A Cache-Based Natural Language Model for Speech Recognition

© Roland Kuhn, School of Computer Science

McGill University, Montreal

August, 1988.

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of M. Sc.

Abstract

Speech recognition systems must often decide between competing ways of breaking up the acoustic input into strings of words. Since the possible strings may be acoustically similar, a **language model** is required; given a word string, the model returns its linguistic probability. This thesis discusses several Markov language models. Subsequently, we present a new kind of language model which reflects short-term patterns of word use by means of a **cache component,** analogous to "cache memory" in hardware terminology. The model also contains a **Markov component** of the traditional type. The combined model and a pure Markov model were tested on samples drawn from the LOB (Lancaster-Oslo/Bergen) Corpus of English text. We discuss the relative performance of the two models, and make suggestions for future improvements.

Resumé

Les systèmes de reconnaissance de la voix doivent souvent évaluer plusieurs alternatives afin de regrouper les sons en mots. Une même suite de sons peut être interprétée de plusieurs façons, d'où le besoin d'un modèle associé à la langue. Les suites de mots possibles peuvent être testées par le modèle pour en obtenir la probabilité linguistique. Cette thèse porte sur différents modèles Markoviens du langage. Un nouveau modèle qui ajuste la probabilité d'un mot selon qu'il a été utilisé dans un passé récent est présenté. Ce modèle comporte une composante similaire à une mémoire cache ainsi qu'une composante Markovienne traditionelle. Le nouveau modèle et le modèle Markovien pur ont été testés sur un échantillon de données tiré de l'ouvrage LOB ("Lancaster-Oslo Bergen Corpus of English Text"). La performance relative des deux modèles est presentée, suivie de suggestions pour des améliorations futures.

Acknowledgements

First, I would like to thank my supervisor, Renato De Mori, for his role in discussing and criticizing my early ideas for this thesis, and for the exemplary patience he showed in waiting for its completion. I have greatly appreciated the skill and good humour shown by the systems staff of the McGill School of Computer Science in dealing with my unreasonable demands; my thanks to Peter Deutsch, Luc Boulianne, and Christopher Rabson. I am grateful to Andrew McGregor for arranging for me to discuss this work with his colleagues at Bell Northern Research; he, Matthew Lennig, and others made many valuable suggestions and criticisms. Warm thanks go to my parents, whose help made it financially possible for me to present a paper at the COLING 1988 conference in Budapest, Hungary. Finally, my wife Susan cheerfully accepted the necessity of our geographical separation while I completed this research, although it was our first year of marriage - I shall always remember this with love and gratitude.

TABLE OF CONTENTS

I. Introduction

A type of system popular today for Automatic Speech Recognition (ASR) consists of two components. An acoustic component matches the acoustic input to words in its vocabulary, producing a list of the most plausible word candidates together with a probability for each. The second component, which incorporates a language model, utilizes the string of previously identified words to estimate for each word in the vocabulary the probability that it will occur next. Each word candidate originally selected by the acoustic component is thus associated with two probabilities, the first based on its resemblance to the observed signal and the second based on the linguistic plausibility of that word occurring immediately after the previously recognized words Multiplication of these two probabilities produces an overall probability for each word candidate.

Our work focuses on the language model incorporated in the second component. The language model we use is based on a class of Markov models identified by Jelinek, the "n-gram" and "Mg-gram" models [9, 10]. These models, whose parameters are calculated from a large training text, produce a reasonable non-zero probability for every word in the vocabulary during the speech recognition task. Our **combined model** incorporates both a Markov 3g-gram component and an added "cache" component which tracks short-term fluctuations in word frequency. The addition of the cache component and the evaluation of its effects are the original contributions of this thesis.

We adopted the hypothesis that a word used in the recent past is much more likely to be used soon than either its overall frequency in the language or a Markov model would suggest. The cache component of our **combined model** estimates the probability of a word from its recent frequency of use. The model uses a weighted average of the Markov and cache components in calculating word probabilities, where the relative weights assigned to each component depend on the Part Of Speech (POS). For purposes of comparison, we also created a pure **Markov model**, consisting of only the Markov component of the **combined model**.

For each POS, the **combined model** may therefore place more reliance on the cache component than on the Markov component, or vice versa; the relative weights were obtained

experimentally for each POS from a training text, using the Forward-Backward Method [26]. The cache-based probabilities $C_j(W,i)$ were calculated as follows. For each POS, a "cache" (just a buffer) with room for 200 words was maintained. Each new word was assigned to a single POS g_j and pushed into the corresponding buffer. As soon as there were 5 words in a cache, it began to output probabilities which corresponded to the relative proportions of words it contained. The lower limit of 5 on the size of the cache before it starts producing probabilities, and the upper size limit of 200, are arbitrary - there are many possible heuristics for producing cache-based probabilities.

The dependence on POS in the **combined model** arose from the hypothesis that a content word, such as a particular noun or verb, will occur in bursts. Function words, on the other hand, would be spread more evenly across a text or a conversation; their short-term frequencies of use would vary less dramatically from their long-term frequencies. One of the aims of our research was to assess this hypothesis experimentally. If it is correct, the relative weight calculated from the training text for the cache component for most content POSs will be higher than the cache weighting for most function POSs.

Our research was greatly facilitated by the availability of a large and varied collection of modern English texts, in which each word is labelled with an appropriate POS. This is the Lancaster-Oslo/Bergen (LOB) Corpus of modern English, consisting of 500 samples (drawn from 15 different categories) of texts published in the United Kingdom in 1961. This corpus is described by Johansson and others in [12, 13, 14]; it is available to researchers from the Norwegian Computing Centre for the Humanities. We chose to employ the same 153 POSs found in the LOB Corpus in our model, in the belief that it was more rational to rely on the syntactical judgments of a large team of trained grammarians and lexicographers than to devise our own idiosyncratic POSs. Part of this corpus (391,658 words) was utilized as a training text which determined the parameters of both models: the standard 3g-gram **Markov model**, and our **combined model** consisting of the same Markov model along with a cache component.

We required a yardstick with which to compare the performance of the two models. The measure chosen is called "perplexity"; it was devised by F. Jelinek [11]. The perplexity of a model

can be estimated by the success with which it predicts a sample text (which should NOT be the one used to train the model). The better the model, the higher the probability it will assign to the sequence of words that actually occurs in the sample text. To compare two models, one employs each to calculate the word-by-word probability of the same sample text One can then calculate the average probability per word of sample text given by each of the two models; the model for which this average probability is higher is better than the other. The perplexity is simply the reciprocal of this average probability - low perplexity implies good performance.

Once the parameters of the two models, the pure **Markov** and the **combined**, had been calculated from part of the LOB Corpus, we could have used any sample text from any source whatsoever to compare the perplexity of the models We chose to use part of the remaining portion of the LOB Corpus because of the wide range of different types of text represented therein. The sample text we constructed (like the training text) includes such diverse types of written English as press reports, religious literature, love stories, and government documents This sample text posed a difficult challenge to the two models · if a model performs well on such a variety of written material, it is likely to perform well on most types of written English

The results of the comparison between the two models exceeded our expectations Jelinek [11] gives two figures for the perplexity of his trigram model, which was trained on a 2.5 million-word collection of office correspondence. When part of this training text was used to estimate the perplexity of the model, the result was 70 This result is unimportant, as it measures the fit of the model to data used to calculate its parameters. When a collection of memos not in the training text was used, estimated perplexity was 128. Our results are not strictly comparable to Jelinek's, for two reasons. First, our training text was much smaller than his, which should make both of our models less reliable. Second, the sample text we used to estimate perplexity was much more varied than his, which should make prediction harder. For both these reasons, even if our models were as good as his, we would expect to get a higher perplexity estimate. This was indeed the case for the pure **Markov model**, whose estimated perplexity was 332. However, the **combined model** had an estimated perplexity of 107. This is a significant improvement over Jelinek's result.

These results indicate that addition of a cache component to a language model can lead to dramatic improvement in the performance of the model, as measured by its perplexity. The cache component reflects short-term fluctuations in the frequency of word use, on the premise that different writers or speakers have idiosyncratic word frequencies. Furthermore, a given subject or context may demand a particular set of word frequencies. The cache component of our **combined model** represents a cheap, easily-implemented technique for permitting Automatic Speech Recognition Systems to track these short-term fluctuations in frequency of word use, whatever their cause.

