Portability of a Class-based Backoff Language Model

by

Angel Xuan Chang

S,B., Electrical Engineering and Computer Science
Massachusetts Institute of Technology, 1999.

Submitted to the Department of Electrical Engineering and Computer Science

in Partial Fulfillment of the Requirements for the Degree of

Master of Engineering in Electrical Engineering and Computer Science

at the Massachusetts Institute of Technology

February 1, 2000

Author_____
Department of Electrical Engineering and Computer Science
February 1, 2000

Certified
by_____
James R. Glass
Thesis Supervisor

Accepted
by_____
Arthur C. Smith
Chairman, Department Committee on Graduate Theses

Portability of a Class-based Backoff Language Model

by

Angel Xuan Chang

Submitted to the
Department of Electrical Engineering and Computer Science

February 1, 2000

In Partial Fulfillment of the Requirements for the Degree of
Master of Engineering in Electrical Engineering and Computer Science

ABSTRACT

In this thesis, we explore how we can combine the advantages of both word and class n-grams by backing off from a word n-gram to a class-based language model (LM). In particular, we are interested in whether the resulting LM will prove to be more robust to slight changes in domain. For our purposes, words are classified based on their part-of-speech (POS) tags. We discuss the incorporation of our class based LM into the search phase of a speech recognition system. In particular, we discuss the creation of a simple online tagger and its performance as compared to the Brill POS tagger originally used to tag the training text. We also present a series of perplexity and recognition experiments to investigate the performance of our class based LM across a variety of broadcast shows. In addition, we examine the effect of out-of-vocabulary (OOV) words on the recognition accuracy and a feature-based LM that considers the separation of root and stem.

Thesis Supervisor: James R. Glass
Title: Principal Research Scientist

Acknowledgements

I would like to thank my advisor, Dave Goddeau, and the rest of the speech group, Pedro Moreno, Beth Logan, and Jean-Manuel Van Thong, at Compaq Computer Corporation's Cambridge Research Lab (CRL) for their invaluable guidance and insights. I would also like to thank Jim Glass, my thesis supervisor at MIT, for his help and support.

In addition, this work would not be possible without the generous support of Compaq Computer Corporation. CRL not only provided the necessary funding and resources, by also a great working environment with wonderful people.

Finally, I want to thank my family and friends for their continuous encouragement and support, to my friend Meena Bharwani for always believing in me and for always being there to make me laugh, and to my parents for all their love and understanding.

Table of Contents

ABSTRACT **2**

ACKNOWLEDGEMENTS **3**

TABLE OF CONTENTS **4**

LIST OF FIGURES AND TABLES **6**

CHAPTER I INTRODUCTION **7**

1.1 OVERVIEW 7
1.2 MOTIVATION 8
1.3 PRIOR WORK 9
1.4 DOCUMENT OUTLINE 10

CHAPTER II BACKGROUND INFORMATION **11**

2.1 LANGUAGE MODELING 11
2.1.1 Information Theory and Speech Recognition *11*
2.1.2 Word n-Grams *13*
2.1.3 Smoothing Techniques *14*
2.1.4 Evaluating Language Models *15*
2.2 PREVIOUS WORK ON CLASS BASED LANGUAGE MODELING 16
2.2.1 Class n-grams *16*
2.2.2 Classification Schemes *18*
2.2.3 Combining Class and Word N-Grams for LVCSR *20*

CHAPTER III CLASS MODELS **23**

3.1 INTRODUCTION 23
3.2 LANGUAGE MODELS 23
3.2.1 Class and Word N-grams *23*
3.2.2 Bigram Backoff Model *24*
3.3 IMPLEMENTATION 26
3.3.1 Overview *26*
3.3.2 Brill Tagger *27*
3.3.3 Other Implementation Issues *28*
3.4 PERPLEXITY EXPERIMENTS 29
3.4.1 Domain and Data *29*
3.4.2 Baseline perplexity results *29*
3.4.3 Class and Word n-gram models *30*
3.4.5 Bigram Backoff Model *32*

CHAPTER IV RECOGNITION EXPERIMENTS **34**

4.1 INTRODUCTION 34
4.2 BASELINE SYSTEM 35

4.3 ONLINE TAGGER 35
4.3.1 Tagger Design *36*
4.3.2 Tagger Experiments *37*
4.4 INTEGRATION INTO SEARCH 38
4.4.1 Overview of Search *38*
4.4.2 Class-based Backoff Nodes *39*
4.5 RECOGNITION RESULTS 40
4.5.1 Class Bigram Backoff Experiments *41*
4.5.2 Unigram Backoff Experiment *42*
4.5.3 Bigram Reduction Experiment *43*
CHAPTER V PORTABILITY 44
5.1 INTRODUCTION 44
5.2 PORTABILITY EXPERIMENTS 44
5.2.1 Motivation *44*
5.2.2 Data Description *45*
5.2.3 Perplexity and Recognition results *45*
5.2.4 Analysis *47*
5.3 OOV EXPERIMENTS 48
5.3.1 Motivation *48*
5.3.2 Method *49*
5.3.3 Results and Analysis *50*
5.3.4 Future Work *51*
5.4 FEATURE BASED LANGUAGE MODEL 51
5.4.1 Motivation *51*
5.4.2 Method *52*
5.4.3 Results and Analysis *53*
5.4.4 Future Work *54*
CHAPTER VI CONCLUSIONS AND FUTURE WORK 55
6.1 SUMMARY 55
6.2 FUTURE WORK 56
APPENDIX A: SUMMARY OF CLASSES 58
APPENDIX B: SUMMARY OF DISCOUNTING METHODS 59
REFERENCES 60

List of Figures and Tables

FIGURE 2.1. SOURCE CHANNEL MODEL OF SPEECH RECOGNITION 12
FIGURE 3.1. BACKOFF PATHS FOR BIGRAM BACKOFF MODEL 25
FIGURE 3.2. COMPONENTS IN CREATING A CLASS-BASED LANGUAGE MODEL 26
TABLE 3.1. SUMMARY OF CORPUS DATA 29
TABLE 3.2. PERPLEXITIES FOR WORD TRIGRAMS USING DIFFERENT BACKOFFS 30
TABLE 3.3. BIGRAM AND TRIGRAM PERPLEXITIES FOR CLASS AND WORD LMS 30
TABLE 3.4A. BIGRAM PERPLEXITIES FOR INTERPOLATION OF WORD AND CLASS LMS 31
TABLE 3.4B. TRIGRAM PERPLEXITIES FOR INTERPOLATION OF WORD AND CLASS LMS 31
TABLE 3.5. NUMBER OF PARAMETERS USED IN EACH LANGUAGE MODEL. 31
TABLE 3.6. PERPLEXITIES FOR DIFFERENT BIGRAM BACKOFF PATHS 32
TABLE 3.7. PERPLEXITIES FOR BIGRAM BACKOFF MODELS WITH CUTOFFS OF 0, 10, 100 33
TABLE 3.8. PERPLEXITY RESULTS FOR DIFFERENT TRIGRAM MODELS 33
TABLE 4.1. ACCURACY OF TAGGER FOR DIFFERENT CUTOFFS 37
TABLE 4.2. COMPLEXITY OF TAGGER FOR DIFFERENT CUTOFFS 37
TABLE 4.3. AFFECT OF CUTOFFS FOR ONLINE TAGGER ON PERPLEXITY 38
TABLE 4.4. PERPLEXITY RESULTS FOR H4E97 41
TABLE 4.5. WER RESULTS FOR CONSTANT BACKOFFS ON H4E97 42
TABLE 4.6. WER OF REDUCED BIGRAM ON H4E97 43
TABLE 5.1. SUMMARY OF SHOWS AND THEIR CONTENT 45
TABLE 5.2. PERPLEXITIES FOR TALKRADIO AND ZDTV 46
TABLE 5.3. WER FOR DR. LAURA, ARTBELL, AND ZDTV 46
TABLE 5.4. WER FOR ARTBELL USING DIFFERENT CLASS WEIGHTS. 47
TABLE 5.5. OUT OF VOCABULARY RATES FOR TALKRADIO AND ZDTV. 50
TABLE 5.6. WER FOR LM WITH OOV WORDS ADDED ON TALKRADIO AND ZDTV. 50
TABLE 5.7. WER FOR LM WITH OOV WORDS ADDED ON ZDTV 50
TABLE 5.8. ENTROPIES FOR FEATURE BASED LM 53

Chapter I Introduction

1.1 Overview

Language models play an essential role in speech recognition systems by providing the *a priori* probability distribution for a sequence of words. Although extremely simple, word bigrams and trigrams have proven to be both fast and robust, and provide state-of-the-art performance that has yet to be surpassed by more sophisticated models [20]. At the same time, they have several serious drawbacks and are limited in their power. Class n-grams have been introduced to reduce the necessary number of parameters, alleviate the sparse data problem, and add robustness. However, due to the limited constraint they provide, class n-grams by themselves can perform much worse than the traditional word n-gram.

In this thesis, we are interested how class n-grams can be combined with word n-grams to improve the word error rate for the broadcast news domain. We are going to work from the DARPA Broadcast News (Hub4) domain and explore the portability of our language model to other news sources. Sponsored by DARPA to "advance the state of the art in technologies required to automatically transcribe and extract information from recordings of radio and television broadcast news," the Hub4 corpus provides an important benchmark for the large vocabulary continuous speech recognition (LVCSR) research community. Since it features well prepared speech in a relatively clean environment as well as audio from a variety of acoustic environments and different speakers, the broadcast news domain serves as a natural development environment for LVCSR technology. Hub4 provides this research effort with a common standard and development data that includes transcriptions and audio data for a variety of broadcast news sources such as ABC news, CNN, and others [15].

Since Hub4 serves as an ideal source of data for the broadcast news domain, we use Hub4 to train our language models and then investigate the performance of our class-based language model on other domains. As our main focus, we concentrate on a bigram backoff model and the integration of this class based language model into the decoding search. To measure the quality of our language model, we conduct a variety of experiments to investigate portability issues and the effect of out-of-vocabulary words. We also examine a feature based language model that considers the separation of root and stem.

1.2 Motivation

The work in this thesis is motivated mainly by the desire to provide transcripts for audio shows. While the Hub4 domain serves as an important benchmark for the community, we are more interested in how well models developed for this domain generalize to other domains. In particular, we are interested in speech recognition accuracy for transcribing different shows on which the language model was not trained. We attempt to study this by measuring word error rates for several popular radio talk shows.

In these shows, we may be faced with vastly different acoustic conditions and language usage, such as different ways of speaking and different statistical distributions of words. For instance, two shows we are interested in is Artbell, which is very conspiracy theory oriented, and Dr. Laura, which is dominated by high-pitched female voices. In contrast, there is much less emphasis on news about stocks and politics, which make up the bulk of the data from Hub4. By exploring a class-based backoff model, we hope to combine the advantages of both word and class n-grams that will prove to be more robust across domains. We use a relatively simple model so that we can incorporate classes directly into the search phase of a two-pass recognition system. In addition, we explore a feature-based model that separates root and stem.

Another application of interest is the multimedia search and retrieval of audio using transcripts produced by the recognizer. For indexing and retrieval, out-of-vocabulary

(OOV) words prove to be an extremely perplexing problem. If we have keywords that are not part of the vocabulary, then the decoder will not be able to recognize important keywords and the user will not be able to locate the desired segments in the audio. Thus, we perform experiments to investigate the effect of OOV words on the word error rate. We use class-based models in the hopes that it will be easier to introduce new words into a class-based model rather than a traditional word N-gram.

1.3 Prior Work

Combinations of word and class n-grams have been shown to provide better perplexity by combining the advantages of both types of models [20]. While word n-grams provide important specific word-word relations, classes provide many other advantages. Using classes, it becomes much easier to extend a language model for use on a different corpus or with new words without having to retrain. Providing important linguistic information, classes can also be useful for natural language understanding. However, despite the research done on interpolating class models with word n-gram models and the existence of many different classification schemes, it is still unclear which type of classes work the best and what is the best method for combining them with word n-grams.

Because most research on class based language models has been done on smaller domains such as ATIS, we will focus on the broadcast news domain for the LVCSR problem. It is only recently that class-based language models have been applied to recognition experiments for the broadcast news domain. Unlike smaller domains such as ATIS which had much less training data available, it is still uncertain as to how well classes will work for the broadcast news domain. And even in the LVCSR community, most work has focused on the Hub4 domain only while we are more interested in recognition performance for a broader range of domains. Although the Hub4 domain serves as an important benchmark, we are more interested in how well models developed for Hub4 generalize beyond that particular domain.

Past work has also used class-based language models mainly as a refinement for multiple stage decoding systems such as the 1998 HTK Broadcast News Transcription

System from Cambridge University [38]. While studies of classes have been show to improve recognition performance in multi-pass systems, we are interested in the merit of classes on their own. In this thesis, we attempt to study the performance of class based language models in a two-pass system.

