.

No. of classes	Perplexity		weights	% improvement	Class model size
	PP_{class}	PP_{interp}	$\lambda_{word}, \lambda_{class}$	over word trigram	(parameters)
204	773.9	567.1	0.62, 0.38	16.2	9,562,340
504	701.9	566.8	0.58, 0.42	16.2	10,657,620
1004	670.7	572.4	0.56, 0.44	15.5	11,316,050
2004	653.0	582.1	0.55, 0.45	14.0	11,755,510
3004	648.6	590.5	0.54, 0.46	12.8	11,895,130
4004	648.2	597.9	0.54, 0.46	11.7	11,958,400
5004	647.7	602.7	0.54, 0.46	11.0	11,989,020
430000	677.0	—	—	—	12,177,700

Table 5.3 *Russian corpus (430k): perplexity on* `eval-test` *data of stand-alone 1-sided class trigram models alone and interpolated class and word trigram models (position independent classification).*



(a) Position independent classification.

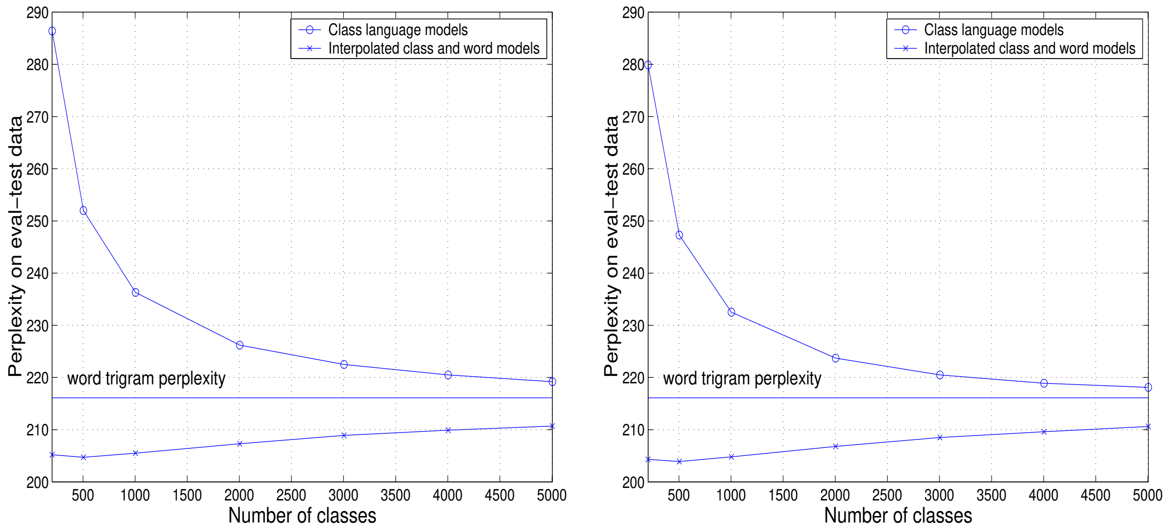
(b) Position dependent classification.

Figure 5.3 *Russian corpus (430k): perplexity on* `eval-test` *data of stand-alone 1-sided class trigram models and interpolated class and word trigram models.*

Perplexity results for the 65k English vocabulary are shown in Table 5.4 and a comparative plot of the position-independent and position-dependent classifications is given in Figure 5.4.

No. of classes	Perplexity PP_{class}	Perplexity PP_{interp}	weights $\lambda_{word}, \lambda_{class}$	% improvement over word trigram	Class model size (parameters)
204	286.4	205.2	0.75, 0.25	5.0	7,851,320
504	252.0	204.7	0.70, 0.30	5.3	9,757,910
1004	236.3	205.5	0.67, 0.33	4.9	10,912,120
2004	226.2	207.3	0.65, 0.35	4.1	11,640,630
3004	222.5	208.9	0.64, 0.36	3.3	11,891,270
4004	220.5	209.9	0.63, 0.37	2.9	12,017,590
5004	219.2	210.7	0.62, 0.38	2.5	12,097,360
65000	216.1	—	—	—	12,431,060

Table 5.4 *English corpus (65k): perplexity on* `eval-test` *data of stand-alone 1-sided class trigram models and interpolated class and word trigram models (position independent classification).*



(a) Position independent classification.

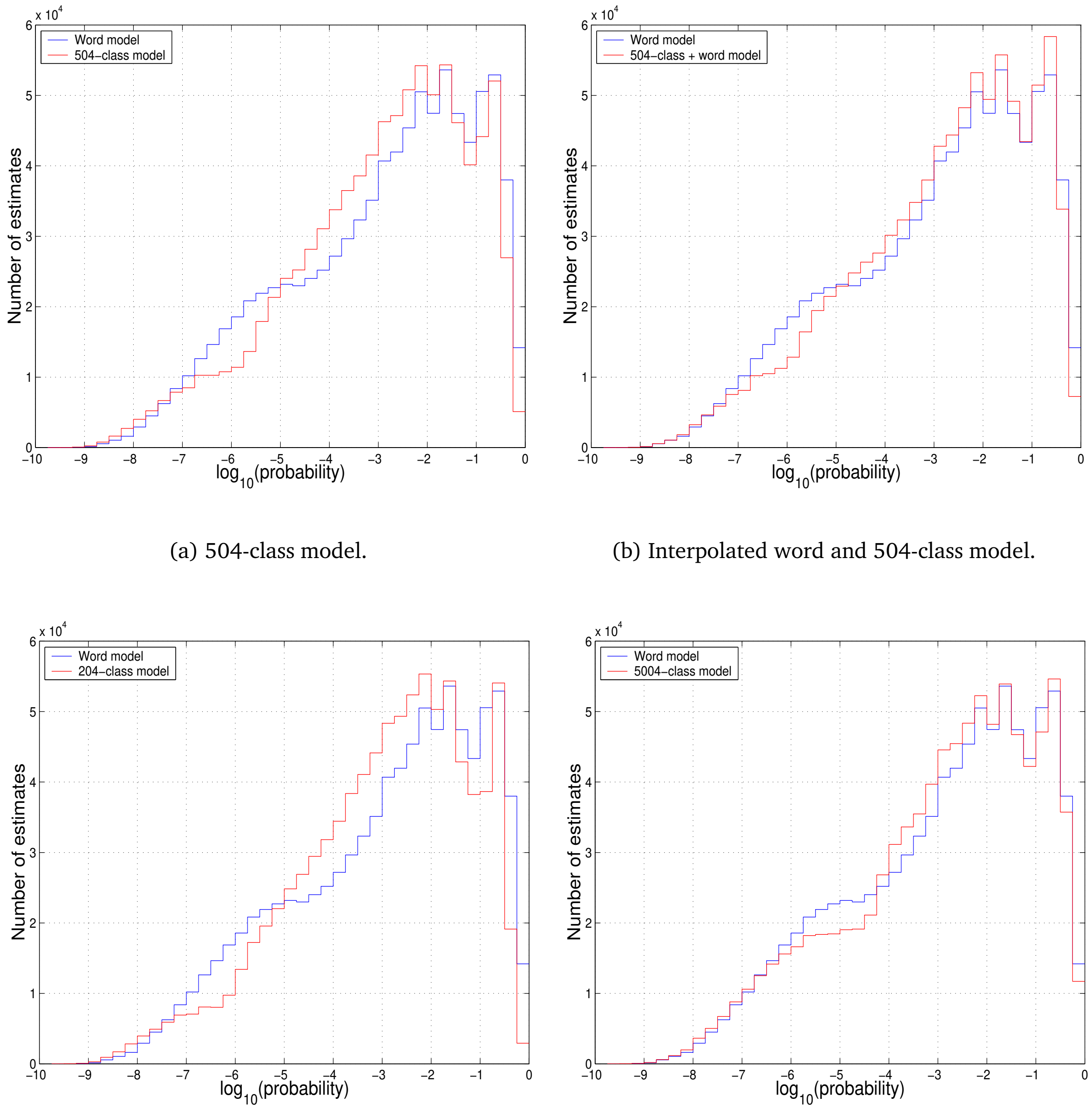
(b) Position dependent classification.

Figure 5.4 *English corpus (65k): perplexity on* `eval-test` *data of stand-alone 1-sided class trigram models and interpolated class and word trigram models.*

5.2.5 Analysis of model performance

One-sided class models have now been built for both corpora using seven different numbers of classes. In order to analyse the properties of the one-sided class model and assess its interaction with the word model when it is interpolated with it, only the best interpolated model on the

Russian corpus using the 430k vocabulary will be examined, together with the performance of the best and the worst stand-alone class models. The same analysis techniques that were applied in the previous chapter will also be used here.



(a) 504-class model.

(b) Interpolated word and 504-class model.

(c) 204-class model.

(d) 5004-class model.

Figure 5.5 *Russian corpus (430k): distribution of log-probabilities for trigram word and class models.*

The distributions of the log-probabilities assigned by the 204, 504 and 5004-class models and the interpolated word and 504-class model are shown in Figure 5.5 for the Russian `eval-test` data. A noticeably different apportioning of probabilities is obtained, compared to that of the two-sided models in the previous chapter, cf. Figure 4.3. One major difference is that the one-sided class models apportion a higher number of very low probabilities ($< 10^{-7}$) than the word model does which was not the case for the two-sided models. (Note that, although not shown here, this was also found to be the case for the one-sided models based on the 65k vocabulary.) Whenever the class model backs off, the word model must also back off. For these cases, it was found that the probability assigned by the class model was generally lower than that assigned by the word model. When the class model backs off to its unigram distribution (note: this is the word unigram distribution in the one-sided class model), the majority of class model probabilities tend to be lower than the word model's probabilities. This is directly related to the back-off weights in the class model which are smaller on average. Words occurring in a particular word context are generally more sparse than words occurring in the corresponding class context, hence for the word model the back-off weight (which can be thought of as determining the significance of the lower-order distributions) tends to be higher.

The distribution of log-probabilities for the interpolated model exhibits a large peak for high probabilities (> 0.1) and the number of very low probabilities is similar to that of the word model's. It will be remembered that for the interpolation with the two-sided model, the number of very low probabilities was much less than for either of the component models so this is a major difference between the characteristics of the interpolated word and one-sided and word and two-sided class models. The distribution of log-probabilities for the one-sided 204-class model appears to be skewed to the left with a peak around probabilities of 0.01. As the number of classes is increased, the distribution moves progressively to the right, up to the point where for the one-sided 5004-class model the distribution of all probabilities becomes very similar to the word model's. The slightly lower perplexity of the 5004-class model can be attributed mainly to the different apportioning of probabilities between 10^{-6} and 10^{-2} by the two models.

	Perplexity of different models			
Back-off case	204-class	504-class	5004-class	430k word
3	111	91.8	71.5	66.4
3-2	624	551	462	429
3-2-1	6.46×10^4	7.34×10^4	1.13×10^5	1.82×10^5

Table 5.5 *Russian corpus (430k): perplexities of several models computed on sets of events that were predicted by different word model back-off cases.*

In Table 5.5 the perplexities of three different class models and the word model for the 430k Russian vocabulary are computed in an identical manner to that used in Section 4.3.4 for the three[7] different backoff cases of the word model, i.e. when the word trigram model finds the requested trigram (3), or backs-off to either the bigram (3-2) or unigram (3-2-1) distribution. Clearly the word model does not perform well when it backs off to its unigram distribution, but compensates for the poor probability estimates by assigning relatively high probabilities when it locates the trigram or only backs off to the word bigram. As the number of classes is increased, the perplexity of the class models tends to decrease for cases where the word

[7]The bigram request at the beginning of the `eval-test` text has been ignored.

model does not back-off and tends to increase for cases where the word model backs off to the unigram distribution i.e. the characteristics of the class model become more similar to those of the word model, as is expected. The lower perplexities of the class model, when the word model backs-off completely, are directly related to the class model's ability to generalise to previously unseen word sequences. Unseen word N-grams that are poorly estimated by the word model are generally better modelled by the class N-gram.

When a position-dependent classification is used instead of the position-independent classification for all word history positions, there is, on average a 1% relative improvement in perplexity. There is also a corresponding improvement in the interpolated perplexity figures. Although it would be instructive to determine the reason for this improvement, no conclusive explanation was found. The contents of the classes for words with multiple parts of speech, e.g. "LIGHT", were examined, yet no satisfactory attribute distinguished the two classifications. For the sake of interest, several word trigrams in which the third word was assigned a much higher (1000$\times$) probability by the English 5004-class model with the position-dependent classification than by the position-independent model, are shown in Table 5.6. It is clear that the words in each sequence are strongly correlated, moreover, the individual words are not the only members of their respective classes i.e. they are not effectively word trigrams. However, no conclusion may be drawn from this observation, since very similar sequences were also obtained for the position-independent model predicting words with a much higher probability than the position-dependent model (not shown here).

CEYLON AND BURMA
MICROSOFT IS PRICING
MANUFACTURERS AND EXPORTERS
AZALEAS AND RHODODENDRONS
CASSEROLE TO TASTE
JOURNALIST AND CRITIC
SNEAKERS AND SWEATSHIRTS
EARRINGS AND BANGLES
ANALYTICAL AND TECHNICAL
WRIT OF MANDAMUS
UNWORTHY OF INCLUSION
FORDS OR BRIDGES
RUFFLED THE FEATHERS
OMF I ELEVEN
CATARRH AND SNEEZING
PACKS OF WOLVES
TAIL HELD ERECT
GREENHOUSE OR GARDEN

Table 5.6 *Word trigrams in which the third word is assigned a probability one thousand times greater by the position-dependent 5004-class model than by the position independent 5004-class model.*

It is interesting to note that the stand-alone one-sided and two-sided class models perform equally well when they have similar numbers of parameters. Using the Russian 65k vocabulary, the perplexity of the 2004-class two-sided model on the training data is around 4% higher than the 504-class one-sided model even though the models' perplexities on the `eval-test` data are almost identical. This further implies that the two-sided model has a greater capacity for generalisation than the one-sided model and also that the one-sided model has captured more of the idiosyncrasies of the training data. Such dependencies will already be well estimated by the word model and this may explain why the improvement that is obtained when the one-sided class model is interpolated with the word model is less than for the interpolation with the two-sided class model.

Class 1	**Class 2**	**Class 3**	**Class 4**	**Class 5**
APPOINTED	MEETING	ATTENTION		
ELECTED	CONFERENCE	APPEAL		
EXPERIENCED	CONGRESS	DUTY	OTHER	
EMPLOYED	ASSEMBLY	EFFORT	FEWER	FELT
TRAINED	SESSION	REACTION	PRECATORY	SOUNDED
REGISTERED	SUMMIT	RESISTANCE	OVVER	
RETIRED	CONCERT	CONSENT	AZANIAN	
QUALIFIED	LECTURE	ADMISSION	WEIGHTIER	
DISTINGUISHED	CEREMONY	COURAGE		
RESPECTED	RALLY	LOYALTY		
Class 6	**Class 7**	**Class 8**	**Class 9**	**Class 10**
BIT	BIG	JOINT	ORIGINALLY	WIND
MITE	GIANT	FORMAL	NEWLY	RAIN
CLOVE	SUPER	DETAILED	FORMALLY	SMOKE
WAFER	MIGHTY	EXTENSIVE	SPECIALLY	SNOW
UNPLEASANTLY	GIGANTIC	COMPREHENSIVE	OFFICIALLY	SAND
TAD	MEGA	INFORMAL	BEAUTIFULLY	DUST
BURBLE	RUNAWAY	INTENSIVE	LEGALLY	STORM
GLUG	MAMMOTH	THOROUGH	POORLY	MUD
INSULTINGLY	VERITABLE	FULLER	DULY	SWEAT
	PROVERBIAL	AUTHORITATIVE	FINELY	BREEZE

Table 5.7 *All, or up to the ten most frequent, words from ten randomly chosen classes of the 1004-class English model.*

In Table 5.7 the contents are presented of ten randomly chosen classes that were taken from the English one-sided 1004-class classifications. It is readily apparent that most of the words in a particular class share some attribute in common, be it semantic or syntactic. However, there are also classes for which an adequate explanation cannot easily be found, notably classes 4 and 5. A comparison of the kinds of classes obtained with the two-sided model in Table 4.5 shows that similar word groupings are obtained with the one-sided model. Despite these apparent similarities, it is worth repeating that the classifications optimised for one particular class model are generally unsuitable for use in other types of class model.