Interestingly, one of the hypotheses we tested in the course of this research was disproved. We thought it likely that the usefulness of the cache component would depend on the POS, with content words such as nouns and verbs being more affected by context than function words such as articles and prepositions, which would not vary much from their overall frequency in the English language. Hence, we expected higher best-fit weights for the cache component of content POSs than for the cache component of function POSs. This turned out to be false. When the best-fit values for the weightings assigned to the cache component for each POS were determined experimentally by means of the Forward-Backward Method, they did vary considerably from POS to POS. But there was no consistent trend of high values for content POSs and lower ones for function POSs. If anything, the pattern was the reverse This and other aspects of the results are discussed in the Conclusion.

A preliminary version of the work presented here can be found in [18].

II. Markov Models for Natural Language

2.1 Mathematical Background

An Automatic Speech Recognition (ASR) system takes an acoustic input, A, and derives from it a string of words $W_1, W_2, \cdots, W_n$ taken from the system's vocabulary, V. In the course of this process, the system considers a set of plausible word strings, assigns each a probability, and outputs the candidate with the highest probability. If V is large, the candidates may all be equally plausible on acoustic grounds. Thus, the system requires a purely linguistic component which assigns probabilities to word strings.

Formally, let $WS = < W_1, W_2, \ldots, W_n >$ denote one of these possible word strings and $P(WS \mid A)$ the probability that it was uttered, given the acoustic evidence A. Then the speech recognizer will pick the word string $\hat{WS}$ satisfying $P(\hat{WS} \mid A) = \max_{WS} P(WS \mid A)$, i.e., the most likely word string given the evidence.

Since by the Bayes Formula we have $P(WS \mid A) = \frac{P(WS) \cdot P(A \mid WS)}{P(A)}$ it follows that $P(\hat{WS} \mid A) = \max_{WS} \frac{P(WS)P(A \mid WS)}{P(A)}$. Since A is fixed, it follows that

$\hat{WS} =$ { WS such that $P(WS) \cdot P(A \mid WS)$ is a maximum }

In this report, we will not discuss $P(A \mid WS)$, the probability of the actual acoustic input being observed if the string $WS = < W_1, \ldots, W_n >$ is uttered. The calculation of $P(A \mid WS)$ is the responsibility of the signal - processing component of the speech recognition system. We are concerned instead with the model that estimates $P(WS)$, the probability of a given word string **independent of the acoustic input.**

From elementary probability theory, we decompose $P(WS)$ as

$$P(WS) = P(W_1) \cdot \prod_{i=2}^{n} P(W_i \mid < W_1, \ldots, W_{i-1} >).$$

Thus, the probability that a word W_i is spoken depends on the past history of the dictation. As Jelinek, from whom the above account is derived [11, pp. 2-3] points out, the probabilities

$P(W_i \mid < W_1,\ldots,W_{i-1} >)$ are in practice impossible to estimate, since each history $< W_1,\cdots,W_{i-1} >$ has occurred at most only a few times in the history of the English language. For a vocabulary of size V, there are V^{i-1} different possible histories; since $P(W_i = \mathrm{W} \mid < W_1,\ldots,W_{i-1} >.)$ must be found for each possible W, that is for each word in V, there are V^i different probabilities to be estimated. V^i is an astronomically large number for reasonable values of V and i - thus, another approach must be found.

Whatever solution we adopt will consist of mapping the set of possible histories $< W_1,\ldots,W_{i-1} >$ into a more manageable number of equivalence classes. Jelinek denotes this many to one mapping by S. Thus, $S(< W_1,\ldots,W_{i-1} >)$ denotes the equivalence class of the string $< W_1,\ldots,W_{i-1} >$.

The probability $P(W_i{=}W)$ is approximated by
$P(W_i{=}W) = P(W_i{=}W \mid S\,(< W_1,\ldots,W_{i-1} >))$.

Any language model for speech recognition - whether past present, or future, influenced by the work of Jelinek's group or not, complicated or simple - will consist of such a mapping S of word strings into equivalence classes.

2.2 Justification of Jelinek's Markov Approach

Whenever we design a system intended to achieve human performance levels in the accomplishment of a certain task, there are two strategies we could follow. These might be termed the "anthropomorphic" strategy and the "abstract" strategy. The first requires that we learn as much as possible about how human beings perform the given task, and then incorporate this knowledge in the software. The second demands that we consider the task in the abstract as a problem to be solved and find the algorithm for solving it that will run most efficiently on our machines. Thus, the details of the procedure we come up with might converge on human strategies in the task domain as we learn more about these strategies, or diverge from them as our approach to the task becomes increasingly abstract and as we exploit our hardware more effectively.

There is no a priori reason for preferring one approach to the other - it depends on our focus of interest and on the task domain. Expert systems arose out of programs that modeled the haphazard, intuitive thought processes of human experts; earlier, "logical" approaches failed in many domains to which expert systems were later successfully applied. Computer chess has evolved in the opposite direction. For several years attempts were made to unravel the complex thought processes of the best human players, and to write programs that incorporated them. Today, these attempts have been more or less abandoned; the best chess programs rely on exhaustive searching of game trees, though one of the few things that is known with certainty about human grand masters is that they do **not** rely on exhaustive search. In the case of chess, it may turn out that the secret of a good program is choosing an algorithm adapted to the hardware. There is no reason to suppose that procedures that work well in an extremely slow, highly connected machine with a vast memory (the human brain) are the most suitable ones for a fast sequential machine with a comparatively small memory (the computer).

Language models held by humans undoubtedly incorporate knowledge about syntax, semantics and the pragmatics of discourse, as well as knowledge about the world and often about the psychology of an individual speaker. Of these knowledge sources, only syntax can claim to have been successfully formalized - or so linguists would have us believe. There is as yet no complete formal grammar for the English language. Furthermore, few of the innumerable parsers in the literature are equipped to make probability estimates; most would assign a probability of 0 to ungrammatical sentences, though these occur with high frequency in spoken English.

Thus, the case for an "abstract" strategy in natural language modeling for speech recognition is very strong. The exceptions occur in specialized domains where the vocabulary, syntax or semantics are so constrained that the mechanisms underlying speech recognition by human beings within the domain can be guessed at and incorporated into a parser. Where unconstrained human speech is concerned, "unmodified parsers can at best serve as final filters accepting word strings that were arrived at with the help of a more appropriate language model" [11,pg.3].

The Markov models employed by Jelinek and his group, therefore, are not intended to reflect natural language models possessed by humans. Instead, they are designed to produce a mapping S of word strings to equivalence classes that facilitate estimation of

$P(W_i = W \mid S(< W_1,\ldots,W_{i-1} >))$.

This involves a compromise between the need for a refined classification that loses little relevant information about the history $< W_1,\ldots,W_{i-1} >$ and the need for a small number of classes so that enough data can be gathered for each one.

The novelty of Jelinek's approach is that he decided it was more important to keep the information contained by the last few words than to concentrate on syntax, which by definition involves the entire sentence. A high percentage of English speech and writing consists of stock phrases that reappear again and again; if someone is halfway through one of them, we know with near-certainty what his next few words will be. Jelinek's **trigram model** automatically picks up this kind of information from a training text; a parser does not. The **3g-gram model** addresses the same task parsers are designed to achieve - the prediction of the part of speech of the next word - but its structure owes everything to Jelinek's approach and nothing to traditional parsers. Like the trigram model, the 3g-gram model uses only the context provided by the two preceding words.

The advantages of the Jelinek approach are the assignment of a probability to every possible word string and the automatic calculation of parameters from a training text, permitting the model to incorporate valuable information that is not described by any existing linguistic theory. An important disadvantage is the loss of information that goes more than a few words back. Furthermore, parameter values do not change once the training text has been processed - no information is gathered in the course of the speech recognition task, so that there is no way for the system to adapt itself dynamically to the speaker. In the next chapter, I will discuss our **combined model**, which tries to overcome both disadvantages by using lexical information gathered during the recognition task. The Markov component of this model is based on the 3g-gram model, which is an adaptation of Jelinek's original trigram model. Section 2.3 of this chapter is therefore concerned with the trigram model, while section 2.4 describes the 3g-gram model.