1.4 Document Outline

In this chapter, we gave a brief introduction to the problem of interest and some motivating factors behind this work. Chapter II gives some background on language modeling and speech recognition, including previous work on class based language models and some different classification schemes. In Chapter III, we describe how we plan to improve the word error rate for LVCSR by using a backoff model to combine word and class n-grams. Beside perplexity experiments on Hub4, we also perform recognition experiments, which are discussed in Chapter IV. We also discuss how this class-based model is integrated into the recognizer, focusing on the integration into the search and the creation of an online tagger. The performance of the class-based model on other domains is explored in Chapter V through a variety of perplexity and recognition experiments. In addition to portability experiments on TalkRadio and ZDTV, we also discuss experiments studying the effect of OOV words and a feature-based language model in Chapter V. Finally, in Chapter VI, we discuss how this work can be extended in future experiments.

Chapter II Background Information

2.1 Language Modeling

2.1.1 Information Theory and Speech Recognition

In speech recognition, we are faced with the problem of extracting a sequence of words from an audio signal. From our own experiences with natural language, we know that not all sequences of words are equally likely and that there will be some structure associated with the string of words. This understanding helps us determine what was likely to have been said and to eliminate ungrammatical sentences. Similarly, language models help constrain the search space of possible word sequences by attempting to use some of the structure and properties of natural language to describe what kind of sentences can be found in a language. These models can be very important in the performance of speech recognition system, especially when there is noise or acoustic ambiguity.

Languages have traditionally been modeled using grammars that describe how to form high-level structures from phrases and words. While grammars can provide important structural information, they are more suited for natural language processing than speech recognition. In speech recognition, language modeling is used mainly to constrain the search space of possible sentences. So far unstructured statistical models such as n-grams, which attempt to give the probability of encountering a certain sequence of words, have been much more effective than structured grammars. Despite attempts to incorporate grammatical structure and long-range information into language modeling, simple word n-grams remain the prevalent language model in most speech recognition systems.

Language models are also useful in many other applications outside of speech recognition. Probabilistic models can be used for applications such as handwriting recognition, machine translation, and spelling correction, all of which can be placed in the framework of the source-channel model of information theory.

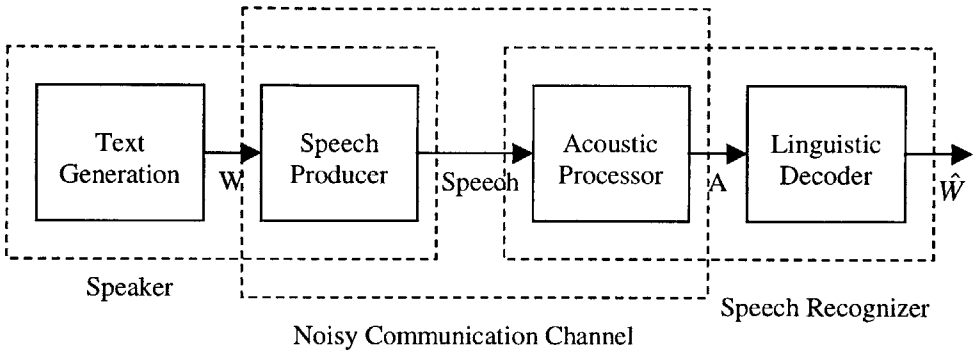

Figure 2.1. Source Channel Model of Speech Recognition

From the perspective of information theory, these are all problems of recovering information sent through a noisy channel. In the case of speech recognition, the original message is the string of words, W, generated by a person. After converting the sentence from text to speech, and some acoustic processing at the receiver, we can view W as having gone through a noisy channel to give the observed acoustic signal, A. Now the problem is to find the message or rather sequence of words $\hat{W}$ that maximize the probability that the original text is W given that the acoustic signal is A:

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

Using Bayes rule, we can represent $\hat{W}$ as:

$$\hat{W} = \arg\max_{W} P(W \mid A) = \arg\max_{W} \frac{P(W)P(A \mid W)}{P(A)} = \arg\max_{W} P(W)P(A \mid W)$$

The probability $P(A \mid W)$ characterizes the acoustic model and is determined using the acoustic models of the recognizer. The probability, $P(W)$, indicates how likely a person is to say a certain sequence of words and characterizes the information source acoustic channel. Calculating these *a priori* probabilities, $P(W)$, is the main concern of language modeling. In statistical language modeling, these parameters are estimated by training on a large amount of text.

Let $W = w_1^n \overset{def}{=} w_1, w_2, w_3, ..., w_n$, where w_1^n is a string of n words and w_i is the ith word in the string. Then

$$P(W) = \prod_{i=1}^{n} P(w_i \mid w_1, ..., w_{i-1}) = \prod_{i=1}^{n} P(w_i \mid w_1^{i-1}) = \prod_{i=1}^{n} P(w_i \mid h_i)$$

where $h_i = w_1^{i-1}$ and is known as the history. Since the event space for (h,w) is extremely large, we simplify the problem by separating the history h_i into equivalence classes $\Phi(h_i)$ and approximate $P(w_i \mid h_i)$ by $P(w_i \mid \Phi(h_i))$.

2.1.2 Word n-Grams

In the n-gram model, we assume that only the immediate history is important and use the last *n-1* words as equivalence classes:

$$\Phi(h_i) = w_{i-n+1}^{i-1} = w_{i-n+1}, ..., w_{i-1}.$$

Then we have

$$P(w_i \mid h_i) \approx P(w_i \mid \Phi(h_i)) = P(w_i \mid w_{i-n+1}, ..., w_{i-1})$$

For bigrams (*n=2*) and trigrams (*n=3*), the probability of seeing a word, w_i, given its history reduces to:

bigram: $P(w_i \mid h) \approx P(w_i \mid w_{i-1})$

trigram: $P(w_i \mid h) \approx P(w_i \mid w_{i-2}, w_{i-1})$

For LVCSR, only bigrams and trigrams are typically used since higher order n-grams increase the complexity of the model and the size of the search space substantially but do improve performance significantly.

From the training data, we can estimate these probabilities by calculating the frequency at which a n-gram occurs. If $c(w)$ is the number of times the sequence of words, *w*, occurred in the training set, then we can estimate the probability of seeing w_i given a history of $\Phi(h_i)$ by:

$$P(w_i \mid \Phi(h_i)) = P(w_i \mid w_{i-n+1}, ..., w_{i-1}) \approx f(w_i \mid w_{i-n+1}, ..., w_{i-1}) = \frac{c(w_{i-n+1}^{i})}{c(w_{i-n+1}^{i-1})}$$

Unfortunately, this estimation can be very bad for n-grams that occur only infrequently or even not at all in the training data. For n-grams that never occur in the training set, this language model will seriously underestimate their probabilities and assign them all probabilities of zero, even though these sequences can still occur outside of the training data. For large vocabulary sizes, this sparse data problem is unavoidable. In large vocabulary recognition systems today, the vocabulary size V is on the order of 60,000. That means that there are over $V^3 = 2.16 \times 10^{14}$ possible trigrams, while the typical size of the training set is much smaller. Obviously some of the possible trigrams are going to appear only a few times or not all in the training data. For these words, our estimate of their probabilities will be highly inaccurate. To adjust for this problem, it is necessary to smooth n-gram parameters [20].

2.1.3 Smoothing Techniques

There are a variety of different techniques for smoothing an n-gram. In his Ph.D. thesis [8], Stanley Chen gives a description and comparison of some common smoothing methods. One approach is to do a linear interpolation of the different n-gram models.

$$P(w_i \mid \Phi(h_i)) \approx \sum_j \lambda_j f(w_i \mid \Phi_j(h_i)) \qquad \sum_j \lambda_j = 1$$

In a trigram model, linear interpolation gives:

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

The parameters, λ_j, can be estimated using either deleted interpolation or by reserving a section of the training data for held-out interpolation [20].

Another common smoothing technique is to use the original n-gram estimate if it is reliable but backoff to lower order n-grams otherwise. If there is enough data for a particular n-gram, say if it occurs more than α times, then training data will probably give a good estimate so we can use the frequency as it is. If the n-gram occurs at least once but less than α times, then we use a discounted frequency. Since there is not enough training data for the n-gram to give a good estimate, its frequency will not be very reliable and so we take away from the probability to distribute it for n-grams which we have no training

data. For those words, we back off to a lower n-gram that will hopefully provide a better estimate. In general, we have

$$P(w_i \mid \Phi_j(h_i)) \approx \begin{cases} f(w_i \mid \Phi_j(h_i)) & c(\Phi_j(h_i)) > \alpha \\ f_d(w_i \mid \Phi_j(h_i)) & 0 < c(\Phi_j(h_i)) < \alpha \\ q(w_i) P(w_i \mid \Phi_{j-1}(h_i)) & c(\Phi_j(h_i)) = 0 \end{cases}$$

where $f_d(w_i \mid \Phi_j(h_i))$ is the discounted count for w_i, and $q(w_i)$ are chosen appropriately so that the sum of the probabilities add up to one. The parameters of the model can be estimated using the Good-Turing estimate, or other methods discounting methods such as absolute or Witten-Bell discounting [10].

2.1.4 Evaluating Language Models

To compare different language models, we need to be able to measure the quality of a language model. Perplexity, which is related to entropy, is usually used to compare different language models. From information theory, entropy is defined to be the amount of randomness in the system and can be expressed as:

$$H(p) = -\sum_i p_i \log p_i$$

If p is the true source distribution and q is the source distribution as estimated by user, then we have:

$$H(p) \leq -\sum_i p_i \log q_i$$

The right hand side of the equation is known as the logprob and can be interpreted as entropy from the point of view of user. Since the user might misestimate the source distribution, the true entropy of the system will be less than the entropy perceived by the user. In language modeling, the logprob or LP measures the difficulty of the task from the recognizer's point of view and can be expressed as:

$$LP = -\sum_i p_i \log q_i = -\frac{1}{n} \sum \log \hat{P}(W)$$

Perplexity is then defined as $PP = 2^{LP}$ and can be viewed as the average branching factor of the model. It gives the average number of words that the recognizer has to choose from

each time. Intuitively, if we can reduce the perplexity, then the recognizer will have fewer words to choose from and should give a better performance.

However, an improvement in perplexity does not necessarily correlate to an improvement in the recognizer itself. Ultimately in speech recognition, we are concerned with the accuracy of the recognizer, which is dependent not only on the language model but how it interacts with the other components of the system. The accuracy of the system is defined by the word error rate (WER), which is the total number of deletions D, insertions I, and substitutions S, over the total number of words N.

$$WER = \frac{S + I + D}{N}$$

Even though lower perplexity does tend to correlate with a lower word error rate, large improvements in perplexity do not always lead to improvements in word error rate. Often large improvements in perplexity can lead to little or even no improvement at all [8].

Despite the dubious correlation between perplexity and word error rates, perplexity results remain more prevalent because running recognition experiments can be extremely time consuming while perplexity experiments are much easier and faster to perform. In addition, calculating1 the word error rate is not without its own pitfalls. Since the word error rate is also a function of the speech recognition system itself, the final accuracy may be influenced not only by the performance of the language model, but also by its integration into the system. It can reflect the combination of the language model with other components such acoustic models, lexical models and search. Thus, perplexity remains a useful and dominant gauge of how good the language model is.

2.2 Previous Work on Class based Language Modeling

2.2.1 Class n-grams

While word n-grams have proven to be extremely powerful and are currently used in many state-of-the-art speech recognition systems, the word n-gram model is limited in many ways. By its very definition, n-gram models can only provide local constraints and so do not incorporate any long-distance relationships. The word n-gram is also very inflexible and cannot be easily adapted to new words or new domains. For instance, in

order to introduce new words, we have to provide training data containing these new words and retrain the entire language model.

To address some of these issues, the idea of word n-grams can be extended to class n-grams. Instead of using single words as the basis for our language model, we separate words into different classes and build a language model using these classes. The class n-grams offer many advantages over conventional word n-grams. For one, class n-grams provide a much more concise representation than word n-grams and has fewer parameters. Classes can also compensate for sparse data since if a word never appears in the training data but the class does, then we can still obtain a reasonable estimate for the word using its class. Because of this, it is easier to introduce new words into class-based models than word-based models. In addition, class-based models are also more robust to changes in the domain. By attempting to capture some of the linguistic information and to exploit the syntactic structure of language, class n-grams also move us closer to natural language understanding.