5.2.6 Discussion

The main advantage of the one-sided class model was identified near the beginning of this chapter as the improved scaling properties of its clustering algorithm. It is clear from the graphs of the clustering times in Figure 5.1 that the algorithm does indeed scale linearly with the number of classes and is significantly faster than the two-sided clustering algorithm. Moreover, in Table 5.1 it is seen that this is also true for the two different Russian vocabulary sizes where the only difference is an offset to the total clustering time for the larger vocabulary size. This offset is partly explained by the complexity of the algorithm given by Equation (5.10): the scaling of the algorithm depends on $(N_V \cdot (\frac{B}{N_V} + 1))$ which for the experiments described here approximates to B. As N_V is increased, B also increases, but much less than linearly and so ultimately this has little additional effect on the way the algorithm scales. Much more significant is the apparent offset in the clustering times for the larger vocabulary. A profile of the software implementation showed that this was directly due to the linked-list implementation used to store the counts $N(c, w)$. On average, $\frac{B}{N_V}$ removal and insertion operations are required when a word is moved between classes and this operation involves a linear search to replace and insert elements from and into a linked-list[8]. The larger the vocabulary is, the longer each list tends to be. This effect is more apparent for smaller values of N_C since more words occur in each class context (the classes are more frequent) and hence the lists are longer.

The perplexity of the best interpolation of the one-sided class model with the word model was slightly less (between 2.8% and 4.5%) than the perplexity of the best interpolation obtained with the two-sided class model (for the 65k English and 65k Russian vocabularies respectively). It may be inferred that the two-sided class model is better able to generalise to unseen word sequences since only N-grams of classes are captured by the two-sided model. In contrast, the one-sided model captures N-grams of $(N - 1)$ classes plus one word, which will inevitably be more sparse, and consequently the one-sided class model's ability to generalise to unseen word sequences will be poorer than the two-sided class model's. Depending on the quantity of training data available and the vocabulary size used, this tradeoff may well be acceptable.

The same trend that was observed with the stand-alone, two-sided class model of the perplexity decreasing as the number of classes increases is also observed with the stand-alone, one-sided class model, although again this will only be true up to a certain point. An optimum number of classes for the Russian corpus can again be expected to lie somewhere between 4004 and 65000 classes for example. It is also clear that increasing the number of classes much beyond 5004 will not reduce the perplexity significantly since the perplexities of class models with more than 3004 classes have more or less converged anyway. For the English corpus it appears that the word model may well outperform all possible class models although further experimentation would be necessary to confirm this. As was noted in the previous chapter, some of the class models built on the Russian corpus were able to compensate better than the word model for the increased data sparsity of the corpus. It may therefore be concluded that the stand-alone, one-sided class models are easily able to compete with the two-sided class models and even the word models in terms of their perplexity and the number of model parameters.

Furthermore, if the language concerned is highly inflected and requires a very large vocabulary to achieve a usable OOV-rate then the one-sided class model represents a very effective solution. The clustering operation for the Russian 430k vocabulary was performed in a relatively short time and the perplexity reduction of the combined word and one-sided class model

[8]It will be remembered that this implementation was necessary to conserve memory usage, however, for the 204-class model, these operations accounted for 40% of the total clustering time of the algorithm.

was 16.2%. These factors recommend the one-sided class model as a particularly efficient means of building a language model in situations where large vocabularies are required.

5.3 Summary

This chapter has examined the one-sided class model and its properties as a language model. A comparison has also been made of its performance against the two-sided class model investigated on the Russian and English corpora. The one-sided and two-sided stand-alone class models were shown to perform similarly well on both corpora. On the Russian corpus with the 430k vocabulary, the class model was able to outperform the word model by 4.3% while still employing slightly fewer parameters. The performance of the best interpolated word and one-sided class models was shown to be slightly inferior to that of the best interpolated word and two-sided class model. This was attributed to the poorer generalisation ability of the one-sided class model.

The use of separate word classification functions, which were optimised for each position of the word in the conditioning context, was shown to give a small but appreciable reduction in perplexity (up to 1.5%) over using the same classification function for all positions in the word history.

The clustering algorithm for the one-sided class model was shown to be significantly faster than that for the two-sided class model. This was an important consideration since it allowed the very large 430k Russian vocabulary to be classified into a large number of classes in a computationally acceptable time. In conclusion, a much faster and similarly effective means of building stand-alone class models has been developed.

6

Particle-based language modelling

This chapter examines a range of techniques which model language at a sub-word level. The term *particle* has been chosen in preference to *sub-word* unit to denote any possible representation of part of a word, be it a single character or a whole word. This representation might also include characters which are not letters, for example punctuation marks or even the phonetic transcription of part of a word.

The motivation for modelling language at a sub-word level using particles is first examined, followed by an overview of several techniques in the literature that may be used for obtaining particles. The development is then presented of one linguistically motivated and two novel, data-driven techniques that determine particle units and word decompositions automatically. The results of perplexity experiments are reported for several different particle 6-gram models and their characteristics and performance are then analysed.

6.1 Introduction

Most language modelling approaches use words as the fundamental modelling units or classes of words like those investigated in the previous two chapters. Although words are a logical choice, since it is ultimately words that are to be output by a speech recognition system, they are not necessarily the best units for capturing dependencies in a text. The optimal set of units will inevitably depend on the language, the sparsity of the training data and the model into which the units are incorporated.

In chapter 3, the necessity was highlighted for an alternative approach to word-based language modelling of Russian. In particular, the very large vocabulary size that is required to achieve a usable OOV-rate with Russian has clearly shown that modelling at the word level is not necessarily the best solution. It was pointed out in the introduction (see Section 1.3) that Russian words often have many morphological units in common. This is particularly evident in the case of Russian word inflections, which are appended to word stems to denote the word's grammatical case, gender and number. Given this characteristic of the Russian language, there is a strong argument for decomposing words into units, smaller than the word themselves, and using these in some form of language model. Unfortunately, there is no satisfactory, rule-based method available for providing a linguistically accurate, morphological decomposition of all the Russian words that might ever be encountered. It was decided, therefore, to investigate several techniques for the automatic generation of a deterministic decomposition function U of words into a set Ψ of N_Ψ particles u_i

$$U : w \rightarrow U(w) = u_0, u_1, \ldots, u_{L(w)-1} \quad u_i \in \Psi.$$

where word w is decomposed into $L(w)$ particles.

A <w> symbol is always attached to the terminal particle $u_{L(w)-1}$ in the decomposition of word w to denote a word boundary. Identification of word boundaries at the particle level is necessary to ensure a deterministic mapping from a stream of particles back to the word-level. A summation over all the possible ways in which the word stream could be obtained from the particle stream would otherwise be necessary. This is generally undesirable when the language model is to be incorporated into a speech recogniser.

A further consequence of this requirement is that the relationship between some morphological strings is inevitably lost. As an example, for the two words "SATISFY" and "SATISFYING", if "SATISFY", "SATISFY<w>" and "ING<w>" are determined to be particles, the connection between the two words is lost. An alternative scheme would be to separate <w> from the terminal particle, however, this would reduce the effective context of the particle N-gram model that will be used and unduly complicate the automatic algorithms which are described later. From the point of view of language modelling, the inclusion of the word boundary marker may be seen as incorporating additional linguistic information. Strings which appear at the ends of words cannot be confused with identical strings which appear in other word positions.

Since the identity of a morpheme is distorted by the environment in which it occurs, it is expected that not all *allomorphs* of a morpheme will be determined correctly by a purely statistical method. Even rule-based decomposition methods must take into account the spelling changes that occur at word boundaries. It may in fact be preferable, from a language modelling point of view, to obtain a decomposition of words into uninflected word stems and inflections, rather than into constituent *morphs*. The objective of the two data-driven algorithms presented in this chapter is to determine the particle units and word decompositions that best model the training data.

Once the decomposition function U has been determined, words in the training text can be decomposed and a particle N-gram language model built:

$$P(w_1, \ldots, w_n) \approx \prod_{i=1}^{N_P} P(u_i \mid u_{i-N+1}, \ldots, u_{i-1}). \tag{6.1}$$

where each word w in the text is decomposed using $U(w)$ into its constituent particles and N_P is the total number of particles in the text when all words have been decomposed. Relative frequencies of the occurrences of particle N-tuples can be used to compute the above conditional particle probabilities and then smoothed in a manner similar to that described for conditional word probabilities in Section 2.3.

In this chapter, it is assumed that it is ultimately words which are to be recognised and therefore all subsequent discussion will only concern word level probabilities and perplexities. It must be emphasised that the particle units under discussion are not being proposed as sub-word units for recognition, only as language modelling units. The word-level probability of a word w_n given some particle context $h = U(w_1) \ldots U(w_{n-1})$ can be computed using the following particle bigram model:

$$P(w_n \mid h) = \frac{1}{Z(h)} \cdot P(u^{w_n}_{L(w_n)-1} \mid u^{w_n}_{L(w_n)-2}) \cdot P(u^{w_n}_{L(w_n)-2} \mid u^{w_n}_{L(w_n)-3}) \cdots P(u^{w_n}_0 \mid u^{w_{n-1}}_{L(w_{n-1})-1}), \tag{6.2}$$

where w_n is decomposed into $L(w_n)$ individual particles $u^{w_n}_i$ $i = 0, \ldots, L(w_n) - 1$, and w_{n-1} is decomposed into $L(w_{n-1})$ individual particles $u^{w_{n-1}}_j$ $j = 0, \ldots, L(w_{n-1}) - 1$, and $Z(h)$ is a normalization constant. In particular, the bigram context for the first particle of word w_n (which is identical to h in the above example) is the terminal particle of the previous word: $u^{w_{n-1}}_{L(w_{n-1})}$. The normalization constant $Z(h) = \sum_{w_i} P(w_i \mid h)$ ensures a correct probability distribution at the word level. The remainder of the probability mass $(1 - Z(h))$ is accounted for by words that are not in the vocabulary, but for which probabilities could be generated by concatenating unused sequences of particles up to some maximum number of particles. This may be seen as a beneficial consequence of the particle modelling approach which permits the word-level vocabulary to be augmented with new words without the complete retraining of the particle language model. If all single letters and all single letters attached to the word boundary symbol were included in the particle vocabulary then a probability could be generated for all possible words.

It is desirable that the largest vocabulary possible should be used for determining sets of particles and word decompositions. This is because the majority of the richness of word inflections and commonality of word stems will only be exhibited by also including words that occur infrequently. Moreover, it is precisely these words that it is hoped will benefit from the particle modelling approach, since their statistics are often more poorly estimated by the word model due to their low frequency of occurrence.

Before reporting on the three different algorithms used in the experimental work in this chapter, several techniques that have been proposed in the literature for automatically obtaining sub-word units will be described.

6.2 Overview of existing techniques

In this section, a description is given of five techniques which appear in the literature and how they might be applied to the task of automatically determining particles and word decompositions. This overview, which is not intended to cover all the methods available, begins with elementary linguistics-based approaches and progresses through to data driven methods. Statistical language modelling was not the motivation for some of the techniques presented below, however their exposition is still instructive and they are included for the sake of interest. It is important to note that only two of the methods given here could be used for determining particles of the kind used later in this chapter; the reasons why the other techniques could not be used are given in the appropriate place.

6.2.1 Suffix-stripping

One obvious method of segmenting a word into sub-word units is to strip the known affixes from a word, wherever they appear in the word. A more acceptable approach must consider spelling changes at morpheme boundaries. A non-statistical approach to the problem of suffix stripping which has been used in the field of information retrieval is presented in (Porter, 1980). A list of suffixes and stripping rules is already assumed to exist and the problem becomes one of determining whether or not the end of a word is a proper suffix or not. The first step of the algorithm converts plurals and past participles into a singular form. Each of steps two, three and four convert words ending in each of a particular type of suffix to a normalized form. The final step involves tidying up all those forms, from which a suffix has been stripped, to make them consistent with each other. Certain heuristics are applied which prevent the removal of suffixes from words where the stem would be too short. Complex suffixes are successively stripped as the algorithm proceeds from one step to the next. In principle, this method could be used for any language given the existence of appropriate rules for stripping suffixes from words. However, such rules are not known to be available for Russian.

6.2.2 Morph decomposition in MITalk

A rule-based approach to the segmentation of words into morphs is employed in the MITalk speech synthesis system (Allen et al., 1987). The morphemes in a word are then used to construct a pronunciation of whole words. Such an approach removes the need for a pronunciation dictionary of all the words that may ever be input to the system.

The morph segmentation algorithm requires a morph lexicon from the outset. The algorithm proceeds by matching the right hand side of a word against the largest lexicon morph possible. When a match is found, the search for further matches is performed recursively on the remaining unmatched portion of the word. If the match fails, at any stage, the previous match is discarded and the next largest matching morph is used instead. Only those morphs which are legal in the current context are considered for matching. This is handled by a set of selectional rules. In addition, a score is used to rank the likelihoods of multiple legal segmentations. As with the previous method, the morph lexicon and selectional rules for a language must be available for this method to work and for Russian they have not been determined.

6.2.3 Statistical-combinatorial modelling

Possibly the first attempt of using a data-driven approach to determine linguistic units and inflectional paradigms appears in (Andreyev, 1965)[1]. Andreyev headed a team of around ten people in the Soviet Union during the 1960s. All data processing and computation were performed manually (Shaikevich, 1997) and experiments were conducted on a wide range of languages, including English and Russian, using technical texts of around a million words in size.

The first step in Andreyev's technique of *statistical-combinatorial modelling* was to rank the probabilities of all letters, conditioned on their relative position from the end or beginning of a word. Using the prior information that inflections occur at the ends of words, Andreyev first found the letter with the highest probability that appears in the word-final position. This single letter is then combined with the letter that appears with the highest probability in the word-penultimate position and adjacent to the previously found letter. This process is repeated successively with each of the next highest ranking letters until the desired set of inflections has been obtained.

6.2.4 Sticky pairs

One method that has been employed to determine language modelling units other than individual words is presented in (Brown et al., 1992). The method determines pairs of words using the mutual information metric, which is defined as the ratio of the probability of two words occurring together, to the product of their independent probabilities. The resulting units were termed *sticky pairs*. No perplexity measurements were quoted in (Brown et al., 1992) using these units, although the units that were obtained were intuitively appealing, with pairs like *Humpty Dumpty* and *Pontius Pilate* ranked highly as very *sticky*. With the original method, which does not directly optimise the perplexity of a text, it is unclear whether such units are indeed useful in language modelling. The derivation of similar word pairs on the German *Verbmobil* task by minimising the perplexity of a text have met with a certain amount of success in terms of word error rate improvements (Ries et al., 1996).

This method can easily be extended conceptually to determining particles: starting with single characters as the set of particles, the algorithm would successively combine each pair of particles that has the highest mutual information. In fact, this agglomerative approach forms the basis of one of the data-driven methods developed in Section 6.3.3. The fundamental difference is that particles are chosen according to how much they increase the likelihood of the training data rather than by the mutual information between the constituent characters in a particle.

[1] Andreyev's work is not well known in the West; the majority of his publications were written in Russian and never translated.

6.2.5 Multigrams

The multigram method of modelling language, described in (Deligne and Bimbot, 1995), was developed for "retrieving sequential variable-length regularities within streams of data." Such a model was observed to be particularly appealing for use with natural language corpora. The n-multigram was defined to be a sequence of up to n non-overlapping symbols which could, for example, be letters or phones[2]. Multigrams are also permitted to span word boundaries.

The multigram model that was originally developed assumed statistical independence between multigrams. Computation of the likelihood of some observation sequence O involves a summation over each possible segmentation U of the observation sequence into a sequence of n-multigrams:

$$LL_{uni}(O) = \sum_{U \in \{U\}} LL_{uni}(O, U). \tag{6.3}$$

Maximum likelihood estimates of the model parameters are obtained, using an E-M algorithm, as a maximum likelihood estimation from incomplete data (Dempster et al., 1977). The observed data is the string of symbols O and the unknown data is the underlying segmentation U. The parameters of the n-multigram model are initialised with the relative frequencies of all symbol co-occurrences of symbols up to length n in the training text. The parameters are iteratively re-estimated using a forward-backward algorithm or Viterbi alignment (Deligne and Bimbot, 1995) until there is no significant increase in training set likelihood, or alternatively until a fixed number of iterations has been executed. Pruning of infrequent multigrams is used to reduce the size of the particle set at each iteration and to reduce the complexity of the multigram model.

In (Deligne and Bimbot, 1997) the psalms of the King James Bible are used as the training text with all the spaces between words removed. The value of n is set to five and the segmentation is found using a Viterbi search. The results clearly show the morphological structure of the text and frequent single words and pairs of words are isolated by the algorithm as multigrams. The original independence assumption between sequences of multigrams is extended in (Deligne et al., 1996) to include N-grams of n-multigrams[3].

It is important to note that the multigram method finds the most probable segmentation of the whole training text, so identical words may be segmented into different sequences of multigrams depending on a word's position in the training text. This renders the technique inappropriate for use in the particle language model that has been proposed.