2.3 The Trigram Model

The **trigram model** is based on the mapping of a history $< W_1,\ldots,W_{i-1} >$ onto the state formed by the two most recent words:

$$S\ (< W_1,\ldots,W_{i-1} >) = < W_{i-2},\ W_{i-1} >.$$

Thus, it is a Markov model, approximating $P(W_i=W \mid < W_1,\ldots,\ W_{i-1} >)$ by $P(W_i=W \mid W_{i-2},W_{i-1})$. The latter, in turn, is estimated from the training text as the ratio of the number of times the word sequence $< W_{i-2},W_{i-1},W >$ occurred to the number of times the sequence $< W_{i-2},W_{i-1} >$ occurred:

$$P(W_i=W \mid W_{i-2},\ W_{i-1}) \simeq f\ (W \mid W_{i-2},\ W_{i-1}) = \frac{N(W_{i-2},W_{i-1},W)}{N(W_{i-2},W_{i-1})}.$$

In practice many trigrams that do not occur in the training text show up during the recognition task, and should therefore not have the zero probability assigned them by this formula. One way of dealing with this problem is to use a weighted average of trigram, bigram, and individual word frequencies:

$$P(W_i=W \mid W_{i-2},\ W_{i-1}) \simeq q_2\ f\ (W_i=W \mid W_{i-2},\ W_{i-1}) + q_1\ f\ (W_i=W \mid W_{i-1}) + q_0\ f\ (W_i=W),$$

where $q_0 + q_1 + q_2 = 1$ and

$f\ (W_i=W \mid W_{i-2},W_{i-1}) = N(W_{i-2},W_{i-1},W)/N(W_{i-2},W_{i-1})$, as before,

$f\ (W_i=W \mid W_{i-1}) = N(W_{i-1},W)/N(W_{i-1})$,

and $f\ (W_i=W) = N(W_i=W)/NT$,

where NT = total number of words in training text.

If $q_0 \neq 0$, this smoothed trigram model guarantees that any word W that occurs at least once in the training text is assigned a non-zero probability, so it avoids the problem with the pure trigram model mentioned above. The calculation of the values for q_0, q_1, and q_2 is quite

complicated. They are chosen to meet the maximum likehood criterion - that is, the probability of a new text calculated by means of the smoothed trigram formula is maximized. Note that these q_j's depend on the size of the training text, since as it gets larger more of the possible trigrams are encountered, $f(W_i=W \mid W_{i-2}, W_{i-1})$ becomes a more reliable estimator, and the value of q_2 can be increased.

There are other ways of using bigram and singlet frequencies to smooth trigram estimates. S. Katz's method "backs off" from a trigram to a bigram to a singlet estimate ([15]; described in [8, 11]):

$$P(W_i=W \mid W_{i-2}, W_{i-1}) = \text{ if } N(W_{i-2}, W_{i-1}, W) > 0$$
$$\text{then } r_2 f(W_i=W \mid W_{i-2}, W_{i-1})$$
$$\text{else if } N(W_{i-1}, W) > 0$$
$$\text{then } r_1 f(W_i=W \mid W_{i-1})$$
$$\text{else } r_0 f(W_i=W).$$

The weightings r_2 , r_1 and r_0 ensure that the probability summed over all words W adds up to 1; as with the q_j's in the previous model, they are chosen to maximize the probability of a new text, and depend on the size of the training text.

2.4 The 3g-gram Model

The 3g-gram model (terminology of A. Martelli [21]) is analogous to the trigram model; this model is also Markov, but not completely divorced from grammatical theory. It employs grammatical **parts of speech** - henceforth abbreviated "POS". Let $g(W_i) = g_i$ denote the POS of the word that appears at time i. Note that we might have $W_i = W = W_k$ for i $\neq$ k, but $g(W_i) \neq g(W_k)$. This is because a word W in the vocabulary can belong to different POSs at different times; for instance, "light" can be a noun, verb, or adjective. By definition, each occurrence of a word only has one POS; in practice, it may be difficult to single out that POS among the set of

POSs associated with the word.

The 3g-gram model has two levels. At time i, it assigns a probability to each POS on the basis of the information provided by g_{i-1} and g_{i-2}. This part of the model functions exactly like the trigram model, except that the vocabulary consists of POSs and not words. Thus, the model gives a non-zero probability that g_i is a noun or a verb or an article, etc. Next, probabilities of individual words are calculated on the basis of their frequency within POSs. Suppose that the model gave a probability of 0.99 to the occurrence of a noun at time at time i. Then the estimated probability that W_i = "desk" would be almost exactly equal to the frequency of the word "desk" among the nouns in the training text.

Let G be the set of POSs recognized by our model, and let g_j be a particular POS whose probability of occurring we wish to predict. The model gives us

$$P(g_i = g_j \mid g_{i-2}, g_{i-1}) = f(g_j \mid g_{i-2}, g_{i-1}).$$

For a word W that only has one possible POS, g(W), the probability $P(W_i=W)$ is estimated by the product of the estimated probability that g(W) will occur at time i by the estimated probability that if g(W) occurs the word will be W:

$$P(W_i=W \mid g_{i-2}, g_{i-1}) \simeq P(W \mid g(W)) \cdot P(g_i=g(W) \mid g_{i-2}, g_{i-1})$$
$$\simeq f(W \mid g(W)) \cdot f(g_i=g(W) \mid g_{i-2}, g_{i-1})$$

where the frequencies f are calculated from the training text as before.

Generally things are not as simple as this, since many words belong to more than one POS category. The probability that "light" will occur is the probability that it will occur as a noun plus the probability that it will occur as a verb plus the probability that it will occur as an adjective. Thus, the general 3g-gram formula is:

$$P(W_i=W \mid < W_1,\ldots,W_{i-1}>) = \sum_{g_j \in G} P(W \mid g_j) \cdot P(g_i=g_j \mid g_{i-2}, g_{i-1})$$
$$\simeq \sum_{g_j \in G} f(W \mid g_j) \cdot f(g_i=g_j \mid g_{i-2}, g_{i-1}).$$

Given a sufficiently large training text, $f(g_i{=}g_j \mid g_{i-2}, g_{i-1})$ could be calculated for every POS g_j in G. In practice, existing training texts are too small - many POS triplets will never appear in the training text but will appear during a recognition task. If we do not modify the procedure to prevent zero probabilities, a particular g_j that actually occurs may have zero estimated probability.

Recall that an analogous problem occurred with the trigram model. The two solutions we described were the "weighted average" approach and the "back-off" approach, both using bigram and singlet frequencies to smooth out the trigram frequencies. These two solutions are also applicable to the 3g-gram model.

Derouault and Merialdo [4,5] employed a variant of the weighted average 3g-gram approach. Their work will be described in some detail, as the Markov component of our model was based on it. It must be emphasized that not all of their conclusions are relevant to our work, as they were dealing with French rather than English. However, their methods are applicable to English.

Their corpus consisted of 1.2 million words of French text tagged with 92 POSs. Only 5 percent of the possible triplets occurred. Thus, the doublets were tabulated as well; this time half of the possible pairs occurred. Instead of using individual POS frequencies as the third component of a weighted average, these researchers chose to add an arbitrary small value $e = 10^{-4}$ to the weighted average of triplet and doublet POS frequencies in order to prevent zero estimates for the probability of occurrence of a given POS. Thus, they approximated $P(g_i{=}g_j \mid g_{i-2}, g_{i-1})$ by

$$l_1 f(g_i{=}g_j \mid g_{i-2}, g_{i-1}) + l_2 f(g_i{=}g_j \mid g_{i-1}) + e,\ e = 10^{-4},\ l_1 + l_2 = 1.$$

They experimented with two different ways of calculating l_1 and l_2. Intuitively, it makes sense that if there are many triplets beginning $<g_{i-2}, g_{i-1}, \ldots>$, the frequency $f(g_i{=}g_j \mid g_{i-2}, g_{i-1})$ gives reliable information; l_1 should therefore be high. If there are few such triplets, l_2 should be given more weight. Following this reasoning, Derouault and Merialdo first let l_1 and l_2 be a function of the count of occurrences of $<g_{i-2}, g_{i-1}>$. Each possible history $<g_{i-2}, g_{i-1}>$ was assigned to one of ten groups, depending on how often it had occurred in the training text. Each of the groups had different values of l_1 and l_2, with the highest value of l_2 occurring in the group for histories $<g_{i-2}, g_{i-1}>$ that never occurred in the training text.

