A Cache-Based Natural Language Model for Speech Recognition

ROLAND KUHN AND RENATO DE MORI

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 paper 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 "3g-gram component" of the traditional type. The combined model and a pure 3g-gram 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.

Index Terms—Cache-based language model, language models for speech recognition, Markov language models, natural language, perplexity, probabilistic language model, 3g-gram language model.

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 set of the most plausible word candidates together with a probability for each. The second component, which incorporates a language model, estimates for each word in the vocabulary the probability that it will occur, given a list of previously hypothesized words. 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. These models, whose parameters are calculated from a large training text, produce a reasonable nonzero 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 paper. In addition, we provide an overview of several probabilistic language models currently used in the field of speech recognition.

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 3g-gram 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 3g-gram and cache components in calculating word probabilities, where the relative weights assigned to each component depend on the part of speech (POS). The 153 POS's used are defined in Johansson *et al.* [6]. For purposes of comparison, we also created a pure *3g-gram model*, consisting of only the 3g-gram component of the *combined model*.

For each POS, the *combined model* may therefore place more reliance on the cache component than on the 3g-gram component, or vice versa; the relative weights were obtained experimentally for each POS from a training text, using the deleted interpolation method [5]. 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 five 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 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. We believed that if it was correct, the relative weight calculated from the training text for the cache component for most content POS's

Manuscript received October 10, 1988; revised November 7, 1989. This work was supported by the National Science and Engineering Research Council of Canada.

The authors are with the School of Computer Science, McGill University, Montreal, P.Q. H3A 2A7, Canada.

IEEE Log Number 9034826.

would be higher than the cache weighting for most function POS's.

Our research was greatly facilitated by the availability of a large and varied collection of modern texts, in which each word is labeled 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 [6]–[8]; it is available to academic researchers (but not, unfortunately, to their colleagues in industry) from the Norwegian Computing Centre for the Humanities. We chose to employ the same 153 POS's 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 POS's. Part of this corpus (391,658 words) was utilized as a training text which determined the parameters of both models: the standard *3g-gram model*, and our *combined model* consisting of the same 3g-gram 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, R. L. Mercer, and L. R. Bahl [4]. 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. In principle, low perplexity implies good performance, although it does not take into account the varying degrees of acoustic difficulty among words. It is possible that a language model that accurately distinguished between easily confused words (such as short words) might be more useful in practice than another model of slightly lower perplexity. On the other hand, perplexity provides a straightforward way of comparing language models independently of the other components of an ASR system.

Once the parameters of the two models, the pure *3g-gram* 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. The pure *3g-gram model*, as expected, had a high estimated perplexity: 332. The estimated perplexity of the *combined model*, on the other hand, was 107. This more than threefold improvement indicates that addition of a cache component to a 3g-gram 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. It is interesting to speculate whether the addition of a cache component to other probabilistic language models in the literature, such as Jelinek's trigram model, would also improve the performance of these models.

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 POS's than for the cache component of function POS's. 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 deleted interpolation method, they did vary considerably from POS to POS. But there was no consistent trend of high values for content POS's and lower ones for function POS's. If anything, the pattern was the reverse. This and other aspects of the results are discussed in the conclusion.

II. Markov Models for Natural Language

A. 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 $WS = \langle W_1, W_2, \cdots, W_n \rangle$ 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

$$P(\hat{WS}|A) = \max_{WS} P(WS|A), \tag{1}$$

i.e., the most likely word string given the evidence. Since by the Bayes formula we have

$$P(WS|A) = \frac{P(WS) \cdot P(A|WS)}{P(A)} \tag{2}$$

it follows that

$$P(\hat{WS}|A) = \max_{WS} \frac{P(WS)\,P(A|WS)}{P(A)}. \tag{3}$$

Since A is fixed, it follows that

$$\hat{WS} = \Big\{ WS \text{ such that } P(WS) \cdot P(A|WS) \text{ is a maximum} \Big\}. \tag{4}$$

In this paper, we will not discuss $P(A | WS)$, the probability of the actual acoustic input being observed if the string $WS = \langle W_1, \cdots, W_n \rangle$ is uttered. The calculation of $P(A | WS)$ is the responsibility of the acoustic modeling part 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

$$P(WS) = P(W_1) \cdot \prod_{i=2}^{n} P(W_i | \langle W_1, \cdots, W_{i-1} \rangle). \tag{5}$$