[2] In this respect the definition of a multigram is very similar to the definition of a *particle* given above, however, use of the term particle will be maintained since there are other connotations associated with multigrams which it is necessary to avoid.

[3] One further use of the multigram approach was the formulation of the joint multigram in (Deligne et al., 1995). This approach aims to determine a many-to-many alignment between the D strings in D streams of observable symbols, drawn from D distinct alphabets. The method is used for a 2-stream case to infer pronunciations for variable-length sequences of graphemes. This is achieved by finding the most likely joint segmentation to align variable-length sequences of phonemes with the graphemes.

6.3 Automatic methods for decomposing words

In a similar vein to several of the methods described above, a simple, pseudo-linguistic algorithm and two data-driven algorithms will now be described which automatically determine particles that reduce the word level perplexity of a particle bigram model on the training data. The aim of the experiments with these algorithms is as follows: to determine particle decompositions of all words into particles, and to build a particle N-gram model which, when it is interpolated with a word model, is able to generalise to unseen word sequences and smooth the word model probability estimates so as to reduce the *word level* perplexity.

It is conceivable that the optimal decompositions of words may be the undecomposed words themselves. Since it is necessary to restrict the search space of possible particles, none of the algorithms will find such a solution. Moreover, the stated aim of these experiments is to determine a particle model which best complements the characteristics of the word model. A particle model in which words are not decomposed will obviously not achieve this. In any case, for Russian it has been convincingly shown that the 430k vocabulary word N-gram model (N=3,4) is not very robust, due to the sparsity of the Russian training data, and combinations with other models will not be detrimental. Perplexity comparisons will be made against the 430k word trigram model for Russian and the 65k word trigram model for English. The perplexities of these models on the appropriate `training` and `eval-test` data sets are shown in Table 6.1.

	Word trigram perplexity		Model size
Language	`training`	`eval-test`	(parameters)
Russian (430k)	463.5	677.0	12,177,700
English (65k)	162.5	216.1	12,431,060

Table 6.1 *Perplexities of word trigram models on* `eval-test` *and* `training` *data partitions for the two corpora together with the number of parameters in each trigram model.*

All three algorithms presented below, simultaneously determine the decomposition function U of vocabulary words into particles and the set of particles Ψ. For language modelling purposes, a unique decomposition is desired for each vocabulary word. This requirement is acceptable from a linguistic point of view since most words are expected to have deterministic morphological decompositions which are not dependent on the surrounding word context; this contrasts with say, the part of speech of a word which is clearly dependent on the word's context. Moreover, if words are forced to have only one decomposition the resulting language model is made less complex than if each word were to have multiple probabilistic decompositions.

The first algorithm uses a list of linguistic affixes for each language which have been deemed to be potentially useful. A string matching operation then separates the affixes from each word in the vocabulary. The two data-driven algorithms use only the statistics of word and character occurrences in the training data and the location of word boundaries to determine the set of particles and word decompositions. An absolute minimum of prior linguistic knowledge about each language is therefore encoded in the two data-driven algorithms.

The first data-driven algorithm determines the optimal decomposition of each word in the vocabulary in turn by maximising the likelihood of the training data using a particle bigram model:

$$\hat{U}(w_i) = u_0^{w_i}, u_1^{w_i}, \dots, u_{L(w_i)-1}^{w_i} = \arg\max_{U(w_i)} \{LL_{bi}(U)\}. \tag{6.4}$$

The second data-driven algorithm iteratively selects the particle which, when it is inserted simultaneously into all possible words, maximises the training set likelihood using a particle bigram model:

$$\hat{u} = \arg\max_{u \in \Psi} \{LL_{bi}(U)\}. \tag{6.5}$$

For both algorithms the log-likelihood of the training data is computed using a particle bigram model for which the probability estimates are the maximum likelihood estimates,

$$LL_{bi}(U) = \sum_{i=1}^{N_P} \log P(u_i \mid u_{i-1}) = \sum_{i=1}^{N_P} \log \frac{N(u_{i-1}, u_i)}{N(u_{i-1})}. \tag{6.6}$$

This may be simplified further by grouping together the occurrences of pairs of particles in the training data,

$$LL_{bi}(U) = \sum_{\forall u_a, u_b : N(u_a, u_b) > 0} N(u_a, u_b) \cdot \log N(u_a, u_b) - \sum_{\forall u_a : N(u_a) > 0} N(u_a) \cdot \log N(u_a). \tag{6.7}$$

The normalization term has been omitted from the above calculations and is not evaluated by the algorithms, since evaluating $Z(h)$ for all possible h is computationally very demanding.

Experimental procedure

The perplexity of the particle bigram model on the training data is computed by predicting *all* tokens in the data using unpruned and unnormalized particle bigram models. The same calculation is performed in each of the data-driven algorithms. Predicting all tokens simplifies the implementation of the algorithms and does not affect the final decompositions. These perplexities are also given in the tables of results where appropriate and are used only to perform qualitative, not quantitative, comparisons between models.

Once U has been determined using one of the three algorithms, particle 6-gram models are built using Katz backoff and Good-Turing discounting. Initial experiments with particle 4-gram models showed the span of the particle 4-gram to be too restricted. This warranted the construction of particle N-grams with longer spans and 6-grams were instead used for all particle models. Since such models would be impractical if they were not pruned, the entropy-based pruning method described in Section 2.4.3 was employed to discard N-grams which were not useful. The threshold parameter for pruning all models was adjusted to ensure that all particle models contained around twelve million parameters, approximately the same number as in the word trigram models. Both the cutoffs and variable-length pruning techniques were also investigated for reducing the number of parameters in the particle models but both were shown to give a significantly lower performance than the entropy-based pruning method for the same number of model parameters.

For reasons already given, $Z(h) = 1$ is used in computing the stand-alone and interpolated perplexities of models in this chapter. As a consequence, the perplexity figures that are presented represent upper-bounds on the perplexity. The effect that correct normalization has on the perplexity is investigated in Section 6.3.4 for the best performing particle models obtained using each algorithm.

Analysis procedure

The same analysis methods that were used in the previous two chapters will also be used to examine the characteristics of each particle model. The distribution of log-probabilities apportioned both by the stand-alone particle model and the interpolated word trigram and particle model will be plotted and examined and the perplexities computed on events which are predicted by different back-off cases of the word model will also be presented.

In addition, since particle 6-grams are being used (for reasons explained above) it will be necessary to ensure that improvements in the performance of the interpolated model are not due to the possible inclusion of word 6-grams in the particle model. Therefore, the *maximum* number of whole (including partial) words in the particle contexts for the particle probabilities which make up each word will also be examined. Many particle contexts will still not have been observed and this is taken into consideration in the analysis. The maximum number of whole (including partial) words in each particle context for a word is found and averaged for all events in the `eval-test` data. It is not possible to consider only the context for the first particle in each word since this context may be shorter than that for particles in other positions of the word, hence the maximum of all these values is used. It should be noted that these values are also of relevance when considering how the particle model should be incorporated into the recognition process. Long word N-grams can significantly complicate time-synchronous search strategies, and would normally be avoided unless their usefulness dictated otherwise. Further analyses and comments of relevance to each algorithm will also be given where appropriate.

6.3.1 Linguistic affixes

The first set of particle experiments was performed using a simple, linguistics-based derivation of word decompositions for the 430k Russian vocabulary and the 65k English vocabulary. For the Russian vocabulary, twenty-eight prefixes and sixty suffixes[4] (given in Appendix A) were first chosen according to their usefulness and productivity (as perceived by the author) with the

[4]In this chapter, suffixes refers to both derivational and inflectional suffixes (inflections).

help of a textbook of Modern Russian Grammar (Offord, 1993). Similarly for the English vocabulary, fifty-two prefixes and sixty-two suffixes (also given in Appendix A) were obtained with the help of an encyclopedia of the English language (Crystal, 1995). The prefixes and suffixes were then systematically separated from the beginnings and ends, respectively, of all words in the vocabulary using a simple string-matching operation. All vocabulary words were eligible for decomposition and the affixes were separated wherever a match was made, irrespective of whether the match was linguistically correct or not. The <w> symbol was then appended to the terminal particle (the suffix or the end of a word) in the decomposition for each word.

6.3.1.1 Results

The numbers of distinct modelling units used in each of the affixes models are given in Table 6.2. In the same table, the average number of particles per word is also shown, where the number of particles in a word is scaled by the unigram count of the word in the training data and where it is not scaled.

	Number of	Average number of particles per word	
Language	modelling units	unweighted	frequency weighted
Russian	192,149	2.08	1.72
English	47,158	1.82	1.41

Table 6.2 *Number of modelling units and average numbers of particle units per word for both languages.*

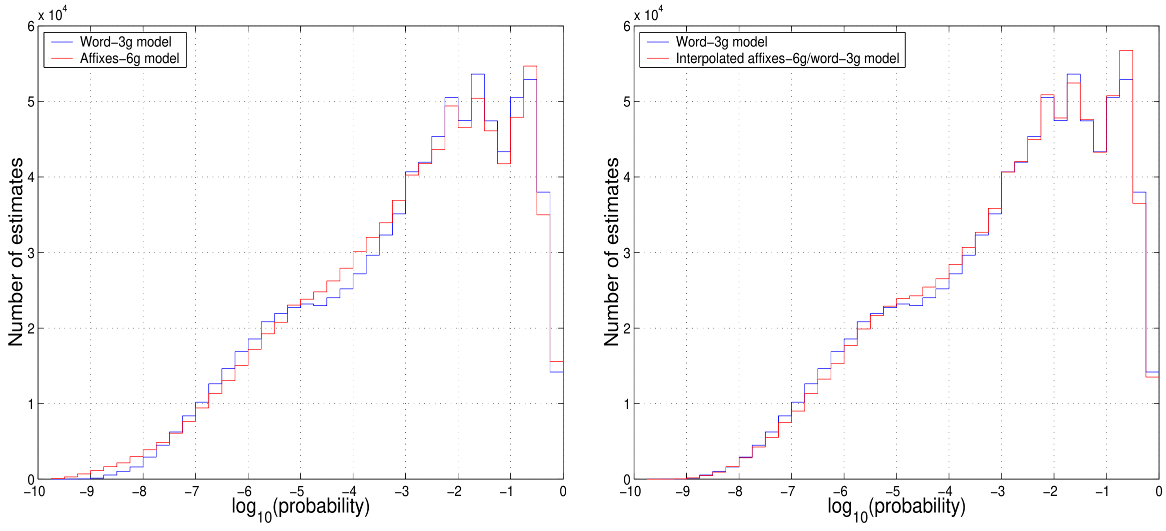
The results obtained using the word decompositions from this simplified linguistic method in particle 6-gram models are shown in Table 6.3. These results will be considered a suitable baseline from which to evaluate the other particle models described later, since the particles and word decompositions have been determined without regard to their statistical properties.

	Perplexity					
	`training`		`eval-test`		weights	% improvement
Language	PP^{2g}_{part}	PP^{6g}_{part}	PP^{6g}_{part}	PP^{6g}_{interp}	$\lambda_{word}, \lambda_{part}$	over word trigram
Russian	1019	510.0	747.5	632.3	0.67, 0.33	6.6
English	335.8	161.8	225.6	204.4	0.60, 0.40	5.4

Table 6.3 *Word level perplexities of stand-alone particle 6-gram models on* `eval-test` *and* `training`*, and interpolated particle and word trigram model on* `eval-test` *only.*

6.3.1.2 Model analysis

The above results show that even with a simple affix stripping scheme, quite substantial perplexity reductions can be obtained by interpolating the particle model with a word model. The distributions of log-probabilities apportioned by the particle model and the interpolated model are given in Figure 6.1.



(a) Affixes model.

(b) Interpolated affixes and word trigram model.

Figure 6.1 *Russian corpus (430k): distribution of log-probabilities for the word trigram, the affixes 6-gram and the interpolated models.*

The distributions differ significantly from those of the class models examined in the previous two chapters. The most notable difference is that the particle model assigns around twice as many very low probabilities ($< 10^{-8}$) as the word model does. On closer examination of the low probability events it was found that the majority of these were the result of predicting infrequent words containing at least one uncommon particle (usually the infix). Interestingly, such words must be reasonably well modelled by the word model since the interpolated model has characteristics of the word model in this region i.e. the very low probabilities assigned by the particle model are combined with higher probabilities from the word model. The distribution of probabilities greater than 10^{-8} is more favourable in the particle model, with a lower number of probabilities assigned between 10^{-8} and 10^{-5} and more between 10^{-5} and 10^{-3}. The distribution of probabilities above 10^{-3} is similar for both models. The peak around 0.1 in the distribution of the interpolated model suggests that some high probability events are boosted as a result of the combination with the particle model.

Table 6.4 shows the perplexities of the particle model computed for different word model back-off cases on the Russian `eval-test` data.

Back-off case	PP_{part}	PP_{word}
3	75.1	66.4
3-2	512	429
3-2-1	1.69×10^5	1.82×10^5

Table 6.4 *Russian corpus (430k): word-level perplexity of word model and affixes model, computed on sets of events that were predicted by different word model back-off cases.*

For the case where the word model backs off to its unigram distribution the particle model has a lower perplexity than the word model. This extends what was explained above, that the low probability estimates assigned by the word model are combined with higher estimates from the particle model, and also that low probability estimates from the particle model are combined with higher estimates from the word model. It is also worthwhile mentioning that the particle model predicted 29% of words that occurred less than five times in the training data with a higher probability than the word model.

Max. words in context	Percentage
1	25.5
2	59.3
3	14.8
4	0.4
5	0.0

Table 6.5 *The average maximum number of words used to predict words in Russian* `eval-test` *data using the affixes model.*

In Table 6.5 the percentages are given of different maximum numbers of whole (or partial) words used in the particle context for predicting words in the Russian `eval-test` data. Clearly, the affixes 6-gram model employs no word 6-grams, very few word 5-grams and relatively few word 4-grams in its predictions of events in the `eval-test` data. The majority of probability estimates use a context containing two words, which is equivalent to a word trigram. The improvement that is obtained with the interpolated model is not due to higher order word N-grams that have been built into the particle model and is due more to the smoothing of the word trigram probabilities. The smoothing has been achieved through the modelling of different dependencies in the words.

6.3.2 Greedy word decomposition algorithm

The first of the two data-driven algorithms for determining word decompositions automatically is now presented. Given an initial decomposition for each vocabulary word the algorithm seeks to optimise the decomposition of each word in turn. The optimal decomposition of a word is that which maximises the likelihood of the training data. The set of particles at any one time comprises those particles which are required to generate all the current word decompositions.

There are certain obvious similarities and differences between this algorithm and the E-M algorithm used in the multigram method described in Section 6.2.5. The important difference between the method proposed here and the multigram method is that this algorithm constrains the decomposition of each vocabulary word to be identical, wherever it appears in the training data. The multigram method determines that segmentation of the whole training data which has the highest probability. Hence, with the multigram method, identical words which appear in different locations in the training data may have different segmentations. Constraining words to have identical decompositions simplifies the final particle language model: each vocabulary word is forced to have a deterministic decomposition rather than multiple decompositions, each of which would otherwise require an associated probability. Another important difference with this algorithm is that word boundaries are known beforehand and therefore particles may never overlap word boundaries.

6.3.2.1 The algorithm

Internally, each word in the vocabulary is stored as a graph as shown in Figure 6.2 which represents compactly all the possible decompositions of a word into any-length particles. At any time, only one path in each graph is active; the sequence of particles on this active path represents the current decomposition of the word.