To create class n-grams, we first partition a vocabulary of V words into C classes using function π, which maps a word, w_i to its class c_i. These classes can be created either manually or automatically. If each word is assigned to exactly one class, we can make the estimate

$$P(w_i \mid w_1^{i-1}) = P(w_i \mid c_i)P(c_i \mid c_1^{i-1}).$$

Then the probability of seeing a particular string of words, W, is:

$$P(W) = \prod_{i=1}^{n} P(w_i \mid w_1^{i-1}) = \prod_{i=1}^{n} P(w_i \mid c_i)P(c_i \mid c_1^{i-1})$$

When words are allowed to belong to multiple class, the formulation of the class n-gram becomes more complicated. To resolve the issue of multiple classification, we have to sum over all possible classes or to redefine the problem as to find the best class sequence in addition to finding the best possible word sequence [16]. Since words often have multiple roles, the increase in complexity is compensated by a more realistic model.

Although classes are a powerful tool, class n-grams tend to have a higher perplexity than word n-grams. For example, when using part-of-speech tags to form class n-grams, Sriniva reported a 24.5% increase in perplexity over a word-based model on

Wall Street Journal [34]. Niesler and Woodland [28] reported an 11.3% increase in perplexity and Kneser and Ney [22] a 3% increase for LOB corpus. In general, other classifications also resulted in an increase in perplexity because word n-grams can model the relationships between specific words. Often when there is enough data to train a good class n-gram model, there is often enough data to train an even better word n-gram model.

To take advantages of both types of language models, we can combine word and class n-grams. By using well-studied smoothing techniques such as interpolation or back-off models to combine word and class n-grams, research shows that classes do improve perplexity and word error rate. Work by various researchers indicated linear interpolation of the two types of language models resulted in lower perplexity. Niesler and Woodland also reported that using a backoff model to combine word and class n-grams also improves the perplexity [29]. While word-class n-grams do improve perplexity, it still remains unclear what is the best way to partition words into classes, especially for LVCSR tasks that are found in the broadcast news domain.

2.2.2 Classification Schemes

With a large amount of data, classes can be automatically derived from their statistical characteristics using data driven clustering. Usually, automatic clustering will assign unique classes to each word. Since there is no reliance on linguistic knowledge and any optimization criterion can be used, automatic clustering is very flexible and can be tailored to its use. One approach suggested by Brown et al is to do a bottom up clustering that attempted to minimize the loss of mutual information during merges [5]. Martin et al extended the Kneser and Ney's exchange clustering maximizing bigram perplexity to maximizing trigram perplexity [22] [25]. However, these methods can only guarantee a local optimum. Jardino and Adda used simulated annealing to give optimal classifications by overcoming local solutions [19]. They also extended their algorithm to allow for multiple classifications [20]. While Jardino claims that the algorithm can be used to optimize the classification of tens of millions of untagged words in a few hours [18], others have found his algorithm to give only a slight improvement in perplexity at the expense of dramatically increased computational costs [25]. It is interesting to note that

simulated annealing resulted in similar groupings for the French, English, and German corpora, composed of functions words, syntactical classes, semantic classes, and other combinations that reflected the short range context [18].

Since these methods all focus on optimizing perplexity for training set, the criterion used in the clustering process might not prove to be optimal for the test set. To resolve this potential problem, there have been efforts to imitate the test corpus. Mori, Nishimura, and Itoh suggested extending the deleted interpolation technique by dividing the learning corpus into k partial corpora, building k language models, and then evaluating word-class relation by using the average perplexity of all possible pairs of language models [24]. Despite these efforts, the main disadvantage of data driven clustering, the need for a large amount of data, remains. In addition, the classification scheme that results in the lowest word error rate does not always correspond to the one with the lowest perplexity reduction, as shown in [12].

An alternative approach to data driven clustering is to cluster using linguistic knowledge and properties. For example, words can also be classified based on their morphology by placing words with same prefix or suffix into the same class. Uebler and Niemann showed that the word error rate for Italian and German improved using automatic prefix and suffix decomposition over frequency and semantic classes [32]. However, while this approach is clearly advantageous for languages with many different inflectional endings, it is unclear how beneficial it will be for English. Words can also be classified by using a thesaurus based approach [2] or by attempting automated semantic analysis [3].

Currently, the most popular classification strategy in using linguistic knowledge is to use part-of-speech (POS) tags. Possible POS features and values capture syntactical information such as whether a word is noun, verb, adjective, or determiner. It can also model other features such as the gender of a word or whether it is singular or plural. Most work have shown POS classes to give slightly worse perplexity results than data driven clustering, probably because linguistic knowledge are not necessarily ideal for language modeling.

According to a comparison of POS and automatically derived class n-grams by Niesler and Woodland [30] [31], automatic clustering procedures gave larger performance improvements than POS classifications. The n-grams based on automatic clustering procedures resulted in both lower perplexity and word error rate, though the changes in word error rate were not that significant. They discovered that automatic clustering to be much more flexible than classification using POS tags. For example, automatic clustering allowed a variable number of different categories, which had a significant impact on performance. Initially, as the number of classes increases, there was an improvement in performance. But at a certain point, more classes lead to deteriorating performance. In their work, Niesler and Woodland also investigated the total contribution to the overall likelihood made by individual word-categories. They found that words with semantic content such as nouns and adjectives were harder to predict than syntactic function words such as prepositions, articles, conjunctions, and personal pronouns. This is not too surprising since the POS does not take into account the semantic information at all, and in order to improve the contribution of those words, we have to incorporate semantic information into the classification scheme.

2.2.3 Combining Class and Word N-Grams for LVCSR

Despite the plethora of classification techniques, it is still uncertain what is the best way to classify words for language modeling. Since words can be classified in many different ways with each providing disparate information about the underlying structure of the language, there is perhaps no one best method. Instead, a more complex model that integrates the different sources of information has been shown to outperform individual models. By combining these different classification schemes and techniques, language models can capture some of the more complex ways that words interact.

For instance, class n-grams and word n-grams have combined to utilize the advantages of both, offering a model with both word specific relationships and more robust class-based relationships for words with sparse data problems. In a backoff model from class to word n-grams, if c_w is the count using words, and c_c is the count using classes, the trigram probabilities can be estimated by:

$$P(w_i \mid w_{i-2}, w_{i-1}) \approx \begin{cases} f(w_i \mid w_{i-2}, w_{i-1}) & c_w(w_{i-2}, w_{i-1}, w_i) > \alpha \\ P(w_i \mid c_i) P(c_i \mid c_{i-2}, c_{i-1})) & c_w(w_{i-2}, w_{i-1}, w_i) < \alpha \quad \& \quad c_c(c_{i-2}, c_{i-1}, c_i) > 0 \\ q(w_i) P(w_i \mid w_{i-1}) & c_c(w_{i-2}, w_{i-1}, w_i) = 0 \end{cases}$$

In general, if $P_w(w_i \mid \Phi_j(h))$ and $P_c(w_i \mid \Phi_j(h))$ are probabilities from a word and class n-gram respectively, the overall model is given by:

$$P(w_i \mid \Phi_j(h)) \approx \begin{cases} P_w(w_i \mid \Phi_j(h)) & c_w(\Phi_j(h), w_i) > \alpha \\ P_c(w_i \mid \Phi_j(h)) & c_w(\Phi_j(h), w_i) < \alpha \quad \& \quad c_c(\Phi_j(h), w_i) > 0 \\ q(w_i) P(w_i \mid \Phi_{j-1}(h)) & c_c(\Phi_j(h), w_i) = 0 \end{cases}$$

Since there are many different ways we can backoff from a word n-gram to a class n-gram, we plan to compare some of them. The specific formulations and experiments we plan to investigate will be discussed in more detail in Section 3.2.

Similarly, different language models can also be combined. Past work has also shown that integrating different language models tends to improve perplexity and word error rate. Using maximum entropy, Rosenfeld [32] demonstrated that different information sources can be combined to create a better language model. In their work on three different probabilistic language models, Cerf-Danon and El-Bèze [6] combined a trigram model with models based on POS tags and morphological features. To integrate the different class models, Cerf-Danon and El-Bèze used the following hierarchical backoff model:

$$P(w_i \mid w_{i-2}, w_{i-1}) = \begin{cases} f(w_i \mid w_{i-2}, w_{i-1}) & c_w(w_{i-2}, w_{i-1}, w_i) > \alpha_w \\ P_{morph}(w_i \mid \Phi_{morph}(w_{i-2}, w_{i-1})) & c_{morph}(w_{i-2}, w_{i-1}) > \alpha_{morph} \\ P_{POS}(w_i \mid \Phi_{POS}(w_{i-2}, w_{i-1})) & c_{POS}(w_{i-2}, w_{i-1}) > \alpha_{POS} \\ q(w_i) P(w_i \mid w_{i-1}) & otherwise \end{cases}$$

The hierarchical structure follows a backoff model in which a word trigram was used when the related probabilities were reliable. Otherwise the model backed off first to a morphological level, and then to a grammatical or POS level. Their results using the hierarchical model showed that there were significant improvements in recognition performance when all three models where combined using either linear interpolation or a

hierarchical structure. Maltese and Mancini's pos-lemma model, which interpolated two class-based models, combined grammatical POS tags and morphological properties [23]. According to their results, the simultaneous presence of two different types of class models gave greater decrease in perplexity than just having one class-based model.

Chapter III Class Models

3.1 Introduction

In this chapter, we describe how we hope to effectively combine class and word n-gram to create a language model that is more robust across domains. The specific class-based language models that we plan to investigate are described in more detail in Section 3.2. In Section 3.3, we discuss the implementation of our class-based language models. Finally, we present some perplexity results using these models in Section 3.4.

3.2 Language Models

3.2.1 Class and Word N-grams

We will focus our investigation on several different methods for backing off from a word n-gram to a class n-gram. In our experiments, we will first compare perplexities for the following four types of models and then investigate combinations of the different models:

1.) A pure word n-gram (www):

Predicting the next word, w_i, given the word history, $\Phi_w(h)$:

$$P(w_i \mid h) = P(w_i \mid \Phi_w(h)) \approx \frac{c(w_{i-2}w_{i-1}w_i)}{c(w_{i-2}w_{i-1})}$$

2.) A pure class n-gram (ccc)

Predicting the next word, w_i, based on the c_i, given the class history, $\Phi_w(h)$:

$$P_c(w_i \mid h) = \sum_{c:w \in c} P(w_i \mid c_i) P(c_i \mid \Phi_c(h)) \approx \sum_{c:w \in c} P(w_i \mid c_i) \frac{c(c_{i-2} c_{i-1} c_i)}{c(c_{i-2} c_{i-1})}$$

3.) (ccw)

Predicting the next word, w_i, given the class history, $\Phi_c(h)$:

$$P_c(w_i \mid h) = P(w_i \mid \Phi_c(h)) \approx \frac{c(c_{i-2} c_{i-1} w_i)}{c(c_{i-2} c_{i-1})}$$

4.) (wwc)

Predicting the next word, w_i, by predicting the next class, c_i, given the word history, $\Phi_w(h)$:

$$P_c(w_i \mid h) = \sum_{c:w \in c} P(w_i \mid c_i) P(c_i \mid \Phi_w(h)) \approx \sum_{c:w \in c} P(w_i \mid c_i) \frac{c(w_{i-2} w_{i-1} c_i)}{c(w_{i-2} w_{i-1})}$$

Ideally, we would like to use the most specific model for which we had enough data, suggesting a hierarchy of backoffs, for instance from (1) to (3) or (4) to (2). However, such a model can become overly complex. Thus, we will only attempt to examine combinations (by backoff and linear interpolation) of model (1) with each of the other models.

3.2.2 Bigram Backoff Model

Besides exploring different combinations of class and word n-grams, our main focus is to investigate a bigram backoff model designed to allow us to systematically analyze different backoff paths. Using this model, we hope to gain some insight into why we don't have the bigram and backoff based on which word is not very well modeled.

In our bigram backoff model, we start with a normal bigram with $P(w_2|w_1)$ estimated using:

$$P(w_2 \mid w_1) = \frac{c(w_1 w_2)}{c(w_1)}$$

However, if there's not enough data for a good estimation of $P(w_2|w_1)$ we backoff to a class based model. From the equation above, it is obvious that we will be unable to

estimate $P(w_2|w_1)$ accurately when there is not enough counts for w_1w_2, i.e. $c(w_1w_2)$ is low. This implies that either $c(w_1), c(w_2)$ is small, or $c(w_1w_2)$ is just low despite w_1 and w_2 being both well modeled. If $c(w_1)$ is high, but $c(w_2)$ low, then we have enough data for w_1 to be able to estimate c_2 given w_1. From c_2, we can then estimate $P(w_2|w_1)$ by $P(c_2|w_1)P(w_2|c_2)$. Otherwise, if $c(w_2)$ is high, but $c(w_1)$ is low, then we can approximate $P(w_2|w_1)$ using $P(w_2|c_1)$. However, if we don't have enough data for either word, we then backoff to a purely class based model or a word unigram model if necessary. Thus, depending on the cause for the lack of w_1w_2 in the training data, we back off to a different class based model.