Thus, the probability that a word W_i is spoken depends on the past history of the dictation. As Jelinek *et al.*, from whom the above account is derived [3]-[5] point out, the probabilities $P(W_i | \langle W_1, \cdots, W_{i-1} \rangle)$ are in practice impossible to estimate, since each history $\langle W_1, \cdots, W_{i-1} \rangle$ 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 $P(W_i = W | \langle W_1, \cdots, W_{i-1} \rangle)$ 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 $\langle W_1, \cdots, W_{i-1} \rangle$ into a more manageable number of equivalence classes. Let us denote this many to one mapping by M. Thus, $M(\langle W_1, \cdots, W_{i-1} \rangle)$ denotes the equivalence class of the string $\langle W_1, \cdots, W_{i-1} \rangle$.

The probability $P(W_i = W)$ is approximated by

$$P(W_i = W) = P\big(W_i = W | M(\langle W_1, \cdots, W_{i-1} \rangle)\big). \tag{6}$$

Any language model for speech recognition to be used for maximum likelihood estimation will consist of such a mapping M of word strings into equivalence classes.

B. 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 system's model. 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.

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, Jelinek and his colleagues believe that parsers should act only as final filters for word strings chosen by means of a more appropriate language model [3]-[5].

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 M of word strings to equivalence classes that facilitate estimation of $P(W_i = W | M(\langle W_1, \cdots, W_{i-1} \rangle))$. This involves a compromise between the need for a refined classification that loses little relevant information about the history $\langle W_1, \cdots, W_{i-1} \rangle$ 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 it considers it 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 one of the tasks 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. In the next section, we will discuss our *combined model*, which tries to overcome this disadvantage by using information about the lexical preferences of the speaker gathered during the recognition task and extending hundreds of words into the past. The Markov component of this model is based on the 3g-gram model, which is an adaptation by Derouault and Merialdo of Jelinek's original trigram model.

C. The Trigram Model

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

$$M(\langle W_1, \cdots, W_{i-1} \rangle) = \langle W_{i-2}, W_{i-1} \rangle. \quad (7)$$

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

$$P(W_i = W | W_{i-2}, W_{i-1}) \simeq f(W | W_{i-2}, W_{i-1}) = \frac{N(W_{i-2}, W_{i-1}, W)}{N(W_{i-2}, W_{i-1})}. \quad (8)$$

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:

$$P(W_i = W | W_{i-2}, W_{i-1}) \simeq q_2 f(W_i = W | W_{i-2}, W_{i-1}) + q_1 f(W_i = W | W_{i-1}) + q_0 f(W_i = W), \quad (9)$$

where $q_0 + q_1 + q_2 = 1$ and

$$f(W_i = W | W_{i-2}, W_{i-1}) = N(W_{i-2}, W_{i-1}, W)/N(W_{i-2}, W_{i-1}),$$

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

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 nonzero probability, so it avoids the problem with the pure trigram model mentioned above. The values for q_0, q_1, and q_2 are chosen in order 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, $f(W_i = W | W_{i-2}, W_{i-1})$ 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. Katz's method "backs off" from a trigram to a bigram to a singlet estimate ([9]; described in [3]):

$$\begin{aligned} P(W_i = W | W_{i-2}, W_{i-1}) = \ & \textbf{if } N(W_{i-2}, W_{i-1}, W) > 0 \\ & \quad \textbf{then } r_2 f(W_i = W | W_{i-2}, W_{i-1}); \\ & \textbf{else if } N(W_{i-1}, W) > 0 \\ & \quad \textbf{then } r_1 f(W_i = W | W_{i-1}); \\ & \textbf{else } r_0 f(W_i = W). \end{aligned} \quad (10)$$

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.