Another way of looking at the problem is to argue that l_1 and l_2 should depend on g_{i-1}, the POS of the last word recognized. If it is an article, for instance, we can be almost certain that the next word, W_i, is a noun or an adjective. In other cases, we may have to look at g_{i-2} as well. Thus, the other way in which these researchers calculated l_1 and l_2 was to allow them to depend on g_{i-1}.

Let $h(<g_{i-2},g_{i-1}>)$ denote the parameter on which l_1 and l_2 depend. For Derouault and Merialdo's first approach, h = $N(<g_{i-2},g_{i-1}>)$ = the number of occurrences of $<g_{i-2},g_{i-1}>$ in the training text; for the second approach, h = g_{i-1} = the POS of the preceding word. They calculated $l_1(h)$ and $l_2(h)$ by the same algorithm in both cases, called the Forward-Backward Method [26]. Having split the training text into two portions in the ratio 3:1, they used the larger portion to calculate $f(g_i \mid g_{i-2},g_{i-1})$ and $f(g_i \mid g_{i-1})$. They then set $l_1(h)$ and $l_2(h)$ to arbitrary values such that $l_1(h) + l_2(h) = 1$, and iteratively reset them from the remaining portion of the corpus. Summing over all triplets $<g_{i-2},g_{i-1},g_i>$ in this portion, they defined

$$S_1(h) = \sum l_1(h) f(g_i \mid g_{i-2},g_{i-1}) / [l_1(h) f(g_i \mid g_{i-2},g_{i-1}) + l_2(h) f(g_i \mid g_{i-1})],$$

$$S_2(h) = \sum l_2(h) f(g_i \mid g_{i-1}) / [l_1(h) f(g_i \mid g_{i-2},g_{i-1}) + l_2(h) f(g_i \mid g_{i-1})].$$

They then redefined

$$l_1(h) = S_1(h)/(S_1(h)+S_2(h)),\ l_2(h) = S_2(h)/(S_1(h)+S_2(h)).$$

Then the first two formulas were calculated again on the same portion of the corpus. Iteration continued until $l_1(h)$ and $l_2(h)$ converged to fixed values. This procedure is guaranteed to produce the l_1 and l_2 that maximize the estimated probability of the smaller portion of the corpus, based on the frequencies obtained from the larger portion.

Derouault and Merialdo found only a small difference between the performance of the model in which l_1,l_2 depend on the count $N(<g_{i-2},g_{i-1}>)$ and that in which they depend on the POS g_{i-1}. Both models were superior to one in which the coefficients were arbitrarily set to $l_1 = 0.99$, $l_2 = 0.01$ for all POS. As expected, when training text size was varied, the algorithm described above gave larger values of l_1 for larger text size.

The first level of both our **combined model** and our **Markov model** - the level that predicts the POS - works in exactly the way we have described for Derouault and Merialdo's 3g-gram model. The other level to be considered is the lexical level, which estimates the probability of a word given its POS. At this level, our **Markov model** is again almost identical to Derouault and Merialdo's model. In both cases, the probability of a word given its POS is estimated by its frequency among the words found in that POS category in the training text. Thus the only substantial difference between our **Markov model** and Derouault and Merialdo's model is the choice of POSs; they define 92 POSs, we use the 153 POSs in the LOB Corpus. In the next chapter, we will see how the **combined model** differs from the **Markov model** and Derouault and Merialdo's model at the level of lexical prediction.

2.5 Perplexity : A Measure of the Performance of a Language Model

This section is a summary of work done by Jelinek (Appendix A in [11]). We wish to derive an objective measure of language model quality based on Information Theory. Consider a source putting out symbols from a finite set V that is known to the user. The output of a symbol removes the user's uncertainty about the identity of that symbol, and thus provides him with information. For a given size L of the set V, the information is maximal if each of the possible symbols is chosen with equal probability, independently of previously chosen symbols. If the source has these characteristics, the information content is

$$I = \log_2 L .$$

In general, let x denote a symbol put out by the source with probability P(x). Then the measure of information (if the symbols are chosen independently of one another) is given by

$$H = -\sum_x P(x)\log_2 P(x).$$

Thus, a source of entropy H has as much information content as one whose symbols are chosen equiprobably from an alphabet of size 2^H.

If x_i is the ith symbol put out by a well-behaved, "ergodic" source, we can estimate its entropy by

$$H = -(1/n)[log_2 P(x_1, x_2, \cdots, x_n)],$$

where n should be as large as possible. Applying this to language, we can view a language as a source whose output symbols are words w_i. Unfortunately, we cannot know the probabilities $P(w_1, w_2, \cdots, w_n)$ for strings of a language. However, each language model provides an estimate $\hat{P}(w_1, w_2, \cdots, w_n)$ for such strings.

The difficulty of recognition of a sample text using a given language model is therefore given by

$$LP = -(1/n)[log_2 \hat{P}(w_1, w_2, \cdots, w_n)].$$

Jelinek suggests that it is intuitively more satisfying to measure the difficulty of the speech recognition task by the value of the **perplexity** given by

$$PP = 2^{LP} = \hat{P}(w_1, \cdots, w_n)^{(-1/n)}.$$

Roughly speaking, if the perplexity is PP, the speech recognition task is as difficult as it would be if the language had PP equiprobable words.

There is another way of looking at the perplexity which Jelinek does not mention. When we employ language models to calculate the probability of a sample text, the better models will assign a higher probability to it (since they are better at prediction than the others). Thus, the better the model, the higher the average probability per word. How could one estimate this average for a text of n words ? The logical answer is to take the nth root of the sample's overall probability as estimated by a given model, since the individual probabilities are multiplicative. But this nth root is simply the reciprocal of Jelinek's perplexity measure. Thus, low perplexity corresponds to high probability per word of sample text; both are signs that the model in question is a good predictor for the sample.

III. The Combined Model

3.1 Argument for the Cache Component

The central idea underlying the work presented in this thesis concerns a crucial limitation of all the Markov models described earlier. Fortunately, this limitation can easily be overcome by means of a mechanism which does not compromise the robust simplicity of the Markov approach.

The main limitation of the Markov models, as we see it, is their inability to reflect short-term patterns of word use. Suppose the word sequence "the old ..." has just been recognized, and that the word "man" followed these two words 10% of the time in the training text, while the word "band" followed them 1% of the time. The trigram model will assign "man" a probability of 0.1 and "band" a probability of 0.01 . If the acoustic component assigns these words roughly equal probability, "man" will be chosen. For an isolated sentence, this would be the reasonable choice to make. But now suppose that several previous sentences contained the word "band", while none contained the word "man". We contend that a human being would then assign overwhelmingly higher probability to the word "band", and that he would be right to do so. A word used in the immediate past - say the last 2,000 words or so - is **much** more likely to be used soon than either its overall frequency in the English language or any of the popular Markov models would suggest.

There is strong empirical evidence for this. Studies on three corpora of English and American texts by S. Johansson [13, 14] show that "word frequencies vary greatly depending upon the type of text, both among content words and function words" [14, pg. 34]. Jelinek [11] was certainly aware of this fact. "His choice of words reflects the speaker's habits of expression that are related, for example, to his level of education. His usage is also conditioned by the general domain of discourse (e.g. data processing, musicology, medical reports, etc.) that calls for a variety of technical terms, cliches, and such" [11, pg. 20]. However, in the passage I have just quoted, Jelinek was primarily concerned with the question of what words to include in the speech recognition system's vocabulary, rather than with the way in which the current context (the identity of the speaker and his domain of discourse)

affect the frequencies of occurrence of the words in the vocabulary. He did briefly consider the latter problem, emerging with a rather pessimistic conclusion.