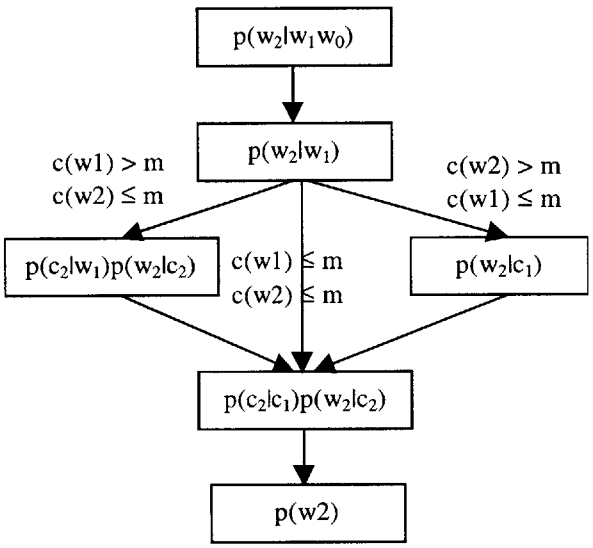

Figure 3.1. Backoff paths for bigram backoff model

While it is unclear whether this backoff strategy offers any advantages over more traditional direct backoffs, through this model we hope to see whether the backoff path matters, and if so which backoff path is the best. We chose to focus on a bigram backoff model instead of a trigram backoff model since it was much simpler to break down the possible backoff paths. In addition, a large percentage of useful information is already included in bigrams and it is a simple matter to add a word trigram on top of our bigram backoff model.

As a possible extension to our model, if counts for both w_1 and w_2 are extremely low, then we can choose to backoff to a very small probability. Thus, we attempt to address the problem of overestimating bigram pairs that don't occur together although individually they may have high unigram counts. Under normal bigram models, these pairs are given a relatively high probability. In our model, such pairs can be forced to have low probabilities.

3.3 Implementation

3.3.1 Overview

In order to create a class based language model, the training data was first tagged with the appropriate part-of-speech (POS) using the Brill tagger [4]. We chose to classify words based on their part-of-speech (POS) to make use of the syntactic information inherent in languages through classes. Using POS tags had the additional advantage that they also provided some stemming information for our feature based model. The Brill tagger will be described in more detail in Section 3.3.2.

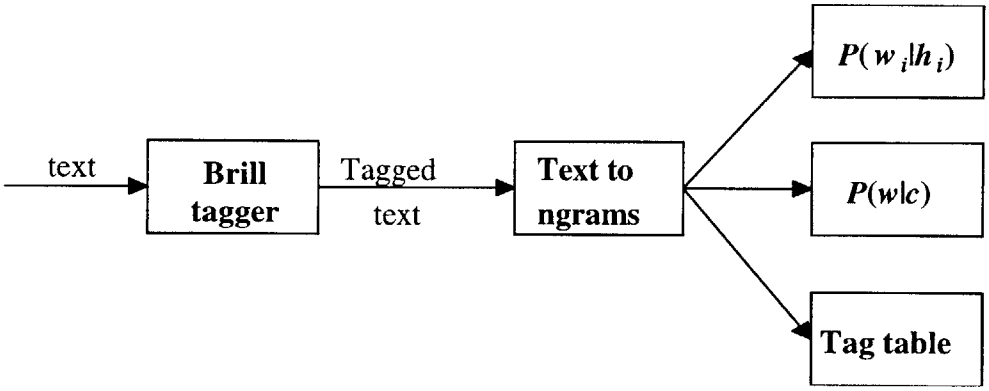

Figure 3.2. Components in creating a class-based language model

Statistics for creating language models are then gathered from the tagged text using CMU's Statistical Language Modeling Toolkit and other tools built on top of it to support classes [10]. Using these statistics, we calculated $P(w|c)$ and created different language models for calculating $P(w_i|h_i)$ as described in Section 3.2.2. The calculations for $P(w|c)$ are described in more detail in Section 3.3.3. These statistics are also used to

create a tag table for an online tagger (see Section 4.3) that allows for direct integration of classes into the search of the decoder.

3.3.2 Brill Tagger

To tag our training text with the proper part-of-speech, we used Eric Brill's rule based POS tagger which is able to achieve an accuracy of 96-97% [4]. We used a lexicon of 78218 words with 62 possible parts of speech. The lexicon was originally trained on data from Brown and the Wall Street Journal (WSJ) and had an average of 1.23 possible tags per word. The POS tags are an extended subset of the Penn Treebank tag-set [24], and include limited information on stem and tense. For instance, we had six different possible verb forms:

POS tag	Description	Example
VB	verb, base form	take
VBD	verb, past tense	took
VBG	verb, gerund/present participle	taking
VBN	verb, past participle	taken
VBP	verb, singular present, non-3rd	take
VBZ	verb, 3rd person singular present	takes

In addition to the normal POS tags that were part of the Penn Treebank tag-set, we include special tags for the two context cues, <s> and </s>, which are used to mark the beginning and end of each sentence. We also distinguish between nouns and their possessives (NN vs. NN's and NNP vs. NNP's). Common words such as IT'S, WE'RE, are also given their own tag. The main reason for these additions to the tag set is to bridge the difference between the Brill tagger and the language modeling toolkit's treatment of the apostrophe. A complete list of the parts of speech we used is given in Appendix A.

Since each word can have more than one part-of-speech, it is possible for the proper POS tag for a word to be ambiguous. Using rules trained over previously tagged text, the Brill tagger attempt to choose the best tag for each word based on its context. First, the Brill tagger assigns every word its most likely tag in isolation. Unknown words are assumed to be nouns or proper nouns depending on capitalization. After the initial pass, learned lexical and contextual rules are used to find a more likely tag [4].

Using lexical rules, unknown words may also be tagged differently based on cues such as prefixes, suffixes, and adjacent word co-occurrences. In our system, we relied on 151 lexical rules derived from roughly 300,000 words of tagged text from the WSJ. As an example, the sample lexical rule

```
NN ing fhassuf 3 VBG x
```

tells the tagger that if there's a word that ends in `-ing`, and has initially been tagged as a noun (NN), then it's POS tag should be changed VBG to indicate a verb, used either as a gerund or in its present participle form.

To refine the POS tags based on contextual cues, we used 313 contextual rules derived from roughly 600,000 words from the WSJ. For instance, the contextual rule

```
NN VB PREVTAG TO
```

tells the tagger to change a tag from noun (NN) to verb (VB) if the previous tag was TO.

3.3.3 Other Implementation Issues

Due to the presence of multiple classes for each word, an additional complexity is introduced into our bigram backoff model. Since each word can have multiple parts of speech, we maintain a table of words and their possible POS tags. Then to implement the model for multiple classes, we take the simple approach of summing over all possible classes.

To create a class-based language model, we also need to provide estimates for $P(w|c)$, the probability of a word given its class. For each possible class c, we calculate $P(w|c)$ using:

$$\hat{P}(w \mid c) = \frac{c(wc)}{c(c)} \quad if \quad c(wc) > cutoff$$

To smooth the distribution, we interpolate $P(w|c)$ with a uniform PDF:

$$P(w \mid c) = cwt * \hat{P}(w \mid c) + (1 - cwt) * \frac{1}{n}$$

3.4 Perplexity Experiments

In this section, we describe the series of perplexity experiments we performed to test and evaluate our class based models. First we describe the domain and data that we used to develop our models. Then, we give some baseline perplexity results using a word trigram before discussing perplexity experiments using our class-based models.

3.4.1 Domain and Data

To develop the class based system, we used data from the Hub4 broadcast domain. Our baseline word trigram language models are trained using the Hub4 language modeling training set, supplemented with additional data from the North American Business News corpus. The language model training data from Hub4 (hub4_lm) consists of data of transcripts of various news shows such ABC News, CNN, and NPR covering the period from January 1992 to June 1996. There are approximately 150 million words in the training data, comprising about 1 gigabyte of uncompressed data [15].

3.4.2 Baseline perplexity results

To develop our language models, we performed perplexity experiments using the Hub4 evaluation sets from 1996 and 1997 (h4e96 and h4e97), text from news.com (news_com), and ZDTV (zdtv1), and the New York Times (nyt). A summary of the different corpuses used for these experiments is given in Table 3.1. Table 3.1 also gives the number of words and the out-of-vocabulary (OOV) rate for each test set. The ccs column gives the number of context cues found in each corpus. For each sentence, we use the context cues <s> and </s> to mark the beginning and end of sentence respectively.

	# of words	OOV words	OOV rate	ccs
h4e96	20736	197	0.95	834
h4e97	32848	202	0.61	
news_com	30000	857	2.86	2897
zdtv1	13929	732	5.26	0
nyt	19999	296	1.48	1665

Table 3.1. Summary of corpus data

Since different smoothing techniques can influence the performance of our language model, we did an initial perplexity comparison of the four different discounting methods supported by CMU's SLM Toolkit. A brief summary of the four discounting methods is given in Appendix B. According to studies by Chen, of the four methods, linear discounting has the worst performance while Good-Turing the best. Both Witten-Bell and absolute discounting performs very poorly on small training sets. However, on large training sets, as is the case here, they are extremely competitive [9]. So it is no surprise to see from Table 3.2, that Witten-Bell, absolute, and Good-Turing smoothing all give about the same perplexity, with Witten-Bell discounting being slightly better for four out of five of the test sets. For consistency in future experiments, Witten-Bell discounting will be used exclusively.

	Witten_Bell	linear	Good_Turing	absolute
h4e96	168.38	201.31	167.80	167.45
news_com	420.44	559.41	426.50	423.40
zdtv1	258.69	318.54	263.11	261.96
nyt	270.30	344.74	273.15	272.12

Table 3.2. Perplexities for word trigrams using different backoffs

3.4.3 Class and Word n-gram models

Table 3.3 shows some perplexity results for different word and class n-gram models. As expected, the performance of the class model is extremely bad as compared to the word trigram model. While the combination models of **ccw** and **wwc** give better perplexity performance that the pure class model, **ccc**, the perplexity is still no where near that of the word trigram model.

	ww	cc	cw	wc	www	ccc	ccw	wwc
h4e96	224.39	706.08	473.29	491.65	168.38	574.51	410.37	410.80
h4e97	251.03	713.88	573.39	561.52	179.56	659.58	488.21	460.96
news_com	474.47	934.21	814.24	869.94	420.44	881.27	712.05	765.73
zdtv1	310.72	642.24	560.40	514.12	258.69	603.76	502.10	449.41
nyt	345.22	946.89	715.44	785.03	270.30	869.23	599.61	640.11

Table 3.3. Bigram and trigram perplexities for class and word LMs

By combining the word and class n-gram models, we are able to improve the perplexity. Tables 3.4a and 3.4b give the bigram and trigram perplexities of combining a word and class n-gram model using linear interpolation with $\lambda = 1/2$. While the best

bigram perplexity results still came from the word bigram model, we were able to obtain comparable perplexities for a trigram model using an interpolated model. For the non-Hub4 corpuses (news_com, zdtv1, and nyt), the lowest perplexities are given by **ccw+www** or **wwc+www** model. For instance, the ccw+www model lowered perplexity from 420.44 to 356.00 for news_com, and from 270.30 to 261.39 for nyt.

	ww	cc	cc+ww	cw+ww	wc+ww
h4e96	224.39	706.08	246.20	241.62	245.39
h4e97	251.03	713.88	283.33	277.08	278.86
news_com	474.47	934.21	450.66	446.52	487.89
zdtv1	310.72	642.24	334.12	331.24	324.66
nyt	345.22	946.89	370.52	361.29	371.58

Table 3.4a. Bigram Perplexities for interpolation of word and class LMs

	www	ccc	ccc+www	ccw+www	wwc+www
h4e96	168.38	574.51	177.54	173.66	174.79
h4e97	179.56	659.58	194.67	189.31	188.85
news_com	420.44	881.27	364.27	356.00	390.74
zdtv1	258.69	603.76	261.07	259.07	251.42
nyt	270.30	869.23	270.73	261.39	264.89

Table 3.4b. Trigram Perplexities for interpolation of word and class LMs

From these results, we see the promise of combining word and class n-grams. However, we are more interested in investigating a backoff model that will give us more control over the combination of word and class n-grams than a simple interpolation model. Thus in the next section, we describe some experiments and results using a class-based bigram backoff model.

While the **ccw** and **wwc** models give a better estimate than a purely class based model of **ccc**, they also require significantly more parameters than the **ccc** model. Table 3.5 gives the number of parameters needed to create the different models.

	1-gram	2-gram	3-gram	1-gram marg.	2-gram marg.	Total
www	59,817	7,572,530	35,511,314			43,143,661
ccc	62	2,964	65,938			68,964
ccw	59,817	837,559	4,064,171	61	2,964	4,964,572
wwc	62	924,292	16,528,325	59,504	7,572,530	25,084,713

Table 3.5. Number of parameters used in each language model.