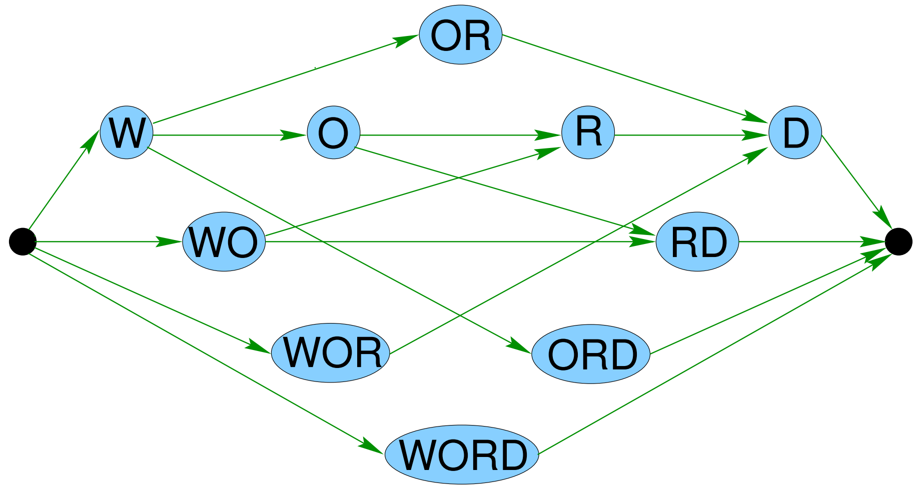


Figure 6.2 *Internal representation of all possible word decompositions of the word "WORD".*

Given an initial set of word decompositions, the unigram and bigram statistics are collected both for word-internal particle bigrams and those which span word boundaries. The initialisation procedure (described in detail in the next section) and the algorithm itself are described concisely by the following steps:

1. **Initialisation:**
   - `Limit maximum particle length to` l_{max} `characters`
   - `Collect word-internal particle statistics`
   - `Determine highest probability decomposition for each word`
2. **Iterate** `for all words` w`:`
   - **Iterate** `for all word decompositions` $U(w)$`:`
     - `compute likelihood of training data`
   - `Select best word decomposition`
   - `Update particle statistics`
3. `Repeat` **step 2** `for fixed number of iterations`

When a word's decomposition changes, the particle N-gram ($N = 1, 2$) counts must be modified to reflect the change. This can be achieved efficiently by traversing the old decomposition path of a word and subtracting the previous particle N-gram counts. The new decomposition path can then be traversed and the new particle N-gram counts included. Count updates of cross-word particle contexts are also performed so that the N-gram statistics remain consistent. When the decomposition of a word changes, it affects the contribution to the total likelihood of all word-internal particle N-gram counts and the counts of those particles which appear in the left and right hand contexts of each vocabulary word w. The update equations for updating particle N-gram counts when an old word decomposition is removed are presented below:

- For word internal particles, the unigram count of word w is subtracted:

$$i = 1 \dots L(w) - 1 : N(u^w_{i-1}, u^w_i) = N(u^w_{i-1}, u^w_i) - N(w). \tag{6.8}$$

- For particle bigram counts spanning word boundaries where the words are not identical (i.e. for particles in the word pairs (w, w') and (w', w) where $w \neq w'$) the update equations are:

$$N(u^w_{L(w)-1}, u^{w'}_0) = N(u^w_{L(w)-1}, u^{w'}_0) - N(w, w') \quad \forall w' : N(w, w') > 0 \tag{6.9}$$

$$N(u^{w'}_{L(w')-1}, u^w_0) = N(u^{w'}_{L(w')-1}, u^w_0) - N(w', w) \quad \forall w' : N(w', w) > 0 \tag{6.10}$$

- For the situation where the particle bigram spans a word boundary where the words are identical only the count update for words in the right-hand context are performed:

$$N(u^w_{L(w)-1}, u^w_0) = N(u^w_{L(w)-1}, u^w_0) - N(w, w) \quad N(w, w) > 0 \tag{6.11}$$

An analogous set of equations exists for obtaining the new counts when a new word decomposition is added.

Since at each step of the algorithm the word decomposition is selected which maximises the likelihood of the training data, the algorithm is guaranteed to converge. However, since the algorithm is greedy, the final set of word decompositions is only guaranteed to be locally optimal. Words are considered for decomposition in order of decreasing unigram frequency. This is motivated by the reasoning that words for which the statistics are more reliable should be considered before words for which the information about them is less reliable.

The value of l_{max}, which fixes the maximum size of particles, affects the total number of modelling units that will be used in the final particle language model. The larger l_{max} is, the larger the set of modelling units will be. Experiments were conducted for $l_{max} = 2, \dots, 6$ for Russian and $l_{max} = 3, \dots, 7$ for English and only one iteration through all vocabulary words was performed. One iteration was deemed sufficient since the perplexity on the Russian `eval-test` data of the stand-alone particle 6-gram language model with $l_{max} = 3$ was found to be only 0.5% less after two iterations than after one iteration.

6.3.2.2 Algorithm initialisation

Since the algorithm is greedy in nature and only capable of local convergence in general, the choice of initialisation method is particularly important. The initialisation method must generate a probability estimate for as many particles as possible, otherwise particles which are assigned a unigram count of zero initially will never be included in the decomposition of any word. Setting a floor unigram count on all particles does not solve the problem because the bigram statistics will then be inconsistent with the unigram statistics.

Ideally, the algorithm would be initialised with the particle statistics for all possible combinations of decompositions of all words. In practice this is only computationally possible if cross-word contexts are ignored. The method used here decomposes all words into all possible combinations of particles up to l_{max} characters in length and collects the word-internal particle statistics. When these statistics have been collected, the highest probability path through each word is found while still ignoring cross-word contexts for the time being. The decompositions which are obtained act as the initialisation for the re-estimation algorithm (step 2) in which cross-word contexts are then considered. In addition, the word decompositions obtained with the *affixes* method in the previous section were also used as an initialisation to assess the algorithm's ability to improve existing word decompositions.

6.3.2.3 Results

Preliminary results using an algorithm similar to the one described here were published in (Whittaker and Woodland, 1998) where the algorithm used the word decompositions from the *affixes* method as the initialisation. Results are presented for the decompositions obtained from the initialisation (ini) in step 1 of the algorithm and the optimised decomposition (opt) after one iteration of step 2 of the algorithm. For clarity, results have only been tabulated for models with l_{max} around that value which produced the best performing interpolated model. The model number refers to the value of l_{max} that was used.

	Number of	Average number of particles per word	
Model	modelling units	unweighted	frequency weighted
2 (ini)	2,591	5.65	3.00
2 (opt)	2,580	5.37	2.85
3 (ini)	20,970	4.07	2.26
3 (opt)	20,563	3.54	2.03
4 (ini)	77,301	3.20	1.94
4 (opt)	73,615	2.77	1.68
affixes (ini)	192,149	2.08	1.72
affixes (opt)	189,805	2.09	1.73

Table 6.6 *Number of modelling units and average numbers of particle units per word for Russian.*

The number of particle modelling units in the final language model is given in Table 6.6 for

Model	Number of modelling units	Average number of particles per word	
		unweighted	frequency weighted
4 (ini)	28,850	2.77	1.68
4 (opt)	27,385	2.43	1.52
5 (ini)	38,980	2.47	1.57
5 (opt)	36,619	2.14	1.39
6 (ini)	42,739	2.37	1.53
6 (opt)	40,019	2.03	1.34
affixes (ini)	47,158	1.82	1.43
affixes (opt)	46,804	1.82	1.44

Table 6.7 *Number of modelling units and average numbers of particle units per word for English.*

Russian and Table 6.7 for English. Both tables also show the average number of particles per word, weighted both by a word's frequency in the training data and unweighted. The figures are given for the word decompositions used as the initialisation for the algorithm and after running the algorithm for one iteration.

The perplexities of the particle model are shown in Table 6.8 for Russian and Table 6.9 for English. The algorithm is guaranteed to choose decompositions which increase the training data likelihood using a bigram particle model and this is shown in the second column of each table. To compare how the bigram criterion extends to the 6-gram models the perplexity on the `training` data of the pruned 6-gram particle model is given in the third column. Perplexity figures are also given for the stand-alone particle model and the interpolated model on `eval-test` data, along with the relative improvement of the interpolated model over the corresponding word trigram model.

	Perplexity					
	`training`		`eval-test`		weights	% improvement
Model	PP^{2g}_{part}	PP^{6g}_{part}	PP^{6g}_{part}	PP^{6g}_{interp}	$\lambda_{word}, \lambda_{part}$	over word trigram
2 (ini)	21110	706.9	905.8	635.1	0.75, 0.25	6.2
2 (opt)	13160	678.7	876.0	630.3	0.73, 0.27	6.9
3 (ini)	2941	611.4	821.4	635.8	0.72, 0.28	6.1
3 (opt)	1573	572.6	779.4	628.2	0.68, 0.32	7.2
4 (ini)	1423	589.9	813.6	641.1	0.74, 0.26	5.3
4 (opt)	729.2	532.1	769.6	635.6	0.71, 0.29	6.1
affixes (ini)	1019	510.0	747.5	632.3	0.67, 0.33	6.6
affixes (opt)	905.0	518.8	751.0	633.3	0.68, 0.32	6.5

Table 6.8 *Russian corpus (430k): word level perplexities of stand-alone particle 6-gram models on `eval-test` and `training`, and interpolated particle and word trigram model on `eval-test` only.*

	Perplexity					
	`training`		`eval-test`		weights	% improvement
Model	PP^{2g}_{part}	PP^{6g}_{part}	PP^{6g}_{part}	PP^{6g}_{interp}	λ_{word}, λ_{part}	over word trigram
4 (ini)	458.2	185.0	245.4	206.7	0.71, 0.29	4.3
4 (opt)	293.2	174.4	237.2	205.2	0.67, 0.33	5.0
5 (ini)	398.5	177.4	240.4	206.5	0.69, 0.31	4.4
5 (opt)	252.8	166.1	230.8	204.9	0.65, 0.35	5.2
6 (ini)	378.9	178.2	242.9	207.0	0.71, 0.29	4.2
6 (opt)	243.1	166.6	232.8	205.4	0.67, 0.33	5.0
affixes (ini)	335.8	161.8	225.6	204.4	0.60, 0.40	5.4
affixes (opt)	310.3	167.2	227.2	205.3	0.62, 0.38	5.0

Table 6.9 *English corpus (65k): word level perplexities of stand-alone particle 6-gram models on* `eval-test` *and* `training`*, and interpolated particle and word trigram model on* `eval-test` *only.*

6.3.2.4 Model analysis

The value of l_{max} that is used in a model has a marked effect on the number of modelling units. Obviously this is because the initialisation method uses all occurring character combinations up to length l_{max} and the number of such combinations increases as l_{max} is increased. It is interesting that many different particles are retained when the highest probability path is found through each word. This may indicate that the initialisation method fails to exploit fully the commonality between particle sequences in different words.

An interesting aspect of Stage 2 of the algorithm is that some modelling units which were present in the initialisation do not appear in the optimised decompositions. The reduction in the number of modelling units varies between 0.42% and 6.3%. For all models, the training set perplexity of the bigram model decreases (the algorithm guarantees that the likelihood will improve or stay the same) and this is effected to some extent by adding extra bigram parameters to the model so as to better fit the training data. Extra parameters are added by changing words' decompositions and in the process, superfluous modelling units are discarded. Since, cross word contexts are used in step 2 of the algorithm, there is significant additional information compared to that used in the initialisation with which to obtain new word decompositions.

Even a cursory glance over the changes that take place between the initial and optimised decompositions shows that the algorithm produces subjectively more sensible linguistic decompositions for words. Many single-character combinations that were present in the initial decompositions are grouped together into recognisable morphological units after optimisation. This is most apparent among suffixes where it may be postulated that the presence of the <w> marker helps to isolate such particles. The agglomeration of smaller particles also partly explains the smaller average number of particles per word in the optimised decompositions, although this is not true for the affixes initialisation and this is examined further below.

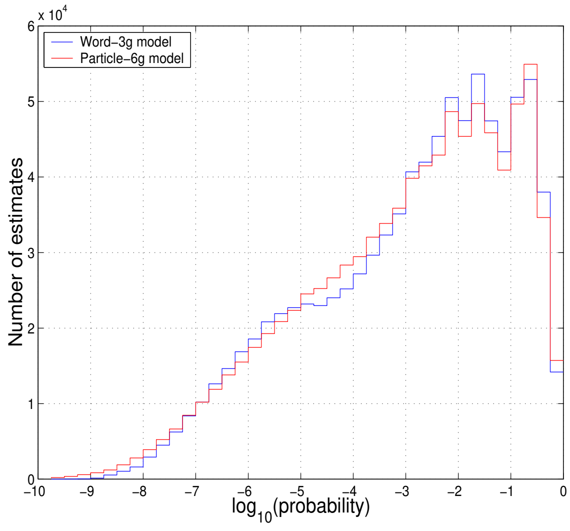
As l_{max} is increased the perplexity of the bigram model on the training data decreases, however for the English corpus this trend does not extend to the perplexity of the pruned 6-gram model on the training data. This was not attributed so much to the bigram criterion being a poor approximation for the 6-gram model but rather to the restricted number of parameters in the 6-gram model. The bigram model can add as many parameters as it requires to model the training data but the 6-gram model is pruned and so begins to model the training data less well. This trend is also observed on the `eval-test` data. A model which cannot make full use of its more refined particle dependencies (because of parameter restrictions) also does not generalise well. If the 6-gram model were pruned using some form of cross-validation the performance might well be improved.

For the affixes initialisation, step 2 of the algorithm has produced worse decompositions for the 6-gram models. Although the perplexity of the bigram model on the training data has decreased by around 11%, for the same reasons given above, the 6-gram model has failed to exploit the optimised particle dependencies. From Tables 6.6 and 6.7 it is clear that the optimised affixes decompositions have more particles per word on average than the initial affixes decompositions. Closer examination of the optimised decompositions revealed that many affixes had been further decomposed into smaller particles. This explains the reduction in perplexity of the unpruned bigram model and also why the performance of the 6-gram model worsened. The effect was offset to a small extent by the recombination of prefixes with infixes in some decompositions to form particles which were themselves infixes of other words in the vocabulary.

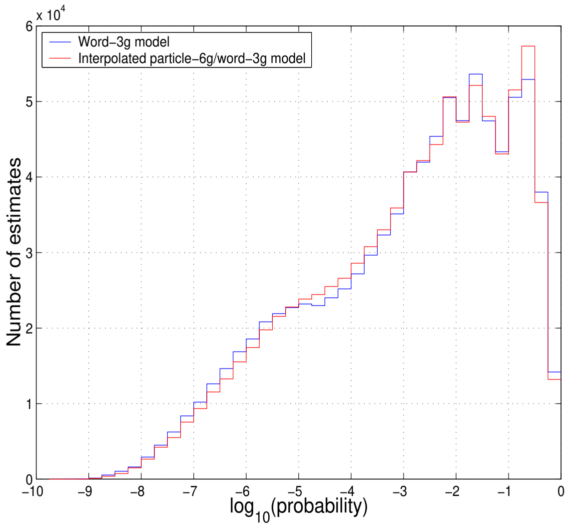
Interpolating the particle model with the word model results in a minimum for Russian at $l_{max} = 3$ and for English at $l_{max} = 4$, however the differences between adjacent models are small (up to 1.2%) so it is difficult to conclude much from these values. Nonetheless the improvements over each word trigram model are appreciable. Compared with the affixes model the improvement over the word model for Russian is 9% better, whereas for English the improvement is around 4% less than the affixes model.

Figure 6.3 shows the distribution of log-probabilities for the stand-alone and interpolated Russian word decomposition model with $l_{max} = 3$. Comparison with Figure 6.1 reveals that the distribution of log-probabilities is similar to that obtained with the affixes model. The number of very low probabilities ($< 10^{-8}$) assigned by the particle model is reduced in the interpolated model where there are fewer even than in the word model (around 20% fewer). Words assigned very low probabilities were found to have 3.4 particles per word on average. This is significant in that the particle model involves taking the product of the component particle probabilities. The smoothing action of the particle model is suggested by there being fewer low probabilities in the range 10^{-8} to 10^{-5} and more between 10^{-5} and 10^{-3}. The peak around 0.1 indicates that the probabilities of events that are well estimated by the word model are boosted to some extent by the particle model. Words assigned these high probabilities were found to have 1.27 particles per word on average.

Table 6.10 gives the perplexities computed on events in the Russian `eval-test` data that were predicted by different back-off cases of the word model. The values show that the particle model does not perform as well as the word model when the word model uses a trigram or bigram estimate but does do better when the word model backs off to the unigram distribution.



(a) Word decomposition model.



(b) Interpolated word decomposition and word trigram model.

Figure 6.3 *Russian corpus (430k): distribution of log-probabilities for the word trigram, the particle 6-gram obtained using the word decomposition algorithm with* $l_{max} = 3$*, and the interpolated model.*

The differences between the two models are not very large but the smoothing effect is sufficient in the interpolated model to produce a 7.2% perplexity reduction over the stand-alone word model. The word decomposition model predicted 36% of words that occurred less than five times in the training data with a higher probability than the word model. This is more than the affixes model predicted and helps to explain its better smoothing effect.

Back-off case	PP_{part}	PP_{word}
3	77.1	66.4
3-2	559	429
3-2-1	1.69×10^5	1.82×10^5

Table 6.10 *Russian corpus (430k): word-level perplexity of word model and stand-alone word decomposition model with* $l_{max} = 3$*, computed on sets of events that were predicted by different word model back-off cases.*

In Table 6.11 the average maximum number of words used for predicting the particles in words of the `eval-test` data by the particle model is given. Very few word 5-grams and no word 6-grams are used and only 13% of predictions use word 4-grams. Nonetheless, to test whether the number of 4-grams was having a disproportionate effect on the perplexity, a particle 4-gram model was interpolated with the word 4-gram model described in Section 3.3.3, and all singleton events were discarded from both models. Consequently, any word 4-grams present in the particle model were also present in the word model and the effect of such events was therefore minimised. A 6.6% perplexity reduction was obtained over the stand-alone word 4-gram model

Max. words in context	Percentage
1	31.4
2	54.8
3	13.0
4	0.8
5	0.0

Table 6.11 *The average maximum number of words used to predict words in Russian* `eval-test` *data using the word decomposition model with* $l_{max} = 3$.