"For a dynamically changing active (personalized) vocabulary, a straight trigram language model cannot be constructed solely from the text created by the user. He will never dictate enough. The only conceivable way is to extract the model from an already stored trigram collection appropriate to a large vocabulary that includes the active subset. It is obvious that the more the active vocabulary of size L differs from the L most frequent words of the fixed training corpus, the less will the trigrams in the dictated text be covered by trigrams collected from the training text. Therefore a truly gigantic corpus would have to be used as a basis for a satisfactory model".

"A more powerful method of language model construction is required not only to accommodate dynamic vocabularies, but also to limit the need to produce a large number of very different models for the many discourse domains. In fact, it is hard to list the latter, and for some important domains it will be impossible to find corresponding data bases" [11, pg. 21].

The idea underlying our research was that a language model that exploited short-term shifts in word-use frequencies might perform significantly better than the pure Markov models described in the previous chapter. A similar problem was faced by computer hardware designers some years ago [17, 25]. It was known that computers often accessed a particular memory location with high frequency within a sequence of accesses. The designers took advantage of these short-term patterns in memory reference by building a small, high-speed, expensive "cache memory" next to the CPU. "When a memory access is made, the contents of the accessed location, plus its neighbours, is copied to the cache. If another reference is made to these locations they can be fetched directly from the cache without having to go to the slower-speed main memory. For a reasonably sized cache a hit rate of 80 percent is common" [25, pg.230]. When space must be made in the cache to insert new information, the least recently used ("LRU") data is overwritten.

Following this analogy, we decided to design a language model that had both a **cache component** and a **Markov component**. Our linguistic intuition suggested that these short-term word frequency fluctuations depend on the POS. For example, a given noun will appear in bursts

whenever a topic that evokes it is being discussed; a given preposition, we thought, would be likely to appear at a steady rate throughout. This consideration led us to employ a model with a component that predicts the POS, so that the model would be able to weight the short-term cache component heavily when, for example, a noun was expected, while virtually ignoring the cache component when a preposition was expected. Any Markov model that predicts the POS would have suited us - we chose the 3g-gram model because it has been thoroughly studied and well described in the literature. Just as was described for the l-values in the Derouault-Merialdo implementation of the 3g-gram model in 2.4, the relative weights assigned to the cache and Markov components within each POS category were determined experimentally by means of the Forward-Backward Method.

Thus, the **combined model** assigns a probability to each POS in the same way as the 3g-gram model. For a fixed POS, the probability of any word which belongs to it is a weighted average of the word's frequency in that POS category in the training text - the Markov component - and its frequency in the cache belonging to the POS category - the cache component. At a given time during the speech recognition task, the cache for a POS will contain the last N words which were guessed to have that POS (we arbitrarily set N to 200). If a word has occurred often in the recent past, it will occur many times in the cache for its POS (supposing for the purposes of argument that the word only has one possible POS). Thus the word will be assigned a higher probability than when its recent frequency of occurrence is low. In this way, the inclusion of a cache component satisfies our goal of dynamically tracking changing patterns of word use.

3.2 Mathematical Treatment of the Combined Model

It is easy to describe the **combined model** mathematically. Recall that the pure 3g-gram **Markov model** is

$$P(W_i=W \mid g_{i-2},g_{i-1}) = \sum_{g_j \in G} P(W_i=W \mid g_i=g_j)P(g_i=g_j \mid g_{i-2},g_{i-1}).$$

The **combined model** leaves the POS component $P(g_i=g_j \mid g_{i-2},g_{i-1})$ of the 3g-gram model

unchanged; this probability was estimated in the precisely the same way as was done by Derouault and Merialdo, as described in 2.4 (above). We chose to use the variant of their model in which the l-values, giving the relative weights of the POS triplet and POS doublet probability estimates, depend on the previous POS. Our modification affects only the component that predicts the probability of a word given the POS, $P(W_i=W \mid g_i=g_j)$. In the 3-gram **Markov model** this is estimated by $f(W_i=W \mid g_i=g_j)$, calculated from the training text. This is certainly a good estimate of the mean around which the value $P(W_i=W \mid g_i=g_j)$ fluctuates; however, it does not take account of the variance around that mean.

We believe that the recent past is a good guide to the direction of the variance. Thus, let $C_j(W,i)$ denote the cache-based probability estimate for word W at time i for POS g_j. This is calculated from the frequency of W among the N most recent words belonging to POS g_j (in our implementation, N = 200). Our **combined model** estimates $P(W_i=W \mid g_i=g_j)$ by

$$k_{M,j} \times f(W_i=W \mid g_i=g_j) + k_{C,j} \times C_j(W,i), \text{ where } k_{M,j} + k_{C,j} = 1,$$

instead of by $f(W_i=W \mid g_i=g_j)$ alone. This should allow the estimate of $P(W_i=W \mid g_i=g_j)$ to deviate from its average value to reflect temporary high or low values. As described in 4.2, the relative weights of $k_{M,j}$ and $k_{C,j}$ are found by the Forward-Backward Method mentioned in 2.4; the values thus obtained maximize the probability of the training text. Note that the **Markov model** is simply the special case of the **combined model** obtained by setting all $k_{M,j}$ to 1.0 and all $k_{C,j}$ to 0.0 .

Only one major modification to this model proved to be necessary in practice. We were faced with severe memory limitations, which required that we economize on the amount of data stored. For this reason, we decided to restrict the number of POSs for which 200-word caches were maintained. To be given a cache, a POS had to meet two criteria. It had to

1). comprise more than 1% of the total LOB Corpus

2). consist of more than one word (for instance, the LOB category BEDZ was excluded because it consists of the single word "was").

Only 19 POSs met these two criteria; however, these 19 together make up roughly 65% of the LOB Corpus. They are listed in 4.1 . Thus, for POSs other than these 19, there is no cache component in the **combined model**; the estimated probability is the pure 3g-gram one, i.e. identical to that of the **Markov model**.

In order to test our hypothesis that each POS should be given a different best-fit pair of weights for its cache and Markov components, we experimented briefly with a model in which all POSs had the same pair of weights. Recall that the two weights must add up to 1.0 . We experimented with $(k_C, k_M) = (0.0, 1.0); (0.1, 0.9); (0.2, 0.8); \ldots ; (0.9, 0.1)$. We did not try to find a best-fit pair of relative weights for this simpler version of the **combined model**.

We also required an estimate of the probability that a word would be encountered in the sample text that was not in the vocabulary, i.e. was not in the training text. There are different ways of doing this [15, 21]; we chose to estimate this probability by Turing's formula, which uses the frequency of unique words among all words in the training text. There were 13,610 unique words among the 391,658 words in the training text, so the probability of encountering a word not in the vocabulary was estimated at about 0.035 . When such a word was actually encountered, it was not stored in any cache, but its POS was guessed as being the one which had maximum probability at that time according to the POS prediction part of the model.

Another serious problem is what to do when the recognition task is beginning and the cache for g_j, containing the previous words that belong to POS g_j, is nearly empty, i.e. the number of words on which our estimate is based is far less than N. One could argue that the closer a cache is to being full, the more weight its probability estimates should be given. In this view, $k_{C,j}$ should not be fixed but should increase with the number of words in the cache corresponding to POS g_j, attaining its maximum when that cache is full. However, we decided to keep things simple. Arbitrarily, we set $k_{C,j} = 0$ until the corresponding cache has 5 words in it; at that moment $k_{C,j}$ attains its maximum value. In future work, we may permit $k_{C,j}$ to increase with the number of elements in the corresponding cache.

A final point must be made. Although the use of a cache component implies the existence of a POS predictor (since short-term word frequencies differ from POS to POS), there is no reason not to merge our combined model with the trigram model. The resulting **cache-trigram model** would estimate $P(W_i=W \mid <W_1, \cdots, W_{i-1}>)$ by

$$a_1 \times f(W_i=W \mid W_{i-2}, W_{i-1}) +$$
$$a_2 \times [\sum_{g_j \in G} f(g_j \mid g_{i-2}, g_{i-1}) \times [k_{M,j} f(W \mid g_j) + k_{C,j} C_j(W,i)]\ ],$$
$$a_1 + a_2 = 1,\ k_{M,j} + k_{C,j} = 1.$$

This model would incorporate features from each of the three main approaches we have discussed so far, and might lead to substantially improved predictive power over any single one of them.