For a vocabulary of 59,817 words and 62 classes, the **ccc** model uses only ~70,000 parameters in comparison with the ~43 million required by the **www** model. Since they are essentially a mixture of class and word n-grams, the **ccw** and **wwc** model offer a

considerable reduction in complexity over the **www** model but are still significantly larger than the **ccc** model. In addition, they require us to keep track of separate counts for the marginals in addition to the normal n-gram probabilities. Because of the additional complexity introduced by the **ccw** and **wwc** models, we will focus on backoff to a purely class based model.

3.4.5 Bigram Backoff Model

In this section, we describe some experiments using the bigram backoff model. To develop the model and determine the best backoff path and cutoff parameters, we performed perplexity experiments on h4e96, news_com, zdtv1, and nyt using different backoff paths. The last data set, h4e97 was used as the test set to evaluate the bigram backoff model in recognition experiments which are discussed in Chapter IV.

The results for perplexity experiments using different backoffs on a bigram are given in Table 3.6. We experimented with backoff from word bigram to class bigram to class unigram (ww→cc→c), word bigram to class bigram to word unigram (ww→cc→w), and the full backoff model discussed in Section 3.2.2 (wc/cw) with a cutoff of 0, compared to normal word bigram backed off to word unigram (ww→w). Of the four models, ww→cc→c had the worst perplexity performance, due to the fact that it was the only model that did not backoff to a word unigram. Perplexity results for the other three models were comparable, with the full bigram backoff model (wc/cw) being slightly better than the other two.

	ww→w (wb)	ww→w	ww→cc(→c)	ww→cc→w	wc/cw (0)
h4e96	224.39	208.70	228.84	205.05	202.97
news_com	474.47	413.73	405.73	389.23	388.40
zdtv1	310.72	282.58	292.13	292.13	292.32
nyt	345.22	307.87	345.41	308.12	305.53

Table 3.6. Perplexities for different bigram backoff paths

Taking the full bigram backoff model, we also studied the effect of cutoffs on the perplexity. Varying the cutoff results in about the same perplexity, with the perplexity rising slightly as the cutoff is set too high. By examining the number of words in each backoff paths, we see that for low cutoffs, we mostly backoff to wc or cw. As the cutoff increases, we use the class bigram more and more in the backoffs. However, the

perplexity calculation is still dominated by the bigram estimate and the results are very similar.

	wc/cw (0)	wc/cw (10)	wc/cw (100)
h4e96	202.97	202.98	202.93
news_com	388.40	388.15	387.47
zdtv1	292.32	292.28	292.20
nyt	305.53	306.07	305.43

Table 3.7. Perplexities for bigram backoff models with cutoffs of 0, 10, 100

Finally, we compare four different trigram models: a pure word trigram (www) that backs off to word bigram, an interpolation of class and word trigram (ccc+www), a word trigram back off to class trigram (www→ccc), and a word trigram on top of bigram backoff model with cutoff of 0 (www→(0)). Again we find the worst performance in backing off to a class trigram only, and the best performance in the full backoff model.

	www	ccc+www	www→ccc	www→(0)
h4e96	168.38	177.54	180.21	132.14
news_com	420.44	364.27	384.11	282.34
zdtv1	258.69	261.07	241.07	204.90
nyt	270.30	270.73	295.56	198.81

Table 3.8. Perplexity results for different trigram models

While we achieve improvements in perplexity using classes, it is uncertain whether our class-based language model will improve recognition accuracy. In the next chapter, we will focus on the class-based bigram backoff model and study its impact on recognition experiments.

Chapter IV Recognition Experiments

4.1 Introduction

Unlike most work done to incorporate class based models into existing recognition systems by running N-best rescoring on lattices generated from word trigrams, we attempt to incorporate classes directly into the search. Instead of studying classes as a refinement, we try to investigate their performance directly through a variety of perplexity and word recognition experiments.

Although perplexity is often used to measure the quality of a language model, ultimately the most effective measure is the impact that a language model has on the accuracy of a recognizer. However, integration with a recognizer can often be complicated, since we have to be concerned not only with the language model but also performance issues. In addition, the word error rate is very domain and system dependent, and can be influenced by how language model interacts with acoustic model.

Thus in this section, we first give a brief overview of our baseline system and the integration of class based language models into the system. In sections 4.2 and 4.3, we describe in more detail the key points of implementation: the integration of classes into the search and the creation of the online tagger. Finally, in section 4.4, we present some recognition results on Hub4.

4.2 Baseline System

To evaluate our language models through recognition experiments, we integrated the models with the Calista speech recognition system developed at Compaq Computer Corporation's Cambridge Research Lab (CRL). Originally derived from the Sphinx-3 speech recognition system at CMU [33], Calista is a hidden Markov model (HMM) based speech recognizer. The acoustic models for our baseline system uses 6000 senonically-tied states, each consisting of 16 diagonal-covariance Gaussian densities. The phonemes are modeled with 49 context-independent phones, including noise phones such as AH and UM, and 87,057 context-dependent phonemes. For our vocabulary, we use the most frequent 59,817 words from the Broadcast News corpus. Normally, Calista uses a word trigram for its language model. For this thesis, we concentrated on the class-based language models described in Section 3.2.

Calista is essentially a two-pass decoding system that consists of two recognition stages. In the first stage, the Viterbi decoding using beam search produces a word lattice for each subsegment and a best hypothesis transcription. The best path search then goes over the word graph and determines the final optimal path. To keep Calista as a two-pass system, we integrated the class based language model directly into the search. In order to accomplish this, we created an online tagger (see Section 4.3) for use in the recognizer. The decoding search and the integration of our class based language model into the search is further discussed in Section 4.4.

4.3 Online Tagger

In this section, we describe an online tagger for providing the class history during recognition experiments. Because of the extensive search space, we did not wish to keep track of class history in addition to the word history. To avoid the extra memory usage, we created an online tagger to give the corresponding POS tags for the last two words. The design and implementation of the online tagger for the speech recognizer is described

in Section 4.3.1. Then, in section 4.3.2 we present some experiments comparing the performance of our online tagger against that of the Brill tagger.

4.3.1 Tagger Design

In order to incorporate classes into the recognition system, we created a deterministic online tagger using a simple tag table of words and their corresponding tags. Using the tag table, we can do online tagging of any given history (i.e. the last two words) without having to keep the tags in memory. Due to the extensive search space, the reduction in memory can be essential. In addition, the tagger has to be simple and fast for it to be included into the decoding search. Thus, a tag table with a straight forward and simple lookup is ideal.

To create the online tagger, the training data (consisting of 150 million words from Hub4) was first tagged using the Brill Tagger. A tag table was then created using probabilities gathered from the tagged text. The tag table consists of three subtables with the following maximum likelihood tags:

ww	$C_1,C_2 \mid W_1,W_2$	=	arg max $P(c_1,c_2 \mid w_1,w_2)$	if $c(w_1w_2) > m_2$
cw	$C_2 \mid C_1,W_2$	=	arg max $P(c_2 \mid c_1,w_2)$	if $c(c_1w_2) > m_1$
w	$C \mid W$	=	arg max $P(c \mid w)$	if $c(w) > m_0$

where m_0, m_1, and m_2 are the cutoffs counts for inclusion in the different tag tables.

The three tables combines to give $C_1,C_2 \mid W_1,W_2$ based on the following strategy:

Use $C_1,C_2 \mid W_1,W_2$	if w_1,w_2 in **ww** table
take $C_1 \mid W_1$ from **w** table and $C_2 \mid C_1,W_2$	else if c_1,w_2 in **cw** table
take $C_1 \mid W_1$ and $C_2 \mid W_2$ both from **w** table	otherwise

To save space, the entry $C_2 \mid C_1,W_2$ is stored only if it gives a different C_2 than the entry $C_2 \mid W_2$. Likewise, the entry $C_1,C_2 \mid W_1,W_2$ is stored only if the resulting tags are different from those given by $C_1 \mid W_1$ and $C_2 \mid C_1,W_2$.

Although the tags generated by the Brill Tagger are neither completely accurate nor unique, the use of statistics should ensure that the tag table is accurate most of the time. For instance, using this strategy, unknown words are automatically tagged as names

since that is what they are most likely to be. However, because POS tags can often be ambiguous and may change depending on context, our tagger, which only uses two words to determine the appropriate tags, is limited in its power and accuracy in comparison with the Brill tagger. But being much simpler, it can be incorporated within the decoder without any problems.

4.3.2 Tagger Experiments

The performance of the online tagger was measured by comparing text tagged using the online tagger against the results of the Brill tagger. For our experiments, we tested the performance of the online tagger on a portion of the training data from hub4_lm and also on some separate test data from news_com. As expected, the online tagger performed better on the training data from hub4_lm with accuracy of around 96%. On the test data from news_com, it had a comparable accuracy of about 93%.

	cutoff = 1	cutoff = 10	cutoff = 100	no cw, cw tags
sample of hub4_lm	95.958	95.909	95.566	92.210
news_com	92.885	92.870	92.291	90.271

Table 4.1. Accuracy of tagger for different cutoffs

	cutoff = 1	cutoff = 10	cutoff = 100	no cw, cw tags
w	59,503	59,503	59,503	59,503
cw	52,836	20,783	5,674	0
ww	452,496	46,639	6,509	0
total	564,836	154,462	76,328	59,504

Table 4.2. Complexity of tagger for different cutoffs

We further tested the effect of cutoffs (for cw and ww) on the performance of the online tagger (as compared to Brill). While increased cutoff gave decreasing accuracy as compared to Brill tagger, the total size of the tag table was reduced considerably. Since by reducing the size of the tag tables, we can not only minimize the amount of memory needed by the tagger but also reduce the lookup time, we decided to use a tag table with a cutoff of 100. Although a cutoff of 100 gave less accurate tags than cutoffs of 1 or 10, the total size of the tag tables was reduced by 7.4 times with the elimination of almost 500,000 cw and ww entries. In comparison, eliminating all cw and ww tags gave a much less accurate tagger while reducing the overall size of the tag table by only 22% more.

To ensure that our choice of a cutoff of 100 for the online tagger will not have a significant impact on perplexity, we studied the affect of three different cutoffs on the perplexity of a class trigram model. From Table 4.3, we can see that the perplexity is barely influenced by the cutoff parameter. However, having no ww or cw tag tables at all can adversely affect the perplexity.

	cutoff = 1	cutoff = 100	no ww, cw tags
h4e96 PP	575.14	574.51	583.58
h4e96 H	9.17	9.17	9.19

Table 4.3. Affect of cutoffs for online tagger on perplexity

4.4 Integration into Search

4.4.1 Overview of Search

The hypothesis search is an important component of any speech recognizer. Given an acoustic signal A, speech recognition can be regarded as the problem of finding the sequence of words, W, that maximizes $P(W \mid A)$. However, since the boundaries between possible words are unclear and each word can have multiple pronunciation, we also have to consider all possible ways that each sentence can be spoken and aligned with a recorded utterance. So the goal of the search phase is to find best sentence with the highest scoring alignment.

This can be accomplished by using the Viterbi decoding algorithm to generate a word lattice and find the best path through the graph. Based on dynamic programming, the Viterbi search is an efficient, time synchronous search algorithm. Starting with a set of initial nodes, the algorithm moves forward in time. At each time boundary, the algorithm maintains the best path from initial nodes to all nodes corresponding to the boundary. For each node at the boundary, the algorithm looks at all incoming arcs to that node and stores the arc that gives the maximum score as the back pointer.

For a trigram backoff model, the language score for each arc is given by:

$$P(w_i \mid h_i) = \begin{cases} \hat{P}(w_i \mid w_{i-1}, w_{i-2}) & \textit{if trigram estimate exists} \\ q_1(w_{i-1}w_{i-2})\hat{P}(w_i \mid w_{i-1}) & \textit{if bigram estimate exists} \\ q_1(w_{i-1}w_{i-2})q_2(w_{i-1})\hat{P}(w_i) & \textit{otherwise} \end{cases}$$

where $\hat{P}(w_i \mid w_{i-1}, w_{i-2})$, $\hat{P}(w_i \mid w_{i-1})$, and $\hat{P}(w_i)$ are the trigram, bigram, and unigram estimates, and q_1,q_2 are the bigram and unigram backoff scores. To make the search efficient, the unigram backoff scores are calculated only once at each boundary. This is possible since the unigram backoff score, $q_2(w_{i-1})$, is not dependent upon the next word w_i. Because the Viterbi search only store the maximum path to each node, we only need to maintain the largest unigram backoff at each boundary in order to determine the transition scores to all possible next words. One side effect is that unigram backoffs are added to the lattice regardless of whether the bigram or trigram score exists or not.

When the search is complete, the best path can be reconstructed by taking the best scoring final node and following the back-pointers back through the graph. Because of the large size of the vocabulary, the search space would be gigantic if all possible paths at each boundary were kept. To restrict the graph to a more reasonable size, a beam search is used to threshold the search by trimming paths with very low probability.