($PP^{4g}_{word} = 659.6$) which shows that the particle model does indeed provide a smoothing effect to the word N-gram probabilities by capturing different dependencies within words and that the improvement is not due to the inclusion of word 4-grams. As further confirmation, the particle 6-gram model was interpolated with a word 6-gram model ($PP^{6g}_{word} = 660.6$) that had been pruned to contain 12 million parameters. The interpolated model produced a 6.8% reduction in perplexity over the stand-alone word model.

The word decomposition algorithm is very sensitive to the initialisation that is used. In particular, only those particles that are present in the initial decomposition can ever occur in the optimised decompositions. Ideally, initialisations that consider all sequences of all possible sequence lengths would be used and would also include cross-word contexts as well. It has already been explained however that this was computationally infeasible.

6.3.3 Greedy particle selection algorithm

The second of the data-driven algorithms presented here adopts a greedy approach to the selection of individual particles by choosing, at each iteration, the best particle from a restricted set under consideration. In this algorithm, the emphasis is on the order in which particles are selected rather than whether particles are selected at all. The order in which particles are chosen affects the decompositions of words since particles are only allowed to replace existing particles in a limited number of ways. As before, l_{max}, the maximum length of a particle unit, will also be seen to be an important factor.

6.3.3.1 The algorithm

The algorithm only requires word unigram and bigram statistics from the training data and a list of all possible candidate particles of different lengths. This list only contains those particles which actually occur *within* words of the vocabulary and is not simply a list of all possible combinations of the single characters that may occur. Moreover, it does not include particles which cross word boundaries. Initialising the algorithm involves decomposing all words into their constituent single characters. The contents of the set of particles Ψ at initialisation therefore comprises all single characters which occur in words of the vocabulary. Single characters must always appear in the final set since they may be necessary as *filler* particles to complete a decomposition which does not divide exactly into larger particles. The algorithm may be described concisely by the following steps:

1. **Initialisation**:
   - $l = 1$
   - decompose words into l-character particles
   - compute likelihood of training data
2. $l = l + 1$
3. **Iterate** for all l-character candidate particles u^{can}:
   - 'insert' particle u^{can} in all words w
   - compute change in likelihood
   - 'remove' particle u^{can} from all words w
4. Insert best l-character particle into Ψ and permanently in all words
5. If desired number of particles obtained then **terminate**
6. If no particles remaining then **terminate**
7. If improvement goto **step 3**, else goto **step 2**

After each iteration, the particle which results in the greatest reduction in perplexity is permanently added to the final set of particles. Naturally, the order in which particles are chosen affects the selection of all subsequent particles. Since the algorithm only accepts configurations which result in an increase in the optimisation function, the algorithm is guaranteed to converge, however due to its greedy nature it is only likely to find a locally optimal solution. In these experiments the algorithm is only used to determine a set of particles up to some maximum size l_{max} and does not run to completion.

The choice of optimisation criterion was found to have a significant effect on the outcome of the algorithm. The performance of the final language model was shown to be poor when particles were selected using the algorithm with a particle *unigram* criterion. Consequently, only results for particles selected with a particle *bigram* criterion are reported here. Other N-gram ($N > 3$) optimisation criteria were not considered because of the increase in algorithm complexity.

6.3.3.2 Update procedure

The optimisation criterion for the algorithm is given by Equation (6.7). There is no simple method for updating the likelihood when a particle is tentatively inserted into a word, since particle bigrams may span word boundaries i.e. the context may be in the preceding word. One solution would be to flag all particles which are affected by the insertion of a particle but this is unappealing since it results in excessive memory use. Instead, the approach that was adopted maintained a separate table of bigram counts for storing all bigram count changes that occur when a particle is inserted into a word. When all words have been examined to see whether the tentative particle can be inserted, the counts table is traversed and the likelihood contributions of each affected particle bigram updated accordingly. Before the tentative insertion of the next particle, the entries in the count table are first removed. This method was shown to be efficient both in terms of memory use and speed.

Since cumulative count changes are stored in the counts table, care must be taken to ensure that the correct identities of particles are used in updating counts since a change in the decomposition of one word will inevitably affect subsequent words. Shown below are the count update equations for replacing a sequence of $L(u^{rep})$ particles $u_b^{rep}, \ldots, u_i^{rep}, \ldots, u_e^{rep}$ in word w with the candidate particle u^{can} that is being tentatively inserted. Naturally, concatenating the sequence $u_b^{rep}, \ldots, u_i^{rep}, \ldots, u_e^{rep}$ produces u^{can} and $l(u_b^{rep}) + \ldots + l(u_i^{rep}) + \ldots + l(u_e^{rep}) = l(u^{can})$ where $l(u)$ is the number of characters in particle u. ($l(w)$ is similarly defined for the number of characters in word w.) Particle replacements may occur at the beginning ($u_b^{rep} = u_0^w$) or the end ($u_e^{rep} = u_{L(w)-1}^w$) of word w or word internally ($(u_b^{rep}, \ldots, u_e^{rep}) = (u_j^w, \ldots, u_{j+L(u^{rep})-1}^w)$):

- Beginning of word w (if $l(u^{can}) \neq l(w)$):
  - left context:

$$\text{if } u_{L(w')-1}^{w'} = u^{can} \begin{cases} N(u^{can}, u_b^{rep}) = N(u^{can}, u_b^{rep}) - N(w', w) & \forall w' : N(w', w) > 0 \\ N(u^{can}, u^{can}) = N(u^{can}, u^{can}) + N(w', w) & \forall w' : N(w', w) > 0 \end{cases} \tag{6.12}$$

$$\text{else} \begin{cases} N(u_{L(w')-1}, u_b^{rep}) = N(u_{L(w')-1}, u_b^{rep}) - N(w', w) & \forall w' : N(w', w) > 0 \\ N(u_{L(w')-1}, u^{can}) = N(u_{L(w')-1}, u^{can}) + N(w', w) & \forall w' : N(w', w) > 0 \end{cases} \tag{6.13}$$

  - right context:

$$N(u_e^{rep}, u_{L(u^{rep})}^w) = N(u_e^{rep}, u_{L(u^{rep})}^w) - N(w) \tag{6.14}$$

$$N(u^{can}, u_{L(u^{rep})}^w) = N(u^{can}, u_{L(u^{rep})}^w) + N(w) \tag{6.15}$$

- End of word w:
  - left context:

$$N(u_{L(w)-L(u^{rep})-1}^w, u_b^{rep}) = N(u_{L(w)-L(u^{rep})-1}^w, u_b^{rep}) - N(w) \tag{6.16}$$

$$N(u_{L(w)-L(u^{rep})-1}^w, u^{can}) = N(u_{L(w)-L(u^{rep})-1}^w, u^{can}) + N(w) \tag{6.17}$$

  - right context:

$$\text{if } u_0^{w'} = u^{can} \begin{cases} N(u_e^{rep}, u^{can}) = N(u_e^{rep}, u^{can}) - N(w, w') & \forall w' : N(w, w') > 0 \\ N(u^{can}, u^{can}) = N(u^{can}, u^{can}) + N(w, w') & \forall w' : N(w, w') > 0 \end{cases} \tag{6.18}$$

$$\text{else} \begin{cases} N(u_e^{rep}, u_0^{w'}) = N(u_e^{rep}, u_0^{w'}) - N(w, w') & \forall w' : N(w, w') > 0 \\ N(u^{can}, u_0^{w'}) = N(u^{can}, u_0^{w'}) + N(w, w') & \forall w' : N(w, w') > 0 \end{cases} \tag{6.19}$$

- Internally within word w at position u_j^w:
  - left context:

$$\text{if } u_{j-1} = u^{can} \begin{cases} N(u^{can}, u_j^w) = N(u^{can}, u_j^w) - N(w) \\ N(u^{can}, u^{can}) = N(u^{can}, u^{can}) + N(w) \end{cases} \tag{6.20}$$

$$\text{else} \begin{cases} N(u_{j-1}^w, u_j^w) = N(u_{j-1}^w, u_j^w) - N(w) \\ N(u_{j-1}^w, u^{can}) = N(u_{j-1}^w, u^{can}) + N(w) \end{cases} \tag{6.21}$$

  - right context:

$$N(u_e^{rep}, u_{j+L(u^{rep})}) = N(u_e^{rep}, u_{j+L(u^{rep})}) - N(w) \tag{6.22}$$

$$N(u^{can}, u_{j+L(u^{rep})}) = N(u^{can}, u_{j+L(u^{rep})}) + N(w) \tag{6.23}$$

- Always update for $i = 0, \dots, L(u^{rep}) - 2$:

$$N(u_i^{rep}, u_{i+1}^{rep}) = N(u_i^{rep}, u_{i+1}^{rep}) - N(w). \tag{6.24}$$

The update equations for the unigram counts are straightforward since only those of the particles being replaced, u_i^{rep} $i = 0, \dots, L(u^{rep}) - 1$ and u^{can}, are affected.

6.3.3.3 Heuristic algorithm speedups

For the experiments conducted with this algorithm, the set of possible candidate particles was chosen to be the most frequently occurring ten thousand particles of each size in the training data. For some sizes of particles, there were in fact far fewer than ten thousand particles, but generally there were far more. Since the algorithm performs a search over all particles of a particular size, finds the best particle and repeats the search over all the remaining particles, the search space of possible particles is crucial to the speed of the algorithm. It was noted that the ranking of particles according to their improvement of the likelihood did not change significantly from one iteration to the next. The conclusion that a high number of useless search operations was being performed led to the following heuristic improvement:

1. Whenever a new size of particle is considered, a full search is performed over all particles and the particles are then ranked according to the contribution their inclusion made to the log-likelihood.

2. For subsequent iterations over particles of the same length, the search is restricted to the top K candidate particles. As each best particle is found, it is removed from the top K candidate particles and the next best particle from the initial ranking is added to the candidate list to form K candidate particles once more.

The ranking of particles should ideally be recomputed after each iteration since the inclusion of the best particle will affect the contribution of all other particles, however, this would have the effect of slowing down the algorithm even more. It is also possible that the would-be best particle does not appear in the top K candidate list, however, the trade-off in accuracy is easily justified by the increase in algorithm speed. The value of $K = 100$ was used for the experiments on English and $K = 10$ for those on Russian since these values allowed the algorithm to produce decompositions in an acceptable amount of time (several days). In fact, it transpired that the performance of the $l_{max} = 3$ model was found to be more or less insensitive to the value of K that was used: the perplexity of the stand-alone particle model on the Russian `eval-test` data was only 0.25% higher with decompositions obtained using $K = 10$, than those obtained using $K = 100$.

6.3.3.4 Permitted particle replacements

There remains the issue of how particles may replace existing combinations of particles during the course of the algorithm. A decision had to be made as to which replacements were possible and these are outlined briefly in Table 6.12 for the case of 4 characters, A, B, C and D, where the brackets denote the particle composed of the characters within the brackets.

<table>
<tr><th>Original</th><th>Replacement</th></tr>
<tr><td rowspan="6">(A)(B)(C)(D)</td><td>(AB)(C)(D)</td></tr>
<tr><td>(A)(BC)(D)</td></tr>
<tr><td>(A)(B)(CD)</td></tr>
<tr><td>(AB)(CD)</td></tr>
<tr><td>(ABC)(D)</td></tr>
<tr><td>(A)(BCD)</td></tr>
<tr><td>(AB)(C)(D)</td><td rowspan="6">(ABCD)</td></tr>
<tr><td>(A)(BC)(D)</td></tr>
<tr><td>(A)(B)(CD)</td></tr>
<tr><td>(AB)(CD)</td></tr>
<tr><td>(ABC)(D)</td></tr>
<tr><td>(A)(BCD)</td></tr>
<tr><td>(AB)(C)(D)</td><td rowspan="2">(ABC)(D)</td></tr>
<tr><td>(A)(BC)(D)</td></tr>
<tr><td>(A)(BC)(D)</td><td rowspan="2">(A)(BCD)</td></tr>
<tr><td>(A)(B)(CD)</td></tr>
</table>

Table 6.12 *Permitted particle replacements for the case of four characters.*

Particles which are already used in a decomposition may only be combined with adjacent particles to form larger units and cannot themselves be split up to be combined with other particles. Due to the way the replacement operation works, the order in which the algorithm selects particles is implicitly encoded in the decompositions. If (AB) is selected before (BC) then (A)(B)(C) must be decomposed into (AB)(C), and it is impossible to obtain (A)(BC). Later on, if (ABC) is selected as a particle then it will replace (AB)(C) as the decomposition, assuming that neither (AB) nor (C) has already been incorporated into some other particle.

6.3.3.5 Results

Particle 6-gram models were built and pruned using decompositions from the algorithm after all particles of a fixed length ($l_{max} = 1, \ldots, 5$ for Russian and $l_{max} = 1, \ldots, 9$ for English) had been considered for inclusion into Ψ. This approach was adopted since the characteristics of a model only changed appreciably when the size of particle under consideration was increased (step 2 in the algorithm). In other words, the relationship between the number of particles selected N_Ψ and the perplexity of a model was similar to a decreasing step function with discontinuities at the points where l_{max} was increased. The number of particle units used in each of the final particle models is shown in Table 6.13 for Russian and Table 6.14 for English, together with the average number of particles per word both weighted by a word's unigram frequency and unweighted.

	Number of	Average number of particles per word	
Model (l_{max})	modelling units	unweighted	frequency weighted
1	128	8.98	4.82
2	1660	5.27	2.93
3	9512	3.85	2.15
4	18675	3.15	1.79
5	26617	2.86	1.58

Table 6.13 *Number of modelling units and average numbers of particle units per word for Russian.*

	Number of	Average number of particles per word	
Model (l_{max})	modelling units	unweighted	frequency weighted
1	64	7.59	4.40
2	985	4.56	2.69
3	5379	3.23	1.91
4	11416	2.69	1.56
5	16717	2.40	1.36
6	21012	2.25	1.25
7	24648	2.15	1.17
8	28004	2.07	1.11
9	31258	1.98	1.08

Table 6.14 *Number of modelling units and average numbers of particle units per word for English.*

The perplexity results for the different particle models are shown for Russian in Table 6.15 and for English in Table 6.16.

	Perplexity					
	`training`		`eval-test`		weights	% improvement
Model (l_{max})	PP^{2g}_{part}	PP^{6g}_{part}	PP^{6g}_{part}	PP^{6g}_{interp}	λ_{word}, λ_{part}	over word trigram
1	347300	1529	1784	662.9	0.90, 0.10	2.1
2	25910	695.2	897.8	630.6	0.74, 0.26	6.9
3	4171	591.8	800.2	626.2	0.69, 0.31	7.5
4	1575	542.6	766.0	627.1	0.68, 0.32	7.4
5	979.2	515.4	750.0	630.1	0.68, 0.32	6.9

Table 6.15 *Russian corpus (430k): word level perplexities of stand-alone particle 6-gram models on* `eval-test` *and* `training`*, and interpolated particle and word trigram model on* `eval-test` *only ($K = 10$).*

	Perplexity					
	`training`		`eval-test`		weights	% improvement
Model (l_{max})	PP^{2g}_{part}	PP^{6g}_{part}	PP^{6g}_{part}	PP^{6g}_{interp}	λ_{word}, λ_{part}	over word trigram
1	93900	429.4	472.3	214.4	0.93, 0.07	1.7
2	6127	217.1	272.4	209.0	0.77, 0.23	3.3
3	1040	191.8	250.3	206.2	0.71, 0.29	4.6
4	462.2	175.2	238.4	204.5	0.66, 0.34	5.4
5	315.3	166.7	231.7	203.9	0.64, 0.36	5.6
6	261.6	160.9	226.7	203.4	0.62, 0.38	5.9
7	233.3	155.4	222.6	202.8	0.59, 0.41	6.2
8	216.3	151.5	219.6	202.3	0.56, 0.44	6.4
9	206.0	148.2	217.5	201.9	0.54, 0.46	6.6

Table 6.16 *English corpus (65k): word level perplexities of stand-alone particle 6-gram models on* `eval-test` *and* `training`*, and interpolated particle and word trigram model on* `eval-test` *only ($K = 100$).*

In Table 6.17 the first ten particles for English selected by the algorithm for $l_{max} = 2, 3, 4, 5$ are shown in the order in which they were selected.