D. The 3g-gram Model

The 3g-gram model (terminology of A. Martelli and of Derouault and Merialdo [1], [2]) 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 POS's 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 POS's 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 POS's and not words. Thus, the model gives a nonzero 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 POS's. Suppose that the model gave a probability of 0.99 to the occurrence of a noun 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 POS's recognized by our model, and let g_j be a particular POS whose probability of occurring we wish to predict. The model will give us an estimate $\hat{P}(g_i = g_j | g_{i-2}, g_{i-1})$ of the probability based on the identity of the two preceding POS's. 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:

$$\begin{aligned} P(W_i = W | g_{i-2}, g_{i-1}) &\simeq \hat{P}(W | g(W)) \cdot \hat{P}(g_i = g(W) | g_{i-2}, g_{i-1}) \\ &= f(W | g(W)) \cdot \hat{P}(g_i = g(W) | g_{i-2}, g_{i-1}) \end{aligned} \tag{11}$$

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:

$$\begin{aligned} P(W_i = W | \langle W_1, \cdots, W_{i-1} \rangle) &\simeq \sum_{g_j \in G} P(W | g_j) \cdot P(g_i = g_j | g_{i-2}, g_{i-1}) \\ &\simeq \sum_{g_j \in G} f(W | g_j) \cdot \hat{P}(g_i = g_j | g_{i-2}, g_{i-1}). \end{aligned} \tag{12}$$

Given a sufficiently large training text, $\hat{P}(g_i = g_j | g_{i-2}, g_{i-1})$ could be estimated for every POS g_j in G as $f(g_i = g_j | g_{i-2}, g_{i-1})$. 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 [1], [2] employed a variant of the weighted average 3g-gram approach. Their work will be described in some detail, as the 3g-gram 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 POS's. Only 5% 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 overall probability estimate of each word in order to prevent zero estimates for the probability of occurrence of any given word. Thus, they approximated the probability of occurrence of a word W at time i, given that W has part of speech g_j, the two preceding parts of speech are g_{i-2} and g_{i-1}, and vocabulary size is n, as

$$\begin{aligned} P(W_i = W | g(W) = g_j, g_{i-2}, g_{i-1})) &\simeq (1 - ne) f(W | g_j) \times [l_1 f(g_i = g_j | g_{i-2}, g_{i-1}) \\ &\quad + l_2 f(g_i = g_j | g_{i-1})] + e, \\ e = 10^{-4}, \; l_1 + l_2 = 1. \end{aligned} \tag{13}$$

They experimented with two different ways of calculating l_1 and l_2. Intuitively, it makes sense that if there are many triplets beginning $\langle g_{i-2}, g_{i-1}, \cdots \rangle$, the frequency $f(g_i = g_j | g_{i-2}, g_{i-1})$ 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 $\langle g_{i-2}, g_{i-1} \rangle$. Each possible history $\langle g_{i-2}, g_{i-1} \rangle$ 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 $\langle g_{i-2}, g_{i-1} \rangle$ 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 $h(\langle g_{i-2}, g_{i-1} \rangle)$ denote the parameter on which l_1 and l_2 depend. For Derouault and Merialdo's first approach, $h = N(\langle g_{i-2}, g_{i-1} \rangle)$ = the number of occurrences of $\langle g_{i-2}, g_{i-1} \rangle$ 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 deleted interpolation method [5]. Having split the training text into two portions in the ratio 3 : 1, they used the larger portion to calculate $f(g_i | g_{i-2}, g_{i-1})$ and $f(g_i | g_{i-1})$. 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 $\langle g_{i-2}, g_{i-1}, g_i \rangle$ in this portion, they defined

$$\begin{aligned} S_1(h) = \sum l_1(h) f(g_i | g_{i-2}, g_{i-1}) / [l_1(h) \\ \cdot f(g_i | g_{i-2}, g_{i-1}) + l_2(h) f(g_i | g_{i-1})], \end{aligned} \tag{14}$$

$$\begin{aligned} S_2(h) = \sum l_2(h) f(g_i | g_{i-1}) / [l_1(h) \\ \cdot f(g_i | g_{i-2}, g_{i-1}) + l_2(h) f(g_i | g_{i-1})]. \end{aligned} \tag{15}$$

They then redefined

$$l_1(h) = S_1(h)/(S_1(h) + S_2(h)),$$
$$l_2(h) = S_2(h)/(S_1(h) + S_2(h)). \quad (16)$$

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 $N(\langle g_{i-2}, g_{i-1} \rangle)$ 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 *3g-gram model*—the level that predicts the POS—works in almost 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 *3g-gram 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 *3g-gram model* and Derouault and Merialdo's model is the choice of POS's; they define 92 POS's, we use the 153 POS's in the LOB Corpus. In the next chapter, we will see how the *combined model* differs from *3g-gram model* at the level of lexical prediction.