4.4.2 Class-based Backoff Nodes

While it is a simple matter to implement the bigram backoff model for perplexity experiment, a straight implementation of bigram backoff model into the search phase proves to be extremely inefficient. For one, the presence of multiple tags for each word presented a computational challenge. To accurately represent $P(w_i \mid h_i)$, we have to sum over all possible classes that the word can take. However, within the decoder, doing this proves to be extremely inefficient and unwieldy. In order to reduce computational costs, we decided to use the maximal likelihood tag only.

Even then, because of the huge search space, it is computationally infeasible if we attempt to calculate the backoff score for each possible trigram. To resolve this, we rely on the time synchronous property of the Viterbi search and use a similar method to calculate class backoffs as was done for unigrams.

In the unigram case, the highest scoring unigram backoff is stored. For the $P(c_2 \mid w_1)$ and $P(c_2 \mid c_1)$ component of the backoff, we keep track of the highest scoring backoff for each class. The $P(w_2 \mid c_1)$ component of the backoff was incorporated into the bigram scores directly so that if there are no bigram for $P(w_2 \mid w_1)$ but $c(w_1)$ is still greater than the cutoff, $P(w_2 \mid c_1)$ would be used as the bigram score. Thus for a word w_1, there are transitions to w_2 using the following bigram and trigram scores:

$$P(w_2 \mid h) \approx \begin{cases} P(w_2 \mid w_1, w_0) & \textit{if trigram exists} \\ P(w_2 \mid w_1) & \textit{if bigram exists} \\ P(w_2 \mid c_1) & \textit{if } c(w_1) > m \end{cases}$$

Backoff scores from unigrams and class backoffs are incorporated by adding transitions to w2 as follows:

For a unigram: $P(w_2 \mid h) = P_{ug} P(w_2)$ where P_{ug} is the largest unigram backoff exiting from w_1.

For classes: $P(w2 \mid h) = \max_{c_2} P(w_2 \mid c_2) P_c(w_1 \mid c_2)$ where $P_c(w_1 \mid c_2)$ is the largest class backoff for c_2 exiting from w_1 and for each class c_2, $P_c(w_1 \mid c_2)$ is calculated by

$$P_c(w_1 \mid c_2) = \begin{cases} P(c_2 \mid w_1) & \textit{if } c(w_1) > m \\ p(c_2 \mid c_1) & \textit{otherwise} \end{cases}$$

Using this scheme, we will essentially always take into account class and unigram scores regardless of whether we have bigram or trigram scores for w_2. Only when the bigram score is larger than the class and unigram backoffs for the bigram will the bigram estimate be used. Thus, we should be able to remove all bigrams whose bigram score is less than its backoff score without affecting the recognition. Recognition results for such a reduced bigram is given in Section 4.5.3.

4.5 Recognition Results

To evaluate our final system, we performed a series of word recognition experiments on h4e97. In addition to experimenting with our class-based bigram backoff

model, we also investigate two other models. In section 4.5.2, we study influence of backoffs on WER in general by backing off to constants instead of unigrams. In section 4.5.3, we examine whether we can reduce the size of our language model by removing unused bigrams.

4.5.1 Class Bigram Backoff Experiments

	bo_ug	bo_cc
ww	251.03	244.64
www	179.64	151.22

Table 4.4. Perplexity results for h4e97

Before looking at the recognition performance of our bigram backoff model on h4e97, we present the perplexities of this corpus. As can be expected from our previous perplexity experiments, our backoff model (bo_cc) gave lower perplexity than the normal backoff (bo_ug).

To investigate the performance of the bigram backoff model, we performed recognition experiments on a word bigram backed off to classes using several different backoff paths. For the full bigram backoff model, we also experimented with several different language weights (lw) and word insertion penalties (ip) to improve the WER. The recognition results for clean, studio conditions (F0) and all acoustic environments (ALL) are summarized below.

Bigram recognition experiments on hub4 97:

	F0	ALL	
ww→w	19.6	29.9	(baseline word bg with backoff to word ug)
ww→cc→w	20.3	30.7	(backoff to classes without using backoff weights)
ww→wc/cw→cc→w (0)	20.6	30.8	(+ bowt, cc→ug)
ww→wc/cw→cc/w (0)	19.2	29.9	(+ bowt, cc+ug)
(lw=9.5, ip=0.45)	20.1	30.9	
(lw=9.7, ip=0.6)	19.3	29.7	
(lw=9.7, ip=0.45)	19.3	29.7	

Trigram recognition experiments on hub4 97:

	F0	ALL
www→ ww→w	17.2	27.4
www→ ww→ wc/cw→cc/w	17.0	27.4

Unfortunately, the improvement in recognition accuracy is not as significant as the perplexity results might lead us to expect. With the full backoff model, we can get a 0.4%

improvement in WER over a normal bigram on F0 conditions, while maintaining the same WER for all conditions. For the trigram model, we were again able to improve the WER slightly on F0 conditions, with the same WER for all conditions.

Because the optimal combination of acoustic and language models might be different for our class backoff model, we investigate whether a different language model weight will give better performance. In addition, we noticed that long words tended to become chopped up into phrases of smaller words when using classes which might be caused by a low word insertion penalty. To find the best parameters using the class based backoff model, we varied the language model weight (lw) and word insertion penalty (ip) over a subset of 100 utterance from the development set of Hub4 1997. By tweaking the two parameters, we were able to improve the performance over all conditions by 0.2%. Overall, the bigram experiments on h4e97 suggests that we can only get slight improvements in the recognition accuracy.

4.5.2 Unigram Backoff Experiment

The difference in perplexity and recognition results from the bigram backoff model suggests that although different backoff schemes can have a tremendous effect on perplexity, they only influence the word error rate minimally. To investigate this matter, we performed word recognition experiments on h4e97 by backing off to constants instead of word unigrams. We found that backing off to a reasonable probability of $1/n$, where n is 59,817 or the number of unigrams, gave only a 1% worse WER. Meanwhile, backing off to a higher probability of 1/5000 gave a much worse WER.

	F0	ALL
backoff to ug	19.6	29.9
1/n_ug	20.4	30.9
1/5000	24.4	34.5

Table 4.5. WER results for constant backoffs on h4e97

These results suggest although the method of backoff has an impact on the recognition accuracy, it is not that significant as long as the backoff probabilities are not absurd. However, it is also important to note that recognition rates are very system dependent and this might not be the case for other systems.

4.5.3 Bigram Reduction Experiment

Because of the nature of the implementation of the search, we suspect that we can reduce the size of our language model without affecting the WER very much. To make the search efficient, calculation of backoff scores are combined so that backoff probabilities are added to the lattice regardless of the possible presence of bigrams. Since the Viterbi search only keeps the best path, bigrams for which the bigram probability is less than their backoff probability are ignored. Therefore, we should be able to remove these bigrams and reduce the size of our language model without affecting the WER.

	F0	ALL
normal bigram	19.6	29.9
reduced bigram	19.7	30.3

Table 4.6. WER of reduced bigram on h4e97.

Table 4.6 shows the impact of eliminating these bigrams on the WER. While we were able to reduce the number of bigrams by 5.5% from 7,572,530 to 7,153,454, the WER increased slightly. For the F0 condition, the WER increased by only 0.1 %, while the overall WER increased by 0.4 %.

Chapter V Portability

5.1 Introduction

In this section, we describe a series of experiments to explore how well do word and class n-grams port to domains outside of Hub4. We will study the robustness of the language model to a new domain by measuring the perplexity and word error rates on data from ZDTV and Talkradio in Section 5.2. Next we examine the effect of out-of-vocabulary (OOV) words on the performance of our language models in Section 5.3. Finally, in section 5.4 we discuss a feature based language model and its performance.

5.2 Portability Experiments

5.2.1 Motivation

While many studies have currently focused on particular tasks (e.g. Hub4), it is also important to see how well language models generalize to other domains without retraining to other corpuses. In particular, we are interested in the application of speech recognition for creating transcripts for radio shows. Often, there are no transcriptions, and thus no training data, available for these shows. In contrast, the Hub4 domain is ideal for development purposes because of the large amount of transcribed data available. Thus, we want to be able to use a language model trained on data from the Hub4 domain and port it to other domains. To investigate the portability of our language model, we describe several perplexity and word recognition experiments conducted on TalkRadio and ZDTV.

5.2.2 Data Description

For these portability experiments, we concentrate on the performance of the bigram backoff model on ZDTV and four popular radio shows, which we will refer to collectively as Talkradio. Each of the Talkradio shows, consists of transcripts for four segment in Real Audio gather from the Internet. For our recognition experiments, we will be focusing mainly on Dr. Laura, Artbell, and ZDTV.

Data	Show	Content
artbell	*Art Bell Coast to Coast*	4 segments from the popular radio show exploring paranormal phenomena
drlaura	*Dr. Laura Show*	4 segments from radio talk show with advice and solutions on family, children, religion, etc. from Dr. Laura
emerson	*Rick Emerson Show*	4 segments from popular radio talk show with Rick Emerson
edtyll	*Ed Tyll Show*	4 segments from talk show focusing on topics for young adults
ZDTV	*ZDTV News*	News about about computing, technology, and the Internet.

Table 5.1. Summary of shows and their content

5.2.3 Perplexity and Recognition results

Perplexity Results

From Table 5.2, we see that the class bigram backoff model gives promising perplexity results. As expected, a pure class model (ccc) had extremely bad perplexity while the class bigram backoff model (www.bo_cc) improved perplexity. For instance, the class trigram increased the perplexity to 785.97 from a word trigram perplexity of 292.71 on the first segment from Artbell (ab050299-10-25). In contrast, the class bigram backoff model (www.bo_cc) reduced the perplexity by 22.7% to 226.15. On average, there was about an 18% decrease in perplexity by using the backoff model across the domains.

	www	www.bo_cc	ccc
ab050299-10-25	292.71	226.15	785.97
ab051799-100-115	257.63	237.93	797.45
ab060199-185-200	278.14	217.94	630.93
ab061199-205-220	210.62	168.48	534.57
drlaura040299-5-20	232.22	188.98	597.77
drlaura062299-150-165	251.37	201.51	626.11
drlaura073099-45-60	184.40	155.02	526.92
drlaura082799-105-120	176.03	148.48	481.15
edtyll062299-10-25	278.69	217.74	695.99
edtyll072299-50-65	267.83	210.73	705.08
edtyll080699-120-135	181.44	149.90	428.42
edtyll082699-140-155	189.79	157.88	510.71
emerson0607-10-25	255.18	199.45	655.89
emerson0705-65-80	271.36	212.46	659.90
emerson0802-150-165	335.72	261.09	688.63
emerson0825-200-215	241.19	258.93	726.18
h4e96	168.38	132.14	574.51
h4e97	179.56	151.22	659.58
news_com	420.44	282.34	881.27
zdtv1	258.69	204.90	603.76
nyt	270.30	198.81	869.23

Table 5.2. Perplexities for Talkradio and ZDTV

Recognition Results

To study the performance of the bigram backoff model on word error rate, we performed recognition experiments on Dr. Laura, Artbell, and ZDTV. Because the overall word error rates for TalkRadio (50%) and ZDTV (35%) are extremely high in comparison to that of Hub4 (27%), we hoped that class-based language model can decrease the WER for TalkRadio and ZDTV. Yet despite the promising perplexity experiments, the recognition results were not quite as good. While we achieved slight improvements in recognition accuracy for Dr. Laura (0.11 %) and Artbell (0.26 %), the accuracy for ZDTV deteriorated slightly.

	Dr. Laura		Artbell		ZDTV
	cutoff = 0	Cutoff = 1	cutoff = 0	cutoff = 1	
www	51.75	51.35	51.84	51.98	34.74
www.bo_cc	51.64	51.42	51.58	51.97	35.67

Table 5.3. WER for Dr. Laura, Artbell, and ZDTV

5.2.4 Analysis

One possible reason for the disappointing results using the word bigram model is the particular selection of classes. As seen in past work, POS tags are not a particularly effective way to classify words because they can only provide weak syntactic constraints. This is especially a problem since the spoken language is often not grammatically correct. POS tags are also limited in the number of classes and the specificity. Our class based model was constructed using only 62 classes while most class based LM's in past work have several hundred classes. In addition, it is not clear if all 62 classes are useful. For example, the two verb forms VBN and VBD are so similar in their usage as to be often indistinguishable.

Another problem is the presence of multiple tags for each word. While it is natural for a word to belong to several different classes, it introduces complexity and differences into the models used for perplexity and recognition experiments. Although our perplexity experiments were performed by summing over all possible classes for each word, computational issues limited recognition experiments to calculating $P(w)$ using the mostly likely class only, resulting in a less accurate $P(w)$.