Table 6.18 shows ten randomly chosen words from the English decompositions for $l_{max} = 5$ and gives an idea of the way in which words are decomposed by the algorithm and how the particle replacement operation works.

6.3.3.6 Model analysis

The perplexity on the training data of both the bigram model and the pruned 6-gram models decreases as more particles are selected by the algorithm. In a manner similar to the word decomposition method, the particle selection algorithm adds unigram and bigram parameters

order	$l_{max} = 2$	$l_{max} = 3$	$l_{max} = 4$	$l_{max} = 5$
1	TH	THE<w>	THER<w>	THOUS
2	AN	AND<w>	THAT<w>	ETEEN<w>
3	IO	ING<w>	BEEN<w>	THERE<w>
4	OF<w>	ERE<w>	DRED<w>	THINK<w>
5	IN	NIN	ENTY<w>	EIGHT<w>
6	RE	WIT	HAVE<w>	THREE<w>
7	RO	FRO	THIS<w>	OTHER<w>
8	TO<w>	DRE	THAN<w>	SEVEN<w>
9	GH	HUN	WITH	GOVER
10	EN	EEN<w>	WERE<w>	INTER

Table 6.17 *The first ten particles of each size chosen by the algorithm for English.*

Word	Decompositions
PRESUMPTUOUS	PRE—SUMP—TUOUS<w>
BRISTLY	BRIST—LY<w>
TRAVERSAL	TRAV—ERSAL<w>
WARDSHIP	WAR—DSHIP<w>
SOMETIMES	SOM—ETIM—ES<w>
DISPATCHED	DIS—PAT—CHED<w>
HEATHEN	HEA—THEN<w>
GRANDMOTHERS	GRAND—MOTH—ERS<w>
NEWSCASTER	NEW—SC—ASTER<w>
OBLIGATION	OBLIG—ATION<w>

Table 6.18 *Ten randomly chosen English words and their decompositions with* $l_{max} = 5$.

to the model as necessary so as to fit the training data better. However, similar reductions in perplexity are also observed on the held-out `eval-test` data which suggests that none of the models that were built were over-trained.

The best stand-alone and interpolated models for both languages are better than for either the affixes or word decomposition methods. For the interpolated word and particle model an optimal value of l_{max} was not found for English however for Russian $l_{max} = 3$ was found to be marginally better than $l_{max} = 4$. For Russian, $l_{max} = 3$ was also found to be the optimal value with the word decomposition algorithm. This shows fairly conclusively that dependencies between particles limited in length to three characters best complement the dependencies in the Russian word model. For the English models, part of the performance improvement was undoubtedly due to the selective inclusion of word N-grams ($N > 3$) in the particle model. Such events are perhaps more robust in English than in Russian and so retaining them improves the performance on unseen data. In Table 6.19 the average maximum number of words used in particle contexts for predicting words is given for English with $l_{max} = 4$ and $l_{max} = 9$. Increasing l_{max} certainly increases the maximum number of words used in particle contexts, although even for $l_{max} = 9$ only 24% of vocabulary words are also their own decompositions.

	$l_{max} = 4$	$l_{max} = 9$
Max. words in context	Percentage	Percentage
1	18.6	21.7
2	52.3	46.1
3	25.6	25.6
4	3.3	5.7
5	0.2	0.9

Table 6.19 *The average maximum number of words used to predict words in English* `eval-test` *data using the particle selection model with $l_{max} = 4$ and $l_{max} = 9$.*

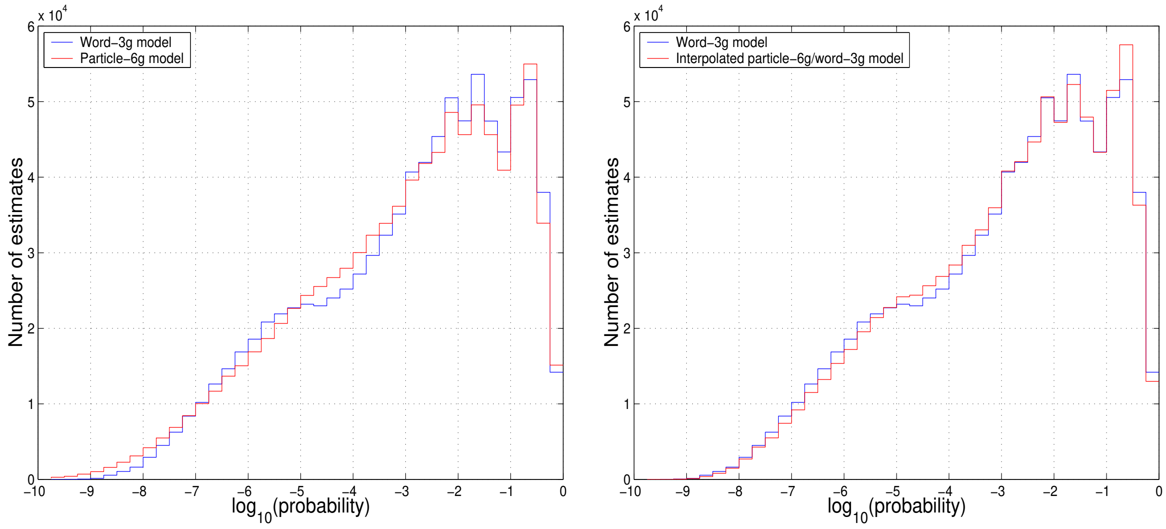
When 4-gram models with all singleton events discarded were built for both the $l_{max} = 9$ particle model and the word model (see Section 3.3.4) the perplexity reduction of the combined model was only 0.4%. The interpolated particle 6-gram and word 6-gram also only gave a 1% reduction in perplexity over the stand-alone word 6-gram. This clearly indicates that both particle models contained a large proportion of word N-grams. It is interesting to note that the particle 6-gram model built using decompositions from the $l_{max} = 10$ word decomposition algorithm did not have characteristics similar to the $l_{max} = 9$ particle selection model i.e. word N-grams ($N > 3$) were not present to the same extent and for those decompositions only 12.5% of words were also their own decompositions.

The first ten particles that were selected by the algorithm for different maximum particle lengths was shown in Table 6.17. For $l_{max} = 2, 3$ the particles appear to be frequent and potentially useful modelling units. (Approximately half of the particles shown for $l_{max} = 2$ are in the ten most frequently occurring 2-character particles in the training corpus.) For $l_{max} = 4, 5$ the tendency is to choose whole words (or ends of words) instead of word-internal particles. This was not the case for Russian and reflects the assertion that whole words rather than sub-word units tend to be more useful for modelling English.

The algorithm is restricted by the limited number of ways in which particles may replace existing particles. This is shown clearly in Table 6.18 where the selection of smaller particles earlier on has formed groups which prevent the subsequent selection (at least in the decompositions shown) of intuitively more useful particles. This is particularly apparent for the words "SOMETIMES" and "NEWSCASTER".

In Figure 6.4 the distribution is plotted of log-probabilities apportioned by the $l_{max} = 3$ particle selection and interpolated models to events in the `eval-test` data. The distribution is very similar to the word decomposition algorithm and this reflects the similar performance of the two models. The salient points are that the interpolated model has fewer very low probabilities

($< 10^{-8}$) than in the word model (18% fewer) and a more favourable distribution of probabilities in the middle range. The peak observed for probabilities around 0.1 again suggests that well estimated words have their probabilities boosted by the particle model. It should be noted, however, that this is still a smoothing effect and is not necessarily undesirable.



(a) Particle selection model.

(b) Interpolated particle selection and word trigram model.

Figure 6.4 *Russian corpus (430k): distribution of log-probabilities for the word trigram, the particle 6-gram model obtained using the particle selection algorithm with* $l_{max} = 3$*, and the interpolated model.*

The perplexities of the particle and word model computed on events in the Russian `eval-test` data for different back-off cases of the word model are given in Table 6.20. Although the particle selection model, when it is combined with the word model, gives the best performance of the three different particle models, the perplexities of the stand-alone model for the different back-off cases are all slightly higher than exhibited by the other two particle models. However, this is simply because the stand-alone particle selection model has a higher perplexity than the other stand-alone particle models. The particle selection model also assigned a higher probability than the word model to 32% of words in the `eval-test` data that had occurred less than five times in the training data which also partly explains the model's smoothing effect.

Back-off case	PP_{part}	PP_{word}
3	78.9	66.4
3-2	575	429
3-2-1	1.75×10^5	1.82×10^5

Table 6.20 *Russian corpus (430k): word-level perplexity of word model and stand-alone particle selection model with* l_{max}*=3, computed on sets of events that were predicted by different word model back-off cases.*

Table 6.21 shows the average maximum number of words used in the context for predicting particles in words of the Russian `eval-test` data. The values are almost identical to those obtained for the word decomposition model with $l_{max} = 3$ which reflects the similar nature and lengths of the particle modelling units that were used, even though word decompositions were identical for only 11% of words in the vocabulary.

Max. words in context	Percentage
1	31.4
2	54.7
3	13.1
4	0.8
5	0.0

Table 6.21 *The average maximum number of words used to predict words in Russian* `eval-test` *data using the particle selection model with* $l_{max} = 3$.

A 4-gram particle model was built (in the same manner as for the analysis of the word decomposition model) with all singleton events discarded, and interpolated with a word 4-gram which had also had all singleton events removed. The interpolated model gave a 6.8% perplexity reduction over the 4-gram word model so it is safe to conclude that the improvement is due to smoothing between the particle and word model probabilities rather than the inclusion of useful higher order dependencies. As yet more confirmation, interpolating the $l_{max} = 3$ particle selection model with the pruned word 6-gram gave a 7.2% reduction in perplexity over the stand-alone word 6-gram.

6.3.4 Effect of Normalization

All the perplexity results for the particle models have been given as an upper bound on the perplexity of those models since the normalization constant in Equation (6.2) was set to one. To evaluate the effect of this approximation on the perplexity figures, each of the best performing models on the Russian data will be examined. However, due to the burden of computing the normalization constant, calculations will only be computed on the first 5,000 sentences ($\approx$ 50,000 words) of the Russian `eval-test` and `dev-test` data. Since it is only the relative changes in perplexity that are of interest, this is expected to be acceptable. The perplexity on the smaller data set for the baseline word trigram model is 614.9 (cf. $PP_{word} = 677.0$ on entire `eval-test` data). The perplexities of the affixes, word decomposition ($l_{max} = 3$) and particle selection ($l_{max} = 3$) models for both normalized and unnormalized probabilities are given in Table 6.22.

Clearly, there is not a large difference between the normalized and unnormalized perplexities of the stand-alone particle models (up to 2.5%) nor between the interpolated models (up to 0.7%). Moreover, normalization will always improve a model's perplexity. Since the effect of normalization is relatively small, using the approximation of $Z(h) = 1$ in all the above results

		Perplexity		weights	% improvement
Model	Normalization	PP^{6g}_{part}	PP^{6g}_{interp}	λ_{word}, λ_{part}	over word trigram
affixes	unnormalized	687.7	580.2	0.70, 0.30	5.6
	normalized	678.4	577.8	0.68, 0.32	6.0
word decomp.	unnormalized	712.6	574.4	0.70, 0.30	6.6
$l_{max} = 3$	normalized	697.7	570.7	0.67, 0.33	7.2
particle select.	unnormalized	741.6	574.8	0.71, 0.29	6.5
$l_{max} = 3$	normalized	723.3	570.8	0.68, 0.32	7.2

Table 6.22 *The effect of normalization on the three best performing interpolated word and particle models computed using the first 5000 sentences of the Russian* `eval-test` *data.*

is seen to be justified especially when consideration is made of the computational complexity involved in computing the correct value of $Z(h)$ for each context h.

6.3.5 Discussion

The performance of three different particle models has now been presented and analysed in some detail. To facilitate a comparison between the overall performance of all the models, the perplexity of the stand-alone and interpolated models is shown on the next page in Figure 6.5 for Russian and in Figure 6.6 for English. (Note that the scales on the y-axis of each plot are different.) Models' perplexities are plotted against the weighted average number of particles per word. This was chosen as a convenient means of relating the different models to each other.

For the stand-alone models the trend is observed of the perplexity decreasing as the average number of particles per word decreases. What is striking is that the stand-alone perplexities are almost identical for models that have a similar number of average particles per word in their decompositions. For the interpolated models there is a clear difference between the plots for the two languages. Convincing minima are obtained for the two data-driven algorithms on Russian whereas for English any minima that are present are much less evident. For English, the perplexity of the interpolated word decomposition model is more or less independent of the average number of particles per word and the interpolated particle selection model gets progressively better as the average number of particles per word gets less and as more higher-order word N-grams are included.

The results obtained with the particle selection algorithm perhaps indicate that the operation of the algorithm is better than that of the word decomposition algorithm. In particular, it would appear that it might be better to determine the effect that a particular particle has on all words simultaneously rather than to consider only the effect that a particle has on an isolated vocabulary word, as was the case for the word decomposition method. Moreover, the problem with algorithm initialisation is largely absent from the particle selection method yet it was of great importance with the word decomposition method.

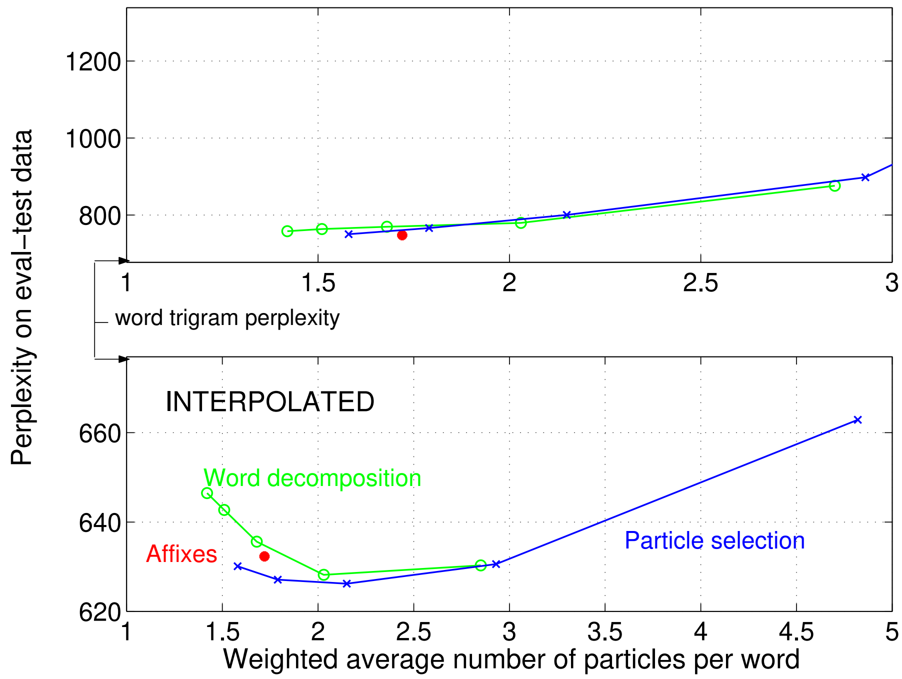


Figure 6.5 *Russian corpus (430k): summarised results for the different particle models.*

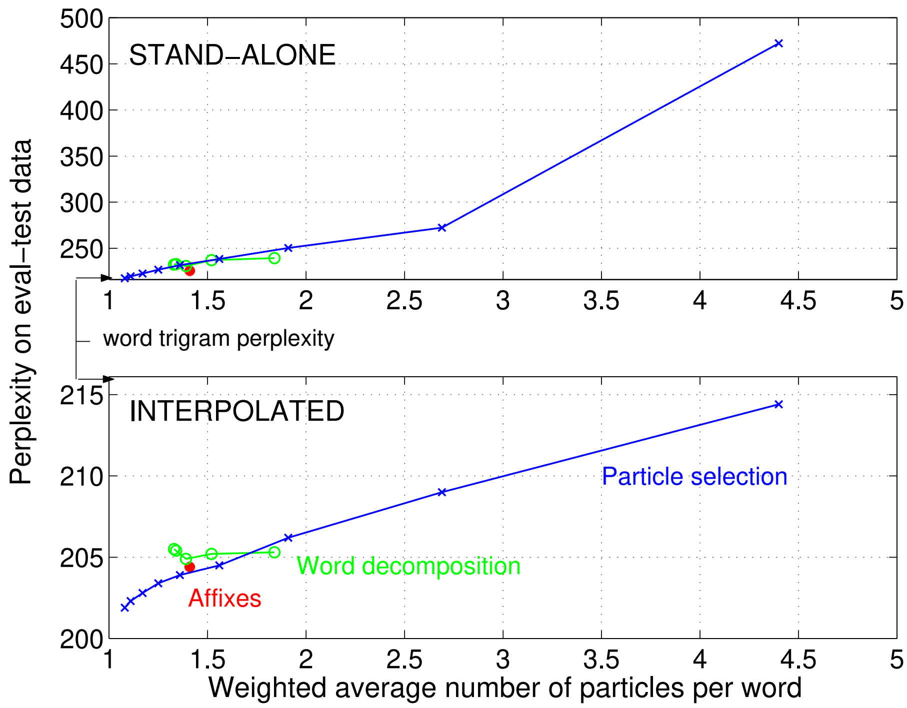


Figure 6.6 *English corpus (65k): summarised results for the different particle models.*

A final question that may be asked is whether the characteristics of the particle model complement those of both the word model and the one-sided class model that was investigated in the previous chapter. A three-way interpolation was performed with the word trigram model, and one-sided class model and particle model, each of which had given the best result when they were individually interpolated with the word trigram. The results are given in Table 6.23 for the appropriate models on Russian and English.