E. Perplexity: A Measure of the Performance of a Language Model

We can view a language as a source of information 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 given by

$$LP = -(1/n)[\log_2 \hat{P}(w_1, w_2, \cdots, w_n)]. \quad (17)$$

Jelinek *et al.* [4] suggest that it is intuitively more satisfying to measure the difficulty of the speech recognition task by the value of the *perplexity* given by

$$PP = 2^{LP} = \hat{P}(w_1, \cdots, w_n)^{-1/n}. \quad (18)$$

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. 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

A. Argument for the Cache Component

The central idea underlying the work presented in this paper 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 is their inability to reflect short-term patterns in 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 would then assign overwhelmingly higher probability to the word "band." A word used in the immediate past—say the last 2000 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 predict.

There is strong empirical evidence for this. Studies on three corpora of English and American texts [7], [8] by S. Johansson show that "word frequencies vary greatly depending upon the type of text, both among content words and function words" [8, p. 34]. 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 [12]. 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 neighbors, is copied to the cache" [12, p. 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 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 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 as—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 Section II-D, the relative weights assigned to the cache and 3g-gram components within each POS category were determined experimentally by means of the deleted interpolation 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 the POS category in the training text—the 3g-gram 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.

B. *Mathematical Treatment of the Combined Model*

The *combined model* is now introduced mathematically. Recall that the pure *3g-gram model* is

$$P(W_i = W | g_{i-2}, g_{i-1}) = \sum_{g_j \in G} P(W_i = W | g_i = g_j)\, P(g_i = g_j | g_{i-2}, g_{i-1}). \tag{19}$$

The *combined model* leaves the POS component $P(g_i = g_j | g_{i-2}, g_{i-1})$ of the 3g-gram model unchanged. Our modification affects only the component that predicts the probability of a word given the POS, $P(W_i = W | g_i = g_j)$. In the *3g-gram model* this is estimated by $f(W_i = W | g_i = g_j)$, calculated from the training text. This is certainly a good estimate of the mean around which the value $P(W_i = W | g_i = g_j)$ 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 $P(W_i = W | g_i = g_j)$ by

$$P(W_i = W | g_i = g_j) \simeq k_{M,j} \times f(W_i = W | g_i = g_j) + k_{C,j} \times C_j(W, i), \tag{20}$$

where

$$k_{M,j} + k_{C,j} = 1,$$

instead of by $f(W_i = W | g_i = g_j)$ alone. This should allow the estimate of $P(W_i = W | g_i = g_j)$ to deviate from its average value to reflect temporary high or low values. The relative weights of $k_{M,j}$ and $k_{C,j}$ are found by the deleted interpolation method mentioned in Section II-D; the values thus obtained maximize the probability of the training text. Note that the *3g-gram 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.

We must also specify how we estimated the POS component $P(g_i = g_j | g_{i-2}, g_{i-1})$ of both the 3g-gram and the combined models. This was done in almost the same way as was done by Derouault and Merialdo, as described in Section II-D. 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 g_{i-1}. To ensure that no POS g_j is ever assigned a probability of zero, we added an arbitrary small number 0.0001. We thus made the approximation

$$P(g_i = g_j | g_{i-2}, g_{i-1}) \simeq l_1(g_{i-1}) f(g_i = g_j | g_{i-2}, g_{i-1}) + l_2(g_{i-1}) f(g_i = g_j | g_{i-1}) + 0.0001, \tag{21}$$

where $l_1(g_x) + l_2(g_x) = 0.9847$ for all x (where $0.9847 = 1 - (\text{no. of POS's}) \times 0.0001$).

The above description ignores the possibility that a word will be encountered in the sample text that is not in the system's vocabulary V. There are different ways of calculating the probability that such a word will occur at time i [9]; we estimated 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 391,658 words in the training text, so the probability of encountering a word not in the vocabulary was estimated as about 0.035. All such unknown words are treated as if they were a single word whose probability of occurrence is always 0.035 no matter what the preceding POS's were. They are discarded instead of being placed in a cache. As is described in Section IV-D, the system guesses the POS g_i of the discarded word on the basis of the two preceding POS's by assigning it the value of g_j that maximizes the value of the previous equation.