Since the accuracy of our estimates of $P(w|c)$ is also uncertain, we experiment with changing the class weight in the interpolation of $P(w|c)$ with a uniform distribution. Even though, lowering the class weight tend to improve the WER, the improvement again is not very significant.

	www	cwt=0.1	cwt=0.2	cwt=0.4
ab050299-10-25	50.73	50.47	50.51	50.69
ab051799-100-115	48.31	48.31	48.27	48.22
ab060199-185-200	53.47	53.47	53.47	54.54
ab061199-205-220	44.97	45.01	45.05	45.43

Table 5.4. WER for Artbell using different class weights.

These problems suggest that perhaps a better classification using semantic information would have given better results. This idea is pursued further in our feature-based model, which considers the separation of root (semantics) and stem (syntax), described in section 5.4. But first in Section 5.3, we investigate the contribution of out-of-vocabulary (OOV) words to the word error rate. From our experiments on the influence of OOV, we will also realize the importance of pronunciation models.

5.3 OOV Experiments

In this section, we describe a series of experiments done to study the affect of out-of-vocabulary (OOV) words on the word error rate. We are mainly interested in how easy it is to adapt these language models to new corpora by introducing new words into existing classes.

5.3.1 Motivation

A major problem with porting language models to new domains is the existence of new words that are not part of our original dictionary. These out-of-vocabulary (OOV) words are impossible for the recognizer to decode and cannot be introduced into a language model without retraining. Even then, they pose a problem since we often have no training data for these new words. The ability to introduce new words into a language is extremely important not only because of portability issues but also because of the temporal nature of broadcast news and radio shows. While something might be a hot topic on one day, it is an entirely different story the next month. Indeed, OOV words tend to be names or other important keywords, words that a user might want to be able to do search and retrieval on. Thus, having a flexible language model, into which new words can easily be added, is doubly important.

For instance, consider using a language model trained on Hub4 data on a ZDTV broadcast. Although we would like the decoder to recognize keywords such as "ZDTV" with the same frequency as the name of a corresponding news source found in Hub4, say "CNN", the language model is rarely that robust. In general, the language model trained on the Hub4 domain will be able to recognize the word "CNN" with no problem. However, it will fail to recognize "ZDTV" even if it was added to vocabulary since there's simply no data for it. But since "ZDTV" and "CNN" are extremely similar in nature, we would expect them to have the same distribution, suggesting that a class-based model could be the ideal solution.

With a class-based model, there is no need to retrain the entire language model, which could not only be extremely time consuming, but also unproductive due to the lack of data. Instead, we only need to add the word into the vocabulary and give it the appropriate $P(w|c)$. Thus, a class-based model offers a convenient way of introducing new words.

5.3.2 Method

While we can introduce new words by adding them directly into the vocabulary and retraining the language model, it can be extremely time consuming. In addition, we might also lack the necessary data for the new words so to obtain a good estimate of $P(w)$ for these new words, we will need to find new training data that uses these words. As an alternative, we can introduce new words into a class-based language model by simply providing possible tags for word and $P(w|c)$. Since it is uncertain what the correct $P(w|c)$ should be, we use a uniform distribution of $\frac{1}{n_c}$, where n_c is the number of classes, to estimate $P(w|c)$. Similarly, we use $\frac{1}{n}$, where n is the number of unigrams, to estimate the unigram probability of $P(w)$.

The procedure for introducing new words into the language model is summarized below:

1. Create list of OOV words
2. Make dictionary with OOV words
3. Add to unigrams with uniform probability:

   $P(w) = \frac{1}{n}$ where n is the number of unigrams.

4. For class based language model:
   a. Tag OOV words
   b. Add tag tables and to $P(w|c)$ using uniform probability:

      $P(w \mid c) = \frac{1}{n_c}$ where n_c is the number of words in class c.

5.3.3 Results and Analysis

To investigate the effect of out-of-vocabulary words on recognition accuracy, we attempt to decrease the WER by adding new words into the language model. We chose to do OOV experiments on Artbell, which has a relatively low OOV rate, and ZDTV, which has the highest OOV rate of the five shows. The out-of-vocabulary rates for ZDTV and the four Talkradio shows is given in Table 5.5.

Show	# words	# OOV	OOV rate
artbell	9319	76	0.82 %
drlaura	8512	72	0.85 %
emerson	12545	171	1.36 %
edtyll	10697	104	0.97 %
zdtv	34486	555	1.61 %

Table 5.5. Out of vocabulary rates for Talkradio and ZDTV.

By adding out-of-vocabulary words to Artbell, we were only able to decrease the WER by .09%. However, the inclusion of OOV rates to ZDTV (which has more OOV words), decreased the WER by almost 2%. Again, the bigram backoff model improved the WER for Artbell while it increased the WER for ZDTV.

	Artbell		ZDTV	
	no OOV	+OOV	no OOV	+OOV
www	51.84	51.75	34.74	32.91
www.bo_cc	51.58	51.49	35.67	33.63

Table 5.6. WER for LM with OOV words added on Talkradio and ZDTV.

When introducing new words, pronunciation models can be a problem. For instance, by changing the pronunciation of a single word, A.O.L, in the dictionary for ZDTV, we are able to lower the WER by 1%: from 32.91 to 31.88 for the word trigram model, and from 33.63 to 32.63 for the word trigram backed off the to bigram backoff model. By tweaking the pronunciations for the new words for Artbell, we were also to improve the WER for Artbell from 51.49 to 51.22.

We also experimented with introducing the OOV words for ZDTV by retraining the entire language model. Table 5.7 gives the results of the experiment, using the new dictionary with the adjusted pronunciations.

	no OOV	+OOV (ug)	+OOV (retrain LM)
www	34.74	31.88	32.76
www.bo_cc	35.67	32.62	33.40

Table 5.7. WER for LM with OOV words added on ZDTV.

By retraining the language model, we get worse performance than simply adding the new words into the unigrams with a uniform probability of $1/n\_ug$ where n_ug is the total number of unigrams. This result can be explained by the fact that these are OOV words for which we had little if any data, thus they would be vastly underestimated by the language model.

5.3.4 Future Work

Our results suggest that out-of-vocabulary words might be more important than the OOV rate indicates. Even when the OOV rate is low, adding the new words into the vocabulary can help the decoder not only recognize the added words but also words spoken close to these OOV words. However, a problem with introducing new words is that until we are given a transcription, it is impossible to determine what the OOV words are. An extension of this work can concentrate on a dynamic language model that is able to flexibly add new words as needed. Due to the nature of English language, in which new words are constantly added, this ability is especially important. For the broadcast news domain, in an age of changing technology and changing news, new acronyms and new names are appearing all the time. Often, these words are keywords, the important words that listeners are most interested, and it is important for a language model to be able to adjust and incorporate these words into its vocabulary.

5.4 Feature Based Language Model

5.4.1 Motivation

Lastly, we wish to investigate a feature based language model. If every word is tagged using several different features such as POS, its tense or number, then by combining the probabilities for each feature, we can obtain a reasonable estimate for the probability of a word. Using a class n-gram approach, we can represent each feature by a different class and attempt to obtain the probabilities for each feature independently.

Assuming that we have m different features and that feature probabilities are independent, we will have:

$$P(w_i \mid h) = P(w \mid c_1, c_2, ...c_m)\, P(c_1 \mid \Phi_{c1}(h)) P(c_2 \mid \Phi_{c2}(h)) \cdots P(c_m \mid \Phi_{cm}(h))$$

Using such a feature-based model, each component, $P(c_i \mid \Phi_{ci}(h))$, can be estimated using a language model optimal for that feature. Then, we can effectively combine the information provided by the different classes.

A feature-based model should offer an advantage over traditional word n-grams for words with the sparse data problem, even for words that might not be part of the vocabulary. While a limited number of common words such as "the", "and", "is" occur many times and are well modeled, many words appear only once or twice in the training data. However, by breaking these words into their feature components, we might have the necessary class information to provide a better probability model for them. In addition, we can even hope to reconstruct some OOV words using their morphological properties.

5.4.2 Method

For our experiments, we concentrate on a simple feature based model that considers the root and stem separately. We take a different approach for stop and non-stop words. For our purposes, stop words will be regarded as all words that are not verbs, nouns, adjectives, or adverbs. Since stop words tend to be common words such as "the", "a", that are well modeled, they are predicted using the normal bigram distribution. For non-stop words, we attempt to predict the root based on the roots of the last two non-stop words. Since the last two non-stop words may not be the last two words, we hope to incorporate some longer-range constraints than normally found in a word n-gram. For our stem, we simply use the POS tag, which includes information such as the verb or noun form. The tags are predicted based on the tags of the last two words. The steps involved in creating the feature based model is summarized below:

1. Extract roots
2. Build table of roots|word,tags
3. Calculate $P(w_i)$ using:

$$P(w_i \mid h_i) = \begin{cases} P(w_i \mid w_{i-1}) & \text{if } w_i \text{ is a stop word} \\ P(t_i \mid t_{i-1}, t_{i-2}) P(r_i \mid r_{i-1}, r_{i-2}) & \text{otherwise} \end{cases}$$

where t_{i-1}, t_{i-2} are the tags of last two words and

r_{i-1}, r_{i-2} are the roots of last two non-stop words

To extract the root from a word, we use simple pattern matching rules on words and their POS tags. For instance, the rule

```
VBZ (.*)IES$ $1Y VERB
```

indicates that if a word ends in "IES" and is labeled as a third singular verb form (VBZ), then the root is a verb with the ending "IES" replaced with "Y". So for the word "flies", the root would be "fly". For verbs with irregular forms, such as "are" and "is", we maintain a list of the verbs and their roots. The list of irregular forms is taken from LDC's (Linguistic Data Consortium) COMLEX Syntax Corpus, a monolingual English Dictionary consisting of 38,000 head words intended for use in natural language processing.

5.4.3 Results and Analysis

We tested the described feature-based language on three data sets: artbell, h4e97, and ZDTV. Unfortunately, our perplexity results were much worse than that of the word trigram.

	www	feature
artbell	292.71	1152
h4e97	179.56	809
ZDTV	258.69	1073

Table 5.8. Perplexities for the feature based LM

One reason is because the probabilities for the two features $P(root_i \mid h_i)$ and $P(tag_i \mid h_i)$ are not independent, we cannot combine them using the proposed method. Another problem is that combining probabilities inevitably lead to smaller probabilities. Thus, we need a better way to combine the information from the different features.

5.4.4 Future Work

Despite our preliminary perplexity results, we believe that further research on an improved feature based model can contribute to the future of language modeling. One can imagine many extensions and improvements upon the described model. For instance, one can investigate different features. One can also build an improved feature based model by using more sophisticated methods to combine the features. In addition, because separating features will allow each component to be modeled using a different language model, determining the optimal model for a feature is also important. By using a feature-based model, many different types of constraints such as long distance and local constraints or syntactical and semantic constraints, can be separately captured using different models and then recombined to give new flexibility to language modeling.

Chapter VI Conclusions and Future Work

6.1 Summary

In this thesis, we explored a bigram backoff language model combining class and word n-grams using POS tags. We described the work done to integrate the language model into a two-pass decoding search for recognition experiments, including the development of an online tagger. As our main focus, we investigated the portability of a language model trained on data from Hub4 to other domains such as TalkRadio and ZDTV. Although the class-based LM lowered perplexity across the different shows, it did not always improve the WER. For Hub4 and TalkRadio, we were able to lower the WER slightly, but using the bigram backoff to classes increased WER for ZDTV slightly.

To analyze the effect of backoff on the WER, we performed unigram backoff experiments where we backed off to a constant instead of the unigram probability. Results indicate that the backoff scheme is not very important as long as we have reasonable backoff probabilities. We also investigated a bigram reduction backoff model unnecessary bigrams were removed. While we were able to achieve a reduced language model, the reduction in size was not significant enough to make the slight increase in WER worthwhile. Our experiments were also limited by using a two-pass system. Perhaps a word trigram is already optimal in capturing local constraints and we should only be using classes as a refinement on top of a word lattice generated using a word trigram.

As part of our portability experiments, we also studied the effect of OOV words on the WER. We discovered that OOV words were not as big a problem as we had

feared, with the shows from Talkradio and ZDTV having a relatively low OOV. Nevertheless, adding the unknown words into the vocabulary helped to lower the WER considerably. However, using a class-based model did not help us improve the WER more than a word trigram when new words are added. This could be due to the fact that POS tags were not specific enough to capture the relationship between words such as "ZDTV" and "CNN" across domains.

Finally, we investigated the separation of root and stem in a feature based language model. While the idea is promising, we were unable to achieve interesting results. More work is needed to fully explore the possibilities of such a model. In the next section, we describe some future work in this area.

6.2 Future Work

There are many possible areas in which the present work can be extended. For one, recognition accuracy is very domain and recognizer specific and we only used a relatively small amount of data to investigate portability issues. Thus, we will need to expand our work by performing experiments on other shows.