IV. Implementation and Testing of the Combined Model

4.1 The LOB Corpus and Texts Extracted from It

The Lancaster-Oslo/Bergen Corpus of British English consists of 500 samples of about 2000 words each. The average length per sample is slightly over 2000, as each sample is extended past the 2000-word mark in order to complete the final sentence. Each word in the corpus is tagged with exactly one of 153 POSs. The samples were extracted from texts published in Britain in 1961, and have been grouped by the LOB researchers into 15 categories spanning a wide range of English prose [12, 13, 14]. These categories are A - press reportage (44 samples); B - editorials (27 samples); C - press reviews (17 samples); D - religion (17 samples); E - skills, trades, and hobbies (38 samples); F - popular lore (44 samples); G - belles lettres, biography, essays (77 samples); H - miscellaneous, mostly government and industry documents (30 samples); J - learned and scientific writings (80 samples); K - general fiction (29 samples); L - mystery fiction (24 samples); M - science fiction (6 samples) ; N - adventure and western fiction (29 samples); P - love stories (29 samples); and R - humour (9 samples).

We extracted three different, non-overlapping collections of samples from the tagged LOB Corpus, and used each for a different purpose. All three were designed to reflect the overall composition of the LOB Corpus as closely as possible.

The first collection served as a **training text** for our models. That is, it was used to obtain counts for triplets, doublets, and singlets of POSs. It also gave rise to the vocabulary for the models, and to the counts for the number of occurrences of a word within each POS. It contained 169 samples altogether. There were 15 samples from category A, 9 from B, 6 from C, 6 from D, 13 from E, 15 from F, 25 from G, 10 from H, 27 from J, 10 from K, 8 from L, 2 from M, 10 from N, 10 from P, and 3 from R, for a total of 391,658 words.

The second collection was used for further **parameter setting**, including calculation of the l-values in the Derouault-Merialdo formula, which give the relative weights to be placed on triplet

and doublet probability estimates for the POS-prediction portion of both models. It was also used to calculate the k-values, which give the relative weights to be placed on the cache component and the Markov component in the **combined model**. It contained 100 samples, distributed as follows : 9 samples from LOB category A, 5 samples from B, 3 from C, 3 from D, 8 from E, 9 from F, 15 from G, 6 from H, 16 from J, 6 from K, 5 from L, 1 from M, 6 from N, 6 from P, and 2 from R.

The third collection formed the **testing text**. It was used to compare the **combined model** with the **Markov model**. It contained 100 samples distributed among the LOB categories in exactly the same way as in the parameter setting text. Note, however, that only the categories and not the samples themselves are the same.

We required labelled texts for training and parameter setting. By contrast, as pointed out in the Introduction, any text from any source whatsoever could have been used as the testing text. The diversity of the testing text poses a difficult challenge to both models we tested. It is true that the composition of the two texts used for model-building resembles that of the testing text, but that has always been the case in this type of research. It seems to us that the difficulty of prediction here, when all three texts are derived from a variety of sources, is greater than when all three are derived from business correspondence alone, as in Jelinek's work. In one way only could we be accused of making the task of the **combined model** easier. We kept samples of the same LOB category contiguous in the testing text, following the order given above. Thus, the testing text consists of the 9 A samples followed by the 5 B samples, and so on. Note that the cache component of our **combined model** will contain many words from samples previous to the current one. If, as we hypothesize, discourse of a certain type has a characteristic vocabulary and pattern of word frequencies, our **combined model** will work much better on our testing text than on one constructed from the same samples in random order. In other words, the final perplexity result for the **combined model** gives an idea of its performance when the domain of discourse changes slowly. This seems a reasonable restriction.

The comprehensiveness of the LOB Corpus made it an ideal training text and a tough test of the robustness of the language model. Furthermore, the fact that it has been tagged by an expert

team of grammarians and lexicographers freed us from having to devise our own tagging procedure.

4.2 Parameter Calculation

All parameters for both the **Markov model** and the **combined model** were calculated from the training text and the parameter setting text. The two models share a POS prediction component which is estimated by the Derouault-Merialdo method. Triplet and doublet POS frequencies were obtained from the 169-sample training text; this text also supplied the vocabulary and the count for each word, subdivided by POS. The vocabulary size can be given in two different ways. If we ignore the POS of a word, there were 24,279 different words in the training text and hence in the vocabulary of our models. On the other hand, if words with the same spelling but different POSs are counted separately, the vocabulary size is 30,718 . The 100-sample **parameter setting text** gave the weights, $l_1(g_{i-1})$ and $l_2(g_{i-1})$, needed for smoothing between the triplet and doublet POS frequencies. These were computed iteratively using the Forward-Backward algorithm described in 2.4 above .

Now the portion of both models that calculates POS probabilities is complete - it remains to find $k_{M,j}$ and $k_{C,j}$ for the **combined model**. This was calculated by means of the Forward-Backward method from the parameter setting text in exactly the same way.

4.3 Implementing the Combined Model

Because of memory limitations, it proved impossible to implement a cache for every one of the 153 POSs in the LOB Corpus. As was mentioned in 3.2, two criteria were used to select the POSs which would be assigned a cache:

a). the POS had to constitute more than 1% of the LOB Corpus

b). the POS had to contain more than one word or symbol

The second criterion is obvious - if only one vocabulary item has a given POS, the cache component

yields no extra information. The first criterion is based on the premise that rare POSs will be more spread out in time, so that the predictive power of the cache component will be weakened.

The 19 POSs that survived this selection process were as follows :

1. AT, singular article (a, an, every)
2. ATI, singular or plural article (the, no)
3. BEZ (is, 's)
4. CC, coordinating conjunction (and, and/or, but, nor, only, or, yet)
5. CD, cardinal (2, 3, etc; hundred, thousand, etc; dozen, zero)
6. CS, subordinating conjunction (after, although, etc)
7. IN, preposition (about, above, etc)
8. JJ, adjective
9. MD, modal auxiliary ('ll, can, could, etc)
10. NN, singular common noun
11. NNS, plural common noun
12. NP, singular proper noun
13. PPS, possessive determiner
14. PP3A, personal pronoun, 3rd pers plur nom (he, she)
15. RB, adverb
16. VB, base form of verb (uninflected present tense, imperative, infinitive)
17. VBD, past tense of verb
18. VBG, present participle, gerund
19. VBN, past participle

4.4 Testing the Combined Model

As described in 4.2, two parts of the LOB Corpus were used to find the best-fit parameters for the pure **Markov model** and the **combined model**, made up of the **Markov model** plus a cache

component. These two models were then tested on 20% of the LOB Corpus (100 samples) as follows. Each was given this portion of the LOB Corpus word by word, calculating the probability of each word as it went along. The probability of this sequence of 230,598 words as estimated by either model is simply the product of the individual word probabilities as estimated by that model. The higher this overall probability, the better the model. Thus the overall probability was calculated for the pure **Markov model** and for the **combined model**; the increase achieved by the latter over the former is one measure of the improvement due to addition of the cache component. This measure is somewhat unsatisfactory, however, since it depends on the number of words. Fortunately, the perplexity measure originally described by Jelinek (see 2.5 above) can easily be calculated from the overall probability and the number of words encountered. In practice, during the operation of the programs implementing the two models, the quantity calculated at each step is the log of the product of the probabilities of words previously encountered, since the product itself quickly becomes infinitesimally small.

Recall that we also tested a simpler version of the **combined model**, in which the cache component has the same weight for all POSs. The weights tried were 0.0, 0.1, 0.2, ... , 0.9; the Markov component is always 1.0 minus the cache component. The perplexity was also estimated from the testing text for these 10 variants of the simpler model.