Language	Perplexity on `eval-test` $PP_{word+class+part}$	weights λ_{word},λ_{class},λ_{part}	% improvement over word trigram
Russian (430k)	548.2	0.44, 0.37, 0.19	19.0
English (65k)	191.8	0.33, 0.28, 0.39	11.2

Table 6.23 *Perplexities of combined word, one-sided class and particle selection models on* `eval-test` *data for the two corpora.*

Although the combined models each have around two and a half times the number of parameters as the word model, the perplexity reductions are significant. Moreover, the results also indicate that the characteristics of the three models are indeed complementary and the performance of the individual models is approximately additive.

The analyses that have been performed in this chapter suggest that the probabilities in the word model have indeed been smoothed by the probabilities from the particle model and that using the combined models in a speech recognition system should reduce the word error rate. It has been pointed out that such experiments were not carried out because there was no appropriate acoustic data available for either of the two language modelling corpora that were used. In addition, it is also worth mentioning that the use of 6-grams in the particle model need not present any difficulty during recognition. Unique paths need only be maintained for different *word* histories and since only a small percentage of particle 6-grams map to word 4-grams (13%) and longer (1%) this represents very little additional computational difficulty over using a word trigram language model. In fact, a greater number of paths can potentially be merged since paths ending in the same particle sequences can be considered identical.

6.4 Summary

This chapter has described a new language modelling technique that captures dependencies in language at a sub-word level using particles. Three different algorithms were proposed and investigated for determining the particle units automatically. The data-driven particle selection method was shown to perform best when the model built using its decompositions was combined with a word model. The decompositions obtained using the data-driven word decomposition algorithm performed similarly well to those obtained using the simple affix stripping algorithm. The particle-based technique, which was motivated by the morphology of Russian words, has also shown itself to be of use in modelling English. For English, the overall reduction in perplexity was similar to that obtained with the one-sided class model. For Russian, the reductions in perplexity were around half those obtained with the one-sided class model. However, combinations of the word, one-sided class and particle models produced further reductions in perplexity which showed that each component model had captured different dependencies.

7

Conclusions and further work

In this short chapter the main topics examined in this dissertation are first reviewed and the original contributions and important results are highlighted. The chapter concludes with some proposals for further work that have been prompted by the experiments conducted in this dissertation.

7.1 Review of experimental work

The principal difference between the work in this dissertation and other work on language modelling has been the examination and comparison of language models for Russian and English. The work began with an examination of the characteristics of a Russian corpus, which had been collected and prepared specially for the experiments, and a well-known English language corpus. The investigation of the variation of OOV-rate and the perplexity of word trigram models against vocabulary size established the basis of the subsequent experimental work. In particular, for Russian it was shown that a very large 430k vocabulary was required to achieve the same vocabulary coverage on held-out data as a 65k English vocabulary. A backoff word trigram model employing permutations of the word history was developed to exploit the Russian language's potential for free word ordering. However, the model's poorer performance together with the observation that the N-gram hit-rates for both languages were similar indicated that localised free word ordering did not pose a serious problem.

The remainder of the experimental work in the dissertation concentrated on reducing the perceived data sparsity problems in both corpora by investigating alternative modelling units to words for use in the N-gram framework.

7.1.1 Two-sided class language models

Two-sided class models have been the subject of considerable attention in the literature, however, the experiments in Chapter 4 were the first to apply such models to Russian and the first to compare automatically derived classes against part-of-speech classifications in terms of perplexity and word error rate.

It was well known that the more general dependencies captured by the class model are able to complement the specific nature of the word model if the two component models are combined. The best linear combination of two-sided class and word model for Russian reduced the perplexity by around twice as much (over 16%) as the best combination for English did. This was attributed primarily to the greater sparsity of the Russian corpus where the class model's ability to generalise resulted in greater perplexity improvements. An interesting observation for the Russian corpus was that the stand-alone 5004-class model had a perplexity 3.2% less than the word model even though it contained 10% fewer parameters. In contrast, none of the class models for the English corpus outperformed the word model. All experiments were necessarily restricted to the 65k vocabularies for each language since the scaling characteristics of the clustering algorithm for the two-sided model made it computationally impractical to cluster the 430k vocabulary.

In the second half of the chapter classifications obtained both automatically and using part-of-speech information were compared in terms of perplexity and speech recognition performance. When the class model was interpolated with the word model, the automatically generated classes were shown to be consistently superior to the linguistic classes in terms of word error rate, and the best interpolated model had a 7% lower word error rate than the word model alone.

7.1.2 One-sided class language models

The one-sided class N-gram model was developed in Chapter 5 as an extension of the work conducted on the two-sided class N-gram model. For the two-sided model it was discovered that the scaling properties of the clustering algorithm prevented the automatic classification of the very large 430k Russian vocabulary in a reasonable time. The realisation that the clustering algorithm for a one-sided class model scaled linearly in the number of vocabulary words and, more importantly, linearly in the number of classes recommended this particular direction of research. Moreover, although one-sided models had been mentioned occasionally in the literature, they did not appear to have been used in any experimental work, nor had the clustering algorithm ever been investigated.

Having described the different language model tools that were necessary to implement this class model and the new update procedure for the clustering algorithm, comparisons between the performance of the one-sided and two-sided class models were then made. In addition, one-sided class models were also built for the 430k Russian vocabulary for which classifications could now be obtained efficiently. Combinations of word and class models were investigated for the 65k vocabularies and significant improvements were obtained (up to 12.5% for Russian and 5.3% for English), however, these were slightly less than had been obtained with the best combination of two-sided class and word model. From this it was concluded that the two-sided class model was able to generalise better than the one-sided class model due to the more coarse-grained dependencies which it captures. In addition, large perplexity reductions (up to 16.2%) were obtained using the 430k Russian vocabulary by combining the one-sided class model with the word model. This could not easily have been accomplished for the two-sided class model and so represented a particularly significant result. Moreover, for the Russian corpus the stand-alone one-sided 5004-class model outperformed the word model by the same amount as the two-sided 5004-class model did, and also contained fewer parameters. Further perplexity improvements were obtained for all one-sided class models by determining separate position-dependent classification functions for each position of the context.

For situations where a very large vocabulary is required and where there is insufficient training data available, the stand-alone one-sided class N-gram model was shown to offer a competitive alternative to the word N-gram model. When very large vocabularies are required and two-sided class models cannot easily be generated, the favourable scaling properties of its clustering algorithm also make the one-sided class model an attractive solution.

7.1.3 Particle language models

The modelling of sub-word dependencies in language using particles was proposed in Chapter 6. Although the concept of modelling sub-word relationships was not new, the form of the model and the way in which particles and word decompositions were determined automatically was not known to have been investigated before. The motivation for particle-based language modelling arose from knowledge about the productive morphology of the Russian language. Nonetheless, the technique was also applied successfully to English and for both languages substantial reductions in perplexity were obtained when the particle model was combined with a word model.

A linguistics-based affix stripping method was compared against two data-driven methods which optimised the set of particles and word decompositions by maximising the likelihood of the training data. The data-driven word decomposition algorithm optimised the decomposition of each vocabulary word in turn given an initial decomposition for the word. This method was shown to perform better than the affixes model for Russian and similarly well for English. The second, data-driven particle selection algorithm selected particles iteratively according to how much their inclusion contributed to increasing the training data likelihood. This method was shown to perform best of the three methods on both languages. This was attributed in part to the way the algorithm considered the change in likelihood that including a particle had on all vocabulary words simultaneously. In contrast, the word decomposition algorithm only considered the effect that changing the decomposition of one vocabulary word at a time had on the likelihood.

The performance of the particle modelling approach in combination with word-based models for English was similar to that of the one-sided class models and gave reductions in perplexity of up to 5.4% (for a comparable particle model). The reductions in perplexity on Russian were also significant (up to 7.5%) but not as large as those obtained with the one-sided class models. However, combinations of word, one-sided class and particle models were found to give reductions in perplexity that were greater than those which either of the component combinations gave. The results that have been obtained with particle-based N-gram modelling of language clearly show that the technique has much to offer.

7.2 Suggested further work

The examination of the Russian language that has been conducted in this dissertation has shown a clear need for a new range of language modelling techniques that capture different kinds of language dependencies. Progress has been made in this area with the development of a class model formulation for which there was an efficient clustering algorithm, and also with the range of particle modelling techniques that were investigated. Any further work must include an examination of the recognition performance of the models that have been proposed. In addition to this, modelling language using particles has stimulated several ideas for other possible avenues of research to exploit the language dependencies at a sub-word level.

7.2.1 Recognition performance of models

All the language models that have been investigated in this dissertation have been designed with speech recognition applications in mind. However, the absence of suitable acoustic databases prevented an assessment of the recognition performance of the models. Since the one-sided class and particle modelling techniques have been shown to reduce perplexity on both the English and Russian corpora, it would now be a relatively simple matter to apply the techniques to other domains for which appropriate linguistic and acoustic data are available. Using class N-gram models to rescore lattices is no more difficult in principle than using a word N-gram model (Odell and Niesler, 1996). Some care must be taken when incorporating a particle N-gram model into the search to ensure that the search space of sentence hypotheses does not become unmanageable when particle N-grams represent long word N-grams. However, the standard techniques for efficient lattice rescoring, including path recombination and lookahead calculations, can all be easily implemented for the class and particle N-gram models.

7.2.2 Improved automatic particle selection

In addition to the algorithms described in Chapter 6, several other methods for selecting particles and determining word decompositions are highlighted below which, it is believed, might improve the performance of the combined word and particle model.

The affixes initialisation was not intended to define what could be achieved with a linguistic approach to particle-based language modelling, primarily because of the simplicity of the affix-stripping operation that was used. If suitable linguistically correct algorithms were to be produced for stripping known affixes from arbitrary words in Russian and English this could well improve the performance of the affixes model. However, the problem that the decompositions are not optimised for the model they are used in still remains and this is likely to limit the ultimate performance of a linguistics-based particle model. It is still unclear, for example, whether better performance would be obtained by decomposing words into all constituent morphs or into stems and inflections.

The word decomposition method was found to be particularly sensitive to the way in which it was initialised. When the affixes decompositions were used as the initialisation the optimisation even resulted in a reduction in model performance so this was obviously not a good initialisation. If sufficient computer memory and processor power were available, initialisations could be generated using all particle sequences of all lengths and importantly could include cross-word contexts. Not using cross-word contexts was a severe limitation to generating useful initial decompositions for the algorithm. It would also be interesting to use the decompositions obtained from the particle selection model to determine whether these could be improved by the application of this algorithm.

The particle selection algorithm, which was found to be the best performing algorithm, would undoubtedly benefit from a less restricted set of permitted particle replacements. These would allow particles that had been selected already to be broken up so that more useful particles might be formed in their place. The set of particle replacements that was considered was chosen due to the simplicity of the replacement operation. Changing this would increase the complexity of the algorithm but performance would almost certainly improve.

Both data-driven algorithms optimised word decompositions using a particle bigram criterion. Extending this criterion to higher order N-grams would take into account any longer-distance effects which are present in the final model. Although the bigram criterion was shown to be a good approximation for the final 6-gram models that were built, there is inevitably a discrepancy in the dependencies which are formed. In connection with this, some form of cross validation, parameter limiting or discounting scheme might also be incorporated into the algorithms to improve a model's ability to generalise to unseen data.

The use solely of the written orthography of words in determining particles and word decompositions may also limit the action of the data-driven algorithms. Much of the underlying morphology of a language is obscured by spelling changes that occur over time and using a purely statistical method to determine particles inevitably fails to exploit relationships hidden in this way. For Russian, a set of transformations exists for mapping words into a regularised orthographic form which largely removes the effect of spelling changes (Derwing and Priestly, 1980). Application of the transformations is simple and the use of regularised word forms in the algorithm in combination with the other improvements proposed above would be expected to improve the performance of the Russian particle N-gram models.

7.2.3 Particle trigger models

It is clear that the N-gram framework is perhaps not the optimal structure in which to employ particles as modelling units. The reduced span of the particle N-gram model is the main detractor in the N-gram approach. Moreover, the relationship between adjacent particles is unlikely to be as strong as that between adjacent words. In fact, strong relationships are expected to exist between particles in adjacent words, which would generally be outside the span of many particle N-grams. For example, the relationship between *boy* and *s* in the sentence "*The boy who kicks the ball.*" would not necessarily be captured by a particle trigram model but would be modelled implicitly by a word trigram model.

In Russian, agreement between morphological units in adjacent words is present to an even greater extent than in English. For example, there are strong relationships between certain verbal prefixes and the inflections of subsequent words, and between the inflections of all adjectives, participles and nouns. A preliminary investigation of particles has been obtained with the linguistic affixes method and used the average mutual information criterion (see Section 2.2.7.1) to select candidate triggers. Precisely those particle-particle relationships which were expected to be useful and many more intuitive ones besides were found using this method. Given the nature of such triggers (mainly syntactic), a within-sentence trigger model or distance particle bigram model (see Section 2.2.7) would both be suitable modelling frameworks since they permit longer range relationships to be captured. Such models would still have to be combined with one that could produce probabilities for whole words. This could be achieved with a combination of particle N-gram and particle trigger techniques, for example. An alternative particle selection scheme might also need to be considered to determine particles that are optimised for use with the particle trigger model.

7.3 Final summary

The central thesis that different language modelling approaches were required for Russian has been consistently demonstrated throughout this dissertation. The proposed techniques, two types of class model and the particle modelling scheme, were shown to be useful for language modelling of both Russian and English. This dissertation, in answering many of the questions that were posed at the outset, has itself prompted many more. These will hopefully provide the stimulus for further research in this area.

A

English and Russian affixes

Russian

The prefixes and suffixes stripped from words in the 430k Russian vocabulary are given in Tables A.1 and A.2 respectively. The affixes were chosen according to their perceived usefulness in a particle N-gram model. A grammar of Russian (Offord, 1993) was used to aid selection.

ДО-	НИ-	ПЕРЕ-	ПРО-
РАС-	ОБ-	ПО-	ВО-
РАЗ-	ОБЕС-	ПОД-	ВС-
РАЗО-	ОБЕЗ-	ПОДО-	ВЫ-
НА-	ОБО-	ПРЕД-	ВЗ-
НЕ-	ОТ-	ПРЕДО-	ВЗО-
НЕДО-	ОТО-	ПРИ-	ЗА-

Table A.1 *Prefixes stripped from Russian words in the affixes model.*

-Ю	-АЯМИ	-ЕМ	-ОМ	-СЯ
-ЮЮ	-АЙ	-ЕМСЯ	-ОМУ	-СЬ
-ЮСЬ	-АМ	-ЕМУ	-ОЙ	-У
-ЮСЬСЯ	-АМИ	-ЕТ	-Я	-УЮ
-ЮТ	-АЯ	-ЕТЕ	-ЯХ	-ЮТ
-ЮТСЯ	-АЯМ	-ЕТЕСЬ	-ЯМ	-ЮТСЯ
-А	-ТЬСЯ	-ЕТСЯ	-ЯМИ	-ЬЮ
-АЕ	-ТЬ	-ЕШЬ	-ЯЯ	-Ы
-АЕМ	-АЯМИ	-ЕШЬСЯ	-ЯЯСЬ	-ЫЕ
-АЕВ	-Е	-О	-ЯТ	-ЫХ
-АХ	-ЕГО	-ОЕ	-ЯТСЯ	-ЫЙ
-АИ	-ЕЙ	-ОГО	-ЫМИ	-ЫМ

Table A.2 *Suffixes stripped from Russian words in the affixes model.*

English

The prefixes and suffixes stripped from all English words for use in the *affixes* particle model are given in Tables A.3 and A.4 respectively. Affixes were chosen from an encyclopedia of the English language (Crystal, 1995) according to their perceived usefulness in the particle model.