In addition, it would be interesting to examine the portability of other class-based language models. For example, we can improve our language model by pursuing more precise classification schemes. A more intriguing direction is to pursue an improved feature based model. By separating the different features of a word, such as stem and root, each component can be modeled using a technique most optimal for that feature. For instance, in our work, we attempted to predict the roots using short-range constraints while predicting the stem using longer-range information on previous stems. However, due to time limitations and challenges with combining the various features into a single model, we were unable to improve on a word n-gram. Yet we believe that using such a scheme, one can create a language model that is more robust and flexible to changes in domain and the addition of new words.

Finally, a language model using dynamic OOV learning should also be investigated. As mentioned in Section 5.3.4, OOV words will continue to be a problem because of the changing nature of both language and the domain. The ability to incorporate new words automatically into a language model is vital for reducing word error rates and the multimedia indexing and retrieval of keywords.

Appendix A: Summary of Classes

POS Tag	Description	Example
<s>	Context Cue: start of segment	<s>
</s>	Context Cue: end of segment	</s>
CC	coordinating conjunction	and
CD	cardinal number	1, third
DT	Determiner	the
EX	existential there	*there* is
FW	foreign word	d'hoevre
IN	preposition/subordinating conjunction	in, of, like
JJ	Adjective	green
JJR	adjective, comparative	greener
JJS	adjective, superlative	greenest
LS	list marker	1)
MD	modal	could, will
NN	noun, singular or mass	table
NNS	noun plural	tables
NN'S	noun, possessive	table's
NNP	proper noun, singular	John
NNP'S	proper noun, possessive	John's
NNPS	proper noun, plural	Vikings
PDT	predeterminer	*both* the boys
POS	possessive ending	friend*'s*
PRN'D	pronoun	I'd, he'd
PRP	personal pronoun	I, he, it
PRP$	possessive pronoun	my, his
RB	adverb	however, usually, naturally, here, good
RBR	adverb, comparative	better
RBS	adverb, superlative	best
RP	particle	give *up*
SYM	punctuation or symbol	
TO	to	*to* go, *to* him
UH	interjection	uh-huh
VB	verb, base form	take
VBD	verb, past tense	took
VBG	verb, gerund/present participle	taking
VBN	verb, past participle	taken
VBP	verb, singular present, non-3rd	take
VBZ	verb, 3rd person singular present	takes
WDT	wh-determiner	which
WP	wh-pronoun	who, what
WP$	possessive wh-pronoun	whose
WP'D	wh-pronoun with 'd	who'd
WP'LL	wh-pronoun with 'll	who'll
WP'S	wh-pronoun with 's	who's
WP'VE	wh-pronoun with 've	
WRB	wh-abverb	where, when

Additional Classes

HE'LL	I'LL	IT'S	THEY'LL	WE'LL	YOU'LL
HE'S	I'M	LET'S	THEY'RE	WE'RE	YOU'RE
	I'VE	THAT'S	THEY'VE	WE'VE	YOU'VE

Appendix B: Summary of Discounting Methods

Using CMU's SLM toolkit, each N-gram is smoothed using one of four different discounting methods before they are combined using backoff. In discounting, the actual count r for an event is replaced by a modified count r^* to redistribute probability mass from common events to unseen events. Here we give a summary of the four discounting methods, see [9] and [10] for more details.

Witten Bell discounting
Originally developed for text compression, Witten-Bell discounting adapts well to the language modeling task for large amounts of data. If t is the number of types (i.e. number of different words) following a particular history h, and n is the total number of words, the Witten-Bell estimations for the probability of a word w following h are given by:

$$P(w \mid h) = \frac{c}{n+t} \qquad \text{if } w \text{ had appeared after } h \text{ in training data}$$

$$P(w \mid h) = \frac{t}{n+t} \qquad \text{for unseen events}$$

Basically, we want $P(w|h)$ to be proportional to how often the word w occurs after the history h. For unseen events, if there are more unique words that follow a particular history h, then we are more likely to have h be followed by other words.

Good Turing discounting (essentially Katz smoothing)
In Good Turing discounting, the modified count r^* is chosen so that r^*/C, where C is the total number of N-grams, is the expected probability of an N-gram given that it occurred r times in the training set.

$$r^* = (r+1)\frac{n(r+1)}{n(r)} \qquad \text{where } n(r) \text{ is the number of n-grams that occurs r times}$$

Since we can assume the original count r to be accurate for extremely common events, this estimation is used only for events that occur fewer than K times. In addition, because the estimate generates ridiculous values if $n(r) = 0$, it is necessary to adjust $n(r)$ so that they are all larger than zero.

Absolute discounting
Absolute discounting takes a fixed discount from each count and redistributes the leftover among unseen event. In absolute discounting, we have

$$r^* = r - b \qquad \text{where b is a constant, typically } b = \frac{n(1)}{n(1)+2n(2)}$$

Linear discounting
Linear discounting takes the modified count to be linearly proportional to actual count, giving us:

$$r^* = \alpha r \qquad \text{where } \alpha \text{ is a constant between 0 and 1.}$$

Typically, $\alpha = 1 - \frac{n(1)}{C}$ where C is the total number of N-grams

References

[1] G. Adda, M. Jardino, and J. L. Gauvain. "Language modeling for broadcast news transcription," *Proceedings of the European Conference on Speech Communication and Technology*, pp. 1759-1762, Budapest, 1999.

[2] A. Ando, A. Kobayashi, and T. Imai. "A thesaurus-based statistical language model for broadcast news transcription," *Proceedings of the International Conference on Spoken Language Processing*, pp. 2383-2386, Sydney, 1998.

[3] J.R. Bellegarda, J.W. Butzberger, Y.L. Chow, N.B. Coccar, and D. Naik. "A novel word clustering algorithm based on latent semantic analysis," *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 172-175, Atlanta, 1996.

[4] E. Brill. "A simple rule-based part of speech tagger," *Proceedings of the Third Conference on Applied Natural Language Processing*, pp. 152-155, Trento, Italy, 1992.

[5] P. Brown, V. Della Pietra, P. deSouza, J. Lai, and R. Mercer. "Class-based n-gram models of natural language," *Computational Linguistics*, vol. 18, no. 4, 467-479, December 1992.

[6] H. Cerf-Danon, and M. El-Bèze. "Three different probabilistic language models: comparison and combination," *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 297-300, Toronto, 1991.

[7] C. Chelba, and F. Jelinek. "Recognition performance of a structured language model," *Proceedings of the European Conference on Speech Communication and Technology*, pp. 1567-1570, Budapest, 1999.

[8] S. Chen. *Building Probabilistic Models for Natural Language*, PhD thesis, Division of Applied Sciences, Harvard University, May 1996.

[9] S. Chen, and J. Goodman. "An empirical study of smoothing techniques for language modeling", Technical Report TR-10-98, Computer Science Group, Harvard University, 1998.

[10] P.R. Clarkson and R. Rosenfeld. "Statistical language modeling using the CMU-Cambridge Toolkit," *Proceedings of the European Conference on Speech Communication and Technology*, pp. 2702-2710, Rhodes, 1997.

[11] S. Della Pietra, V. Della Pietra, J. Gillett, J. Lafferty, H. Printz, and L. Ureš. "Inference and estimation of a long-range trigram model," *Grammatical Inference and Applications*, Lecture notes in Artificial Intelligence, vol. 862, R.C. Carrasco and J. Oncina, eds., pp. 78-82, Springer Verlag, Berlin 1994.

[12] A. Farhat, J.F. Isabelle, D. O'Shaughnessy. "Clustering words for statistical language models based on contextual word similarity," *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 180-183, Atlanta, 1996.

[13] P. Geutner. "Introducing linguistic constraints into statistical language modeling," *Proceedings of the International Conference on Spoken Language Processing*, pp. 402-404, Philadelphia, 1996.

[14] P. Geutner. "Fuzzy class rescoring: a part-of-speech language model." *Proceedings of the European Conference on Speech Communication and Technology*, pp. 2743-2746, Rhodes, 1997.

[15] D. Graff. "The 1996 broadcast news speech and language-model corpus," *Proceedings of the DARPA Speech Recognition Workshop*, pp. 11-14, Virginia, 1997.

[16] P.A. Heeman, and J.F. Allen. "Incorporating POS Tagging into language modeling," *Proceedings of the European Conference on Speech Communication and Technology*, pp. 2767-2770, Rhodes, 1997.

[17] P.A. Heeman. "POS tagging versus classes in language modeling," *Proceedings of the Sixth Workshop on Very Large Corpora*, Montreal, 1998.

[18] M. Jardino. "Multilingual stochastic n-gram class language models," *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 161-63, Atlanta, 1996.

[19] M. Jardino, and G. Adda. "automatic word classification using simulated annealing", *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 41-44, Minneapolis, 1993.

[20] F. Jelinek. *Statistical Methods for Speech Recognition.* MIT Press, Cambridge, Massachusetts, 1997.

[21] S. Katz. "Estimation of probabilities from sparse data for the language model component of a speech recognizer," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 35, pp. 400-401, 1987.

[22] R. Kneser, and H. Ney. "Improved clustering techniques for class-based statistical language modeling," *Proceedings of the European Conference on Speech Communication and Technology*, pp. 973-976, Berlin, 1993.

[23] G. Maltese, and F. Mancini. "An automatic technique to include grammatical and morphological information in a trigram-based statistical language model," *Proceedings of*

the IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 157-60, San Francisco, 1992.

[24] M. Marcus, Beatrice Santorini, and M.A. Marcinkiewicz. "Building a large annotated corpus of English: The Penn Treebank," *Computational Linguistics*, vol. 19, no. 2, pp. 313-330, 1993.

[25] S. Martin, J. Liermann, and H. Ney. "Algorithms for bigram and trigram word clustering," *Speech Communication*, vol. 24, pp. 19-37, 1998.

[26] S. Mori, M. Nishimura, and N. Itoh. "Word clustering for a word bigram model," *Proceedings of the International Conference on Spoken Language Processing*, pp. 2467-2470, Syndey, 1998.

[27] H. Ney, U. Essen, and R. Kneser. "On structuring probabilistic dependencies in stochastic language modeling," *Computer Speech and Language*, vol. 8, pp. 1-38, 1994.

[28] T.R. Niesler, and P.C. Woodland. "A variable-length category-based n-grams for language model," *Proceedings of the IEEE International Conference on Audio, Speech and Signal Processing*, pp. 164-167, Atlanta, 1996.

[29] T.R. Niesler, and P.C. Woodland. "Combination of word-based and category-based language models," *Proceedings of the International Conference on Spoken Language Processing*, pp. 220-223, Philadelphia, 1996.

[30] T.R. Niesler, and P.C. Woodland. *Comparative evaluation of word-based and category-based language models.* Tech. Report CUED/F-INFENG/TR.265, Dept. Engineering, University of Cambridge, U.K., July 1996.

[31] T.R. Niesler, E.W.D. Whittaker, and P.C. Woodland. "Comparison of part-of-speech and automatically derived category-based language models for speech recognition," *Proceedings of the IEEE International Conference on Audio, Speech and Signal Processing*, pp. 177-180, Seattle, 1998.

[32] R. Rosenfeld. *Adaptive Statistical Language Modeling: A Maximum Entropy Approach*, PhD thesis, School of Computer Science, CMU, April 1994.

[33] K. Seymour, S. Chen, S. Doh, M. Eskenazi, E. Gouvea, B. Raj, M. Ravishankar, R. Rosenfeld, M. Siegler, R. Stern, and E. Thayer. "The 1997 CMU Sphinx-3 English broadcast news transcription system," *Proceedings of the DARPA Broadcast News Transcription and Understanding Workshop*, Virginia, 1998.

[34] B. Srinivas. "Almost parsing technique for language modeling," *Proceedings of the International Conference on Spoken Language Processing*, pp. 1169-1172, Philadelphia, 1996.

[35] U. Uebler and H. Niemann. "Morphological modeling of word classes for language models," *Proceedings of the International Conference on Spoken Language Processing*, pp. 1687-1690, Sydney, 1998.

[36] Y. Wakita, J. Kawai, H. Iida. "An evaluation of statistical language modeling for speech recognition using a mixed category of both words and parts-of-speech," *Proceedings of the International Conference on Spoken Language Processing*, pp. 530-533, Philadelphia, 1996.

[37] P. Witschel. "Optimized POS-based language models for large vocabulary Speech recognition," *Proceedings of the International Conference on Spoken Language Processing*, pp. 2543-2546, Sydney, 1998.

[38] P.C. Woodland, T. Hain, G.L. Moore, T.R. Niesler, D. Povey, A. Tuerk, and E.W.D. Whittaker. "The 1998 HTK broadcast news transcription system: development and results," *Proceedings of the DARPA Broadcast News Workshop*, Virginia, 1999.