Note that in order to calculate word probabilities, both models must have guessed the POSs of the two preceding words. Thus every word encountered must be assigned a POS. There are three cases :

a). the word did not occur in the tagged training text and therefore is not in the vocabulary;

b). the word was in the training text, and had the same tag wherever it occurred;

c). the word was in the training text, and had more than one tag (e.g. the word "light" might have been tagged as a noun, verb, and adjective).

The heuristics employed to assign tags were as follows :

a). in this case, the two previous POSs are substituted in the Derouault-Merialdo weighted-average formula and the program tries all 153 possible tags to find the one that maximizes the

probability given by the formula.

b). in this case, there is no choice; the tag chosen is the unique tag associated with the word in the training text.

c). when the word has two or more possible tags, the tag chosen from them is the one which makes the largest contribution to the word's probability.

Thus, although the portion of the LOB Corpus used for testing is tagged, these tags were not employed in the implementation of either model; in both cases the heuristics given above guessed POSs. A separate part of the program compared actual tags with guessed ones in order to collect statistics on the performance of these heuristics.

V. Results

5.1 Calculation of the L-Values

The first results of our calculations are the values $l_1(g_{i-1})$ and $l_2(g_{i-1})$, obtained iteratively to optimize the weighting between the POS triplet frequency $f(g_i \mid g_{i-2},g_{i-1})$ and the POS doublet frequency $f(g_i \mid g_{i-1})$ in the estimation of $P(g_i{=}g_j \mid g_{i-2},g_{i-1})$. As one might expect, $l_1(g_{i-1})$ tends to be high relative to $l_2(g_{i-1})$ when g_{i-1} occurs often, because the triplet frequency is quite reliable in this case. For instance, the most frequent tag in the LOB Corpus is "NN", singular common noun; we have $l_1(NN) = 0.57$. The tag "HVG", attached only to the word "having", is fairly rare; we have $l_1(HVG) = 0.17$.

However, there are other factors to consider. Derouault and Merialdo state that when g_{i-1} was an article, l_1 was relatively low because we need not know the POS g_{i-2} to predict that g_i is a noun or adjective. Thus doublet frequencies alone were quite reliable in this case. On the other hand, when g_{i-1} is a negation, knowing g_{i-2} was very important in making a prediction of g_i, because of French phrases like "il ne veut" and "je ne veux", so l_1 was high.

Our results from English texts show somewhat similar patterns. The tag "AT" for singular articles had an l_1 that was neither high nor low, 0.46 . The tag "XNOT", including only "not" and "n't", had a high l_1 value, 0.84 . Adjectives ("JJ") and adverbs ("RB") had l_1 values even higher than one would expect on the basis of their high frequencies of occurrence : 0.85 and 0.80 respectively.

5.2 Calculation of the K-Values

For each part of speech g_j, we calculated the weight $k_{C,j}$ given to the cache component of the combined model and the weight $k_{M,j}$ given to its Markov component. Recall that we originally created a different cache for each POS because we had hypothesized that the cache

component would be more useful for prediction of content words than for function words.

The following optimal weights, calculated by means of the Forward-Backward Method, decisively refute this hypothesis :

POS Description	$k_{C,j}$	$k_{M,j}$
AT (singular article)	0.999	0.001
ATI (sing. or pl. art.)	0.998	0.002
BEZ (is, 's)	0.999	0.001
CC (coord. conjunction)	0.997	0.003
CD (cardinal)	0.783	0.217
CS (subord. conjunction)	0.973	0.027
IN (preposition)	0.919	0.081
JJ (adjective)	0.402	0.598
MD (modal auxiliary)	0.989	0.011
NN (sing. noun)	0.403	0.597
NNS (pl. noun)	0.498	0.502
NP (sing. proper ncun)	0.592	0.408
PPS (possessive det.)	0.997	0.003
PP3A (pers. pron. 3rd pers. nom)	1.000	0.000
RB (adverb)	0.660	0.340
VB (verb base form)	0.456	0.544
VBD (verb past tense)	0.519	0.481
VBG (present part., gerund)	0.518	0.482
VBN (past part.)	0.326	0.673

The pattern here is just the opposite of what we had expected, with function POSs having significantly higher optimal weights for the cache component of the **combined model** than content POSs. This intriguing result is discussed in the Conclusion.

5.3 Performance of Both Models on the Testing Text

The most important results will be given first. The pure **Markov model** gives perplexity equal to 332 (average probability per word is 0.003008). This compares unfavourably to Jelinek's value of 128. On the other hand, the **combined model** gives perplexity equal to 107 (average probability per word is 0.009341). This dramatic, more than three-fold, improvement can only be attributed to the inclusion of a cache component in the **combined model**.

Would such a dramatic improvement have been obtained if all caches had had the same weight? Recall that we experimented with a simpler version of the **combined version** in which all 19 caches had the same weight. The results were as follows: for cache component weight equal to 0.1, perplexity was 180 (average prob. 0.005551); for weight of 0.2, perplexity was 152 (0.006570); for 0.3, perplexity of 137 (0.007277); for 0.4, perplexity was 128 (0.007790); for 0.5, perplexity of 122 (0.008150); for 0.6, perplexity was 119 (0.008363); for 0.7, perplexity was 118 (0.008411); for 0.8, perplexity was 121 (0.008232); and for 0 9, perplexity was 131 (0.007620). Thus the lowest perplexity, 118, was obtained when the cache component weight was 0 7 and the Markov component weight was 0.3. It is difficult to be sure without using the Forward-Backward Method to obtain the optimal weights, but these figures seem to indicate a minimum for the perplexity of this simpler version of the **combined model** of about 116 - still a vast improvement over the **Markov model**.

We collected statistics on the success rate of the POS component of both models in guessing the POS of the latest word (using the tag actually assigned the word in the LOB Corpus as the criterion). This rate has a powerful impact on the performance of both models, especially the **combined model**; each incorrectly guessed POS leads to looking in the wrong cache and thus to a cache-based probability of zero (unless the same incorrect guess has been made in the recent past). We are particularly interested in forming an idea of how fast this success rate will increase as we increase the size of the training text.

There were 230,598 words in the testing text. Of these, 14,436 (6.2%) had never been encountered in the training text and were thus assumed not to be in the vocabulary (not recognised).

Of the remaining 216,162 words that had occurred at least once in the training text, 202,882 (93.8%) had tags that were guessed correctly (6.2% incorrectly). The 14,436 words that never occurred in the training text were assigned the correct tag only 3676 times (25.4% correct, 74.6% incorrect). Recall that a word that was encountered in the training text is always assigned one of the POS tags that it had there. Apparently the information contained in the counts of POS triplets, doublets, and singlets is a good POS predictor when combined with some knowledge of the possible tags a word may have, but not nearly as good on its own. Overall, of the 230,598 words in the training text, 206,558 (89.5%) were assigned the correct POS.

Among the 216,162 words that appeared at least once in the training text, a surprisingly high number - 111,319 (51.4%) - had more than one possible POS. Of these, 99,242 (89.1%) had POSs that were guessed correctly. Of the 12,077 faulty guesses that occurred for words with more than one possible POS, only 294 (2.4%) occurred because the POS for the word in the testing text had not been encountered in the training text.

VI. Conclusions

The results listed in the previous chapter strongly confirm our hypothesis that recently-used words have a higher probability of occurrence than a pure Markov model would predict. When a 3g-gram **Markov model** and a **combined model** resembling it but containing in addition a cache component (whose effect is to assign a higher probability to recently encountered words) were used to calculate the perplexity of a testing text, the perplexity of the **combined model** was lower by a factor of more than three. This dramatic result emphasizes the utility of including a cache component in a Markov language model.

Surprisingly, the cache component seems to be even more helpful for predicting function words than it is for predicting content words, if we regard the magnitude of the best-fit weight for the cache component as an indication of the usefulness of this component. An observation that may help to explain this is that the weight of the cache component seems to be inversely proportionate to the diversity of the POS in question. For instance, POS category BEZ is not a function word category; however, it contains only the word "is" and its variant "'s". It has a best-fit cache weight of 0.999 . POS category PP3A contains only the two words "he" and "she"; its cache weight is 1.000 .