ANTI-	EN-	MEGA-	POLY-	TELE-
ARCH-	EX-	MINI-	POST-	TRANS-
AUTO-	EXTRA-	MIS-	PRE-	TRI-
BI-	FORE-	MONO-	PRO-	ULTRA-
CO-	HYPER-	MULTI-	PROTO-	UN-
CONTRA-	IL-	NEO-	PSEUDO-	UNDER-
COUNTER-	IN-	NON-	RE-	UNI-
DE-	INTER-	OUT-	SEMI-	VICE-
DEMI-	INTRA-	OVER-	SUB-	
DI-	IR-	PALEO-	SUPER-	
DIS-	MAL-	PAN-	SUR-	

Table A.3 *Prefixes stripped from English words in the affixes model.*

-'RE	-ER	-IC	-LET	-SHIP
-'S	-ERY	-IFY	-LETS	-SHIPS
-ABLE	-ESE	-ING	-LIKE	-STER
-AGE	-ESQUE	-ISE	-LING	-STERS
-AL	-ESS	-ISH	-LINGS	-TION
-ANT	-EST	-ISM	-LY	-TIONS
-ATE	-ETTE	-ISMS	-MENT	-WARD
-ATION	-FUL	-IST	-N'T	-WARDS
-DOM	-FULS	-ITE	-NESS	-WISE
-ED	-HOOD	-ITES	-OCRACY	-Y
-EE	-HOODS	-ITY	-OR	
-EER	-IAL	-IVE	-OUS	
-EN	-IAN	-IZE	-S	

Table A.4 *Suffixes stripped from English words in the affixes model.*

Bibliography

Aarts, E. and Korst, J. (1990). *Simulated Annealing and Boltzmann Machines - A Stochastic Approach to Combinatorial Optimization and Neural Computing*. John Wiley and Sons Ltd.

Allen, J., Hunnicutt, M. S., and Klatt, D. (1987). *From Text to Speech: The MITalk System*. Cambridge University Press.

Andreyev, N. D. (1965). *Статистико-Комбинаторное Моделирование Языков (Statistical Combinatorial Modelling of Languages)*. Nauka Moscow, Leningrad.

Bahl, L. R., Brown, P. F., de Souza, P. V., and Mercer, R. L. (1989). A Tree-Based Statistical Language Model for Natural Language Speech Recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 37(7):1001–1008.

Bahl, L. R., Brown, P. F., de Souza, P. V., Mercer, R. L., and Nahamoo, D. (1991). A fast algorithm for deleted interpolation. In *Proceedings of the European Conference on Speech Communication and Technology*, Genova, Italy.

Bahl, L. R., Jelinek, F., and Mercer, R. L. (1983). A Maximum Likelihood Approach to Continuous Speech Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 5(2):179–190.

Baker, J. K. (1979). Trainable Grammars for Speech Recognition. In Klatt, D. H. and Wolf, J. J., editors, *Speech Communication Papers for the 97th Meeting of the Acoustical Society of America*, pages 547–550.

Baum, L. E. (1972). An Inequality and Associated Maximization Technique in Statistical Estimation of Probabilistic Functions of Markov Processes. *Inequalities*, 3:1–8.

Bellegarda, J. R., Butzberger, J. W., Chow, Y., Coccaro, N. B., and Naik, D. (1996). A Novel Word Clustering Algorithm based on Latent Semantic Analysis. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, Atlanta, USA.

Berger, A. L., Della Pietra, S. A., and Della Pietra, V. J. (1996). A Maximum Entropy Approach to Natural Language Processing. *Computational Linguistics*, 22(1):39–71.

Blasig, R. (1999). Combination of Words and Word Categories in Varigram Histories. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, Phoenix, USA.

Brown, P. F., de Souza, P. V., Mercer, R. L., Della Pietra, V. J., and Lai, J. C. (1992). Class-Based n-gram Models of Natural Language. *Computational Linguistics*, 18(4):467–479.

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

Caflisch, J. (1995). *Issues in Russian Linguistics*. University Press of America, Inc.

Cave, R. L. and Neuwirth, L. P. (1980). *Hidden Markov Models for Speech*, chapter “Hidden Markov Models for English”, pages 8–15. IDA-CRD, Princeton, USA.

Cerf-Dannon, H. and El-Beze, M. (1991). Three Different Probabilistic Language Models: Comparison and Combination. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, Toronto, Canada.

Charniak, E. (1993). *Statistical Language Learning*. The MIT Press, Cambridge, Massachusetts.

Chelba, C. and Jelinek, F. (1999a). Recognition performance of a structured language model. In *Proceedings of the European Conference on Speech Communication and Technology*, Budapest, Hungary.

Chelba, C. and Jelinek, F. (1999b). Structured language modelling for speech recognition. In *Proceedings of the Applications of Natural Languages to Information Systems*, Klagenfurt, Austria.

Clarkson, P. R. (1999). *Adaptation of Statistical Language Models for Automatic Speech Recognition*. PhD thesis, Cambridge University.

Clarkson, P. R. and Robinson, A. J. (1997). Language Model Adaptation using Mixtures and an Exponentially Decaying Cache. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, Munich, Germany.

Clarkson, P. R. and Rosenfeld, R. (1997). Statistical Language Modeling Using the CMU-Cambridge Toolkit. In *Proceedings of the European Conference on Speech Communication and Technology*, Rhodes, Greece.

Crystal, D. (1995). *The Cambridge Encyclopedia of the English Language*. Cambridge University Press.

Darroch, J. N. and Ratcliff, D. (1972). Generalized iterative scaling for log-linear models. *Annals of Mathematical Statistics*, 43:1470–1480.

Deligne, S. and Bimbot, F. (1995). Language Modelling by Variable length Sequences: Theoretical Formulation and Evaluation of Multigrams. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, Detroit, USA.

Deligne, S. and Bimbot, F. (1997). Inference of Variable-length Linguistic and Acoustic Units by Multigrams. *Speech Communication*, 23:223–241.

Deligne, S., Yvon, F., and Bimbot, F. (1995). Variable-length Sequence Matching for Phonetic Transcription using Joint Multigrams. In *Proceedings of the European Conference on Speech Communication and Technology*, Madrid, Spain.

Deligne, S., Yvon, F., and Bimbot, F. (1996). Introducing Statistical Dependencies and Structural Constraints in Variable-length Sequence Models. In Miclet, L. and de la Higuera, C., editors, *Grammatical Inference: Learning Syntax from Sentences, Lecture Notes in Artificial Intelligence 1147*, pages 156–157. Springer, Berlin.

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society*, 39(1):1–38.

Derwing, B. L. and Priestly, T. M. S. (1980). *Spelling Rules for Russian*. Slavica Publishers Inc.

Duda, R. O. and Hart, P. E. (1973). *Pattern Classification and Scene Analysis*, pages 227–228. Wiley, New York.

El-Beze, M. and Derouault, A.-M. (1990). A Morphological Model for Large Vocabulary Speech Recognition. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, Albuquerque, USA.

Ginzburg, E. (1989). *Within the Whirlwind (Krutoy Marshrut)*. Collins Harvill. translated from the Russian by Boland, I.

Good, I. J. (1953). The Population Frequencies of Species and the Estimation of Population Parameters. *Biometrika*, 40(3,4):237–264.

GuoDong, Z. and KimTeng, L. (1999). Interpolation of N-gram and Mutual-information based Trigger Pair Language Models for Mandarin Speech Recognition. *Computer Speech and Language*, 13:125–141.

Jardino, M. (1996). Multilingual Stochastic n-gram Class Language Models. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, Philadelphia, USA.

Jardino, M. and Adda, G. (1993a). Automatic Word Classification using Simulated Annealing. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, Minneapolis, USA.

Jardino, M. and Adda, G. (1993b). Language Modelling for CSR of Large Corpus using Automatic Classification of Words. In *Proceedings of the European Conference on Speech Communication and Technology*, Berlin, Germany.

Jardino, M. and Adda, G. (1994). Automatic Determination of a Stochastic Bi-gram Class Language Model. *Proceedings of ICGI'94*.

Jelinek, F. (1990). *Readings in Speech Recognition*, chapter "Self-organized Language Modeling for Speech Recognition", pages 450–506. Morgan Kaufmann.

Jelinek, F. (1997). *Statistical Methods for Speech Recognition*. The MIT Press, Cambridge, Massachusetts.

Jelinek, F. and Chelba, C. (1999). Putting Language into Language Modelling. In *Proceedings of the European Conference on Speech Communication and Technology*, Budapest, Hungary.

Jelinek, F. and Mercer, R. L. (1980). Interpolated Estimation of Markov Source Parameters from Sparse Data. In *Proceedings of the Workshop on Pattern Recognition in Practice*, Amsterdam, The Netherlands.

Jelinek, F., Merialdo, B., Roukos, S., and Strauss, M. (1991). A Dynamic Language Model for Speech Recognition. In *Proceedings of Speech and Natural Language DARPA Workshop*.

Kanevsky, D., Monkowski, M., and Sedivy, J. (1996). Large Vocabulary Speaker-Independent Continuous Speech Recognition in Russian Language. In *Proceedings of Speech and Computer (SPECOM96)*, St. Petersburg, Russia.

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

Klakow, D. (1998). Log-linear Interpolation of Language Models. In *Proceedings of the International Conference on Spoken Language Processing*, Sydney, Australia.

Kneser, R. and Ney, H. (1993). Improved Clustering Techniques for Class-based Statistical Language Modelling. In *Proceedings of the European Conference on Speech Communication and Technology*, Berlin, Germany.

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

Kuhn, R. and De Mori, R. (1992). Corrections to 'A Cache-Based Language Model for Speech Recognition'. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(6):691–692.

Lafferty, J., Sleator, D., and Temperley, D. (1992). Grammatical Trigrams: A Probabilistic Model of Link Grammar. Technical Report CMU-CS-92-181, School of Computer Science, Carnegie Mellon University.

Lari, K. and Young, S. J. (1990). The Estimation of Stochastic Context-free Grammars using the Inside-Outside Algorithm. *Computer Speech and Language*, 4:35–56.

Lau, R., Rosenfeld, R., and Roukos, S. (1993). Trigger-based Language Models: A Maximum Entropy Approach. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, Minneapolis, USA.

Lloyd-Thomas, H., Wright, J. H., and Jones, G. J. F. (1995). An Integrated Grammar/Bigram Language Model using Path Scores. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, Detroit, USA.

Maltese, G. and Mancini, F. (1992). An Automatic Technique to include Grammatical and Morphological Information in a Trigram-based Statistical Language Model. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, San Fransisco, USA.

Martin, S., Liermann, J., and Ney, H. (1995). Algorithms for Bigram and Trigram Word Clustering. In *Proceedings of the European Conference on Speech Communication and Technology*, Madrid, Spain.

Martin, S., Liermann, J., and Ney, H. (1998). Algorithms for bigram and trigram word clustering. *Speech Communication*, 24:19–37.

Nadas, A. (1985). On Turing's Formula for Word Probabilities. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 33(6):1414–1416.

Ney, H., Essen, U., and Kneser, R. (1994). On Structuring Probabilistic Dependences in Stochastic Language Modelling. *Computer Speech and Language*, 8:1–38.

Ney, H., Essen, U., and Kneser, R. (1995). On the Estimation of 'small' Probabilities by Leaving-one-out. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(12):1202–1212.

Ney, H. and Ortmanns, S. (1999). Dynamic Programming Search for Continuous Speech Recognition. *IEEE Signal Processing Magazine*, pages 64–83.

Niesler, T. R., Whittaker, E. W. D., and Woodland, P. C. (1998). Comparison of Part-of-speech and Automatically Derived Category-based Language Models for Speech Recognition. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, Seattle, USA.

Niesler, T. R. and Woodland, P. C. (1996a). A Variable-length Category-based n-gram Language Model. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, Atlanta, USA.

Niesler, T. R. and Woodland, P. C. (1996b). Combination of Word-based and Category-based Language Models. In *Proceedings of the International Conference on Spoken Language Processing*, Philadelphia, USA.

Niesler, T. R. and Woodland, P. C. (1997). Modelling Word-pair Relations in a Category-based Language Model. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, Munich, Germany.

Odell, J. J. and Niesler, T. R. (1996). *Lattice and Language Modelling Toolkit V2.0*. Entropic Cambridge Research Laboratories Inc.

Offord, D. (1993). *Modern Russian: An Advanced Grammar Course*. Bristol Classical Press.

Paul, D. B. and Baker, J. M. (1992). The Design for the Wall Street Journal-based CSR corpus. In *Proceedings of the International Conference on Spoken Language Processing*, Banff, Canada.

Placeway, P., Schwartz, R., Fung, P., and Nguyen, L. (1993). The Estimation of Powerful Language Models from Small and Large Corpora. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, Minneapolis, USA.

Porter, M. F. (1980). An Algorithm for Suffix Stripping. *Program*, 14(3):130–137.

Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P. (1992). *Numerical Recipes in C - The Art of Scientific Computing (second edition)*. Cambridge University Press.

Rabiner, L. R. (1989). *Readings in Speech Recognition*, chapter "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition", pages 267–296. Morgan Kaufmann.

Ries, K., Dag Buo, F., and Waibel, A. (1996). Class Phrase Models for Language Modelling. In *Proceedings of the International Conference on Spoken Language Processing*, Philadelphia, USA.

Robinson, A. J. and Fallside, F. (1991). A Recurrent Error Propagation Network Speech Recognition System. *Computer Speech and Language*, 5(3):259–274.

Rosenfeld, R. (1994). *Adaptive Statistical Language Modelling: A Maximum Entropy Approach*. PhD thesis, School of Computer Science, Carnegie Mellon University. Technical Report CMU-CS-94-138.

Schwartz, R. and Chow, Y.-L. (1990). The N-best Algorithm: An efficient and exact procedure for finding the N most likely sentence hypotheses. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, Albuquerque, USA.

Seymore, K. and Rosenfeld, R. (1996). Scalable Backoff Language Models. In *Proceedings of the International Conference on Spoken Language Processing*, Philadelphia, USA.

Shaikevich, A. (1997). Personal communication.

Sleator, D. D. K. and Temperley, D. (1991). Parsing English with a Link Grammar. Technical Report CMU-CS-91-196, School of Computer Science, Carnegie Mellon University.

Stolcke, A. (1995). An Efficient Probabilistic Context-free Parsing Algorithm that Computes Prefix Probabilities. *Computational Linguistics*, 21(2):165–201.

Stolcke, A. (1998). Entropy-based Pruning of Backoff Language Models. In *Proceedings of the 1998 DARPA Broadcast News Transcription and Understanding Workshop*.

Ueberla, J. P. (1995). More Efficient Clustering of N-grams for Statistical Language Modeling. In *Proceedings of the European Conference on Speech Communication and Technology*, Madrid, Spain.

Ueberla, J. P. and Gransden, I. R. (1996). Clustered Language Models with Context-Equivalent States. In *Proceedings of the International Conference on Spoken Language Processing*, Philadelphia, USA.

Whittaker, E. W. D. and Woodland, P. C. (1998). Comparison of Language Modelling Techniques for Russian and English. In *Proceedings of the International Conference on Spoken Language Processing*, Sydney, Australia.

Witten, I. H. and Bell, T. C. (1991). The Zero-Frequency Problem: Estimating the Probabilities of Novel Events in Adaptive Text Compression. *IEEE Transactions on Information Theory*, 37(4):1085–1094.

Woodland, P. C., Hain, T., Johnson, S. E., Niesler, T. R., Tuerk, A., Whittaker, E. W. D., and Young, S. J. (1998). The 1997 HTK Broadcast News Transcription System. In *Proceedings of the 1998 DARPA Broadcast News Transcription and Understanding Workshop*.

Woodland, P. C., Leggetter, C. J., Odell, J. J., Valtchev, V., and Young, S. J. (1995). The 1994 HTK Large Vocabulary Speech Recognition System. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, Atlanta, USA.

Woodland, P. C., Odell, J. J., Valtchev, V., and Young, S. J. (1994). Large Vocabulary Continuous Speech Recognition using HTK. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, Adelaide, Australia.

Wright, J. H., Jones, G. J. F., and Wrigley, E. N. (1992). Hybrid Grammar-Bigram Speech Recognition System with First-order Dependence Model. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, San Fransisco, USA.

Yokoyama, O. T. (1985). *Issues in Russian morphosyntax*, volume 10 of *UCLA Slavic Studies*, chapter "A Diversified Approach to Russian Word Order", pages 187–208. Slavica Publishers, Inc.

Young, S. J., Valtchev, V., and Odell, J. J. (1997). The HLM Book (*for HLM V1.0*). (unpublished).