This suggests the following explanation. The size of the best-fit weight of the cache component for a given POS does not depend only on the extent to which words belonging to that POS tend to appear in bursts, as we naively assumed when we began our work; there is another factor involved. The less diverse a POS category is, the better an estimator its cache component will be of the short-term frequencies of the small number of different words belonging to that category. Content categories are usually much more diverse than function categories, and this may explain why the former tend to have lower cache weights in spite of greater "burstiness". This line of thought can be illustrated by what causes the worst performance of the cache component. This will occur when the current word is not in the appropriate cache, so that the cache estimate of its probability will be 0. This will happen often with nouns, verbs, adjectives, and indeed with any POS category with many different members. On the other hand, for a POS category with few members, it is likely that all of them will appear in a collection of 200 successive words with that POS. Take the PP3A category as

an example. It is almost inconceivable that the word "he" could appear 200 times in a text without the word "she" appearing once. More investigation is required to determine if this explanation is correct. It is quite possible that content words are NOT more "bursty" than function words !

Since the cost of maintaining a 200-word cache, in terms of memory and time, is modest, and the increase in predictive power can be great, the approach outlined above should be considered as a simple way of improving on the performance of a 3g-gram language model for speech recognition. If memory is limited, one would be wise to create caches only for POSs that occur with high frequency and ignore other POSs, as we have done.

How could this research be extended ? One might explore the possibility of building a morphological component so that the occurrence of a word would increase the estimated probability of related words. Thus the occurrence of the singular form of a noun would raise the probability of its plural (and vice versa). Different tenses and persons of a verb could be related in the same way.

Another promising idea would be to extend the idea of a model that dynamically tracks the linguistic behaviour of the speaker or writer from the lexical to the syntactic component of the model. In other words, perhaps the recent past is a good guide to the POSs that will be employed, as well as to the words that will be uttered. Recently employed POSs would be assigned higher probabilities.

One might also consider combining the model described here with the trigram model. Word probabilities could be a weighted average of trigram and cache-based 3g-gram components; alternatively, one could use the latter only when a bigram not found (or rarely found) in the training text appeared.

Trigram purists who dislike the use of POSs in the 3g-gram model might prefer to construct a dynamic version of the pure trigram model, which would include a **cache-based trigram** component. In other words, the system would keep track of trigrams encountered during the recognition task. It is obvious that this model would, at a minimum, handle noun phrases better than any existing model - one has only to glance at a newspaper story to see the same noun phrases appearing again and again.

(I am grateful to Matthew Lennig, Andrew McGregor, and their colleagues at Bell Northern Research for discussing these possible extensions with me; some of them had already occurred to me, but some had not).

The line of research described in this thesis has more general implications. The results above seem to suggest that at a given time, a human being works with only a small fraction of his vocabulary. Perhaps if we followed an individual's written or spoken use of language through the course of a day, it would consist largely of time spent in language "islands" or sublanguages, with brief periods of time during which he is in transition between islands. One might attempt to chart these "islands" by identifying groups of words which often occur together in the language. If this work is ever carried out on a large scale, it could lead to pseudo-semantic language models for speech recognition, since the occurrence of several words characteristic of an "island" makes the appearance of **all** words in that island more probable.

VII. Bibliography

1. R. Campo, L. Fissore, A. Martelli, G. Micca, and G. Volpi, "Probabilistic Models of the Italian Language for Speech Recognition". *Recent Advances and Applications of Speech Recognition* (international workshop), pp. 49-56, Rome, May 1986.

2. R. De Mori, L. Lam, and M. Gilloux, "Learning and Plan Refinement in a Knowledge-Based System for Automatic Speech Recognition", *Technical Report No. SOCS 86.14*, McGill University, May 1986.

3. R. De Mori, P. Laface, and Y. Mong, "Parallel Algorithms for Syllable Recognition in Continuous Speech", *IEEE Trans. Pattern Anal. Machine Intell.*, Vol. PAMI-6, pp. 56-69, Jan. 1985.

4. A.M. Derouault and B. Mérialdo, "Natural Language Modeling for Phoneme-to-Text Transcription", *IEEE Trans. Pattern Anal. Machine Intell.*, Vol. PAMI-8, pp. 742-749, No. 1986.

5. A.M. Derouault and B. Mérialdo, "Language Modeling at the Syntactic Level", *7th Int. Conf. Pattern Recognition*, Vol. II, pp. 1373-1375, Montreal, Aug. 1984.

6. P. D'Orta, M. Ferretti, et al, "Large-vocabulary speech recognition : a System for the Italian Language", *IBM J. Res. Develop.*, Vol. 32, No. 2, pp. 217-226, Mar. 1988.

7. W.N. Francis, "A Tagged Corpus - Problems and Prospects", in *Studies in English Linguistics for Randolph Quirk*, S. Greenbaum, G. Leech, and J. Svartvik, Eds. London: Longman, 1980, pp. 193-209.

8. F. Jelinek, "The Development of an Experimental Discrete Dictation Recognizer", *Proc. IEEE*, Vol. 73, No.11, pp. 1616-1624, Nov. 1985.

9. F. Jelinek, R.L. Mercer, and L.R. Bahl, "A Maximum Likehood Approach to Continuous Speech Recognition", *IEEE Trans. Pattern Anal. Machine Intell.*, Vol. PAMI-5, pp. 179-90, Mar. 1983.

10. F. Jelinek, "Markov Source Modeling of Text Generation", personal communication.

11. F. Jelinek, "Self-Organized Language Modeling for Speech Recognition",

personal communication.

12. S. Johansson, E. Atwell, R. Garside, and G. Leech, *The Tagged LOB Corpus Users Manual*, Norwegian Computing Centre for the Humanities, Bergen, Norway, 1986.

13. S. Johansson, "Some Observations on Word Frequencies in Three Corpora of Present-Day English Texts", *ITL Review of Applied Linguistics*, Vol. 67-68, pp. 117-126, 1985.

14. S. Johansson, "Word Frequency and Text Type: Some Observations based on the LOB Corpus of British English Texts", *Computers and the Humanities*, Vol. 19, pp. 23-36, 1985.

15. S. Katz, "Recursive M-gram Modeling Via a Smoothing of Turing's Formula", *forthcoming paper.*

16. E.F. Kelly and P.J. Stone, *Computer Recognition of English Word Senses*: North-Holland Publishing Co., 1975.

17. D.E. Knuth, "An Analysis of Optimum Caching", *J. Algorithms*, Vol. 6, pp. 181-99, 1985.

18. R. Kuhn, "Speech Recognition and the Frequency of Recently Used Words", *Proc. COLING 88*, Budapest, Hungary, Vol. I, pp. 348-350, 1988.

19. S.E. Levinson, L.R. Rabiner, and M.M. Sondhi, "An Introduction to the Application of Probabilistic Functions of a Markov Process to Automatic Speech Recognition", *The Bell System Technical Journal*, Vol. 62, No. 4, pp. 1035-1074, Apr. 1983.

20. I. Marshall, "Choice of Grammatical Word-Class Without Global Syntactic Analysis: Tagging Words in the LOB Corpus", *Computers and the Humanities*, Vol. 17, No. 3, pp. 139-150, Sept. 1983.

21. A. Martelli, "Probability Estimation of Unseen Events for Language Modeling", personal communication.

22. H. Méloni, "Etude et Réalisation d'un Système de Reconnaissance Automatique de la Parole Continue", Ph.D. dissertation, University of Aix-Marseille II, Feb. 1982.

23. E.M. Muckstein, "A Natural Language Parser with Statistical Applications", *IBM Research Report* RC7516 (#38450), Mar. 1981.

24. A. Nadas, "Estimation of Probabilities in the Language Model of the IBM Speech Recognition System", *IEEE Trans. Acoust., Speech, Signal Processing*, Vol. 32, pp. 859-861, Aug. 1984.

25. J. Peterson and A. Silberschatz, *Operating System Concepts*. Addison-Wesley Co., 1983.

26. L.R. Rabiner and B.H. Juang, "An Introduction to Hidden Markov Models", *IEEE ASSP Magazine*, pp. 4-16, Jan. 1986.

27. C. Scagliola, "Probability Estimation of Unseen Events for Language Modeling", personal communication.