We can now give the overall formula that we used:

$$P(W_i = W | g_{i-2}, g_{i-1}) =$$

if W in V,

$$(1 - d) \sum_{g_j \in G} [[k_{M.j} \times f(W_i = W | g_i = g_j) + k_{C.j} \times C_j(W, i)] \times [l_1 f(g_i = g_j | g_{i-2}, g_{i-1}) + l_2 f(g_i = g_j | g_{i-1}) + 0.0001]],$$

else d,

$$\text{where } d = 0.035,\ k_{M.j} + k_{C.j} = 1,\ l_1 + l_2 = 0.9847. \tag{22}$$

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 POS's 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 POS's met these two criteria; however, these 19 together make up roughly 65% of the LOB Corpus. They are introduced in Section IV. Thus, for POS's other than these 19, there is no cache component in the *combined model*; the estimated probability is identical to that of the pure *3g-gram model*.

Another problem was 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 $k_{C,j}$ 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 five 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.

In order to test our hypothesis that each POS should be given a different best-fit pair of weights for its cache and 3g-gram components, we experimented briefly with a model in which all POS's had the same pair of weights. Recall that the two weights must add up to 1.0. We experimented with $(k_C, k_M) = (0.0, 1.0); (0.1, 0.9); (0.2, 0.8); \cdots ; (0.9, 0.1)$. We did not try to find a best-fit pair of relative weights for this simpler version of the *combined model*.

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 so dramatically 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 $P(W_i = W | \langle W_1, \cdots, W_{i-1} \rangle)$ as follows:

$$\begin{aligned} P(W_i = W | \langle W_1, \cdots, W_{i-1} \rangle) &\simeq a_1 \times \hat{P}(W_i = W | W_{i-2}, W_{i-1}) \\ &+ a_2 \times \left[\sum_{g_j \in G} \hat{P}(g_j | g_{i-2}, g_{i-1}) \times [k_{M,j} f(W | g_j) + k_{C,j} C_j(W, i)] \right], \\ a_1 + a_2 = 1,\ k_{M,j} + k_{C,j} = 1. \end{aligned} \tag{23}$$

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

A. 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 POS's. 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 [6]–[8]. These categories are shown in Table I.

The table shows the 15 text categories. The column labeled "Corpus" gives the number of samples in each category in the original LOB Corpus. We extracted three different, nonoverlapping 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 column labeled "Training Text" shows the number of samples in each category for the first of these collections; the last column applies to both remaining collections.

The training text for our models was used to obtain counts for triplets, doublets, and singlets of POS's. 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, for a total of 391,658 words.

The second collection of samples was used for further parameter setting, including calculation of the l-values in the Derouault–Merialdo formula (Section II-D), 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

TABLE I
DISTRIBUTION OF LOB CATEGORIES

Symbol	Description	Corpus	Training Text	Para Setting & Testing Texts
A	Press reportage	44	15	9
B	Editorials	27	9	5
C	Press reviews	17	6	3
D	Religion	17	6	3
E	Skills and hobbies	38	13	8
F	Popular lore	44	15	9
G	Biography and essays	77	25	15
H	Miscellaneous	30	10	6
J	Learned writings	80	27	16
K	General fiction	29	10	6
L	Mystery fiction	24	8	5
M	Science fiction	6	2	1
N	Adventures and westerns	29	10	6
P	Love stories	29	10	6
R	Humor	9	3	2

component and the 3g-gram component in the combined model. It contained 100 samples altogether.

The third collection of samples formed the testing text. It was used to compare the combined model with the 3g-gram 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 labeled 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.

B. Parameter Calculation

All parameters for both the 3g-gram 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 POS's 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 interpolated deletion as described in Section II-D 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 interpolated deletion from the parameter setting text in exactly the same way.

C. Implementing the Combined Model

Because of memory limitations, it proved impossible to implement a cache for every one of the 153 POS's in the LOB Corpus. As was mentioned in Section III-B, two criteria were used to select the POS's which would be assigned a cache:

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

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

The second criterion is obvious—if only one vocabulary

TABLE II
PARTS OF SPEECH FOR WHICH CACHES ARE DEFINED

Cache Number	POS Name	Description
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

item has a given POS, the cache component yields no extra information. The first criterion is based on the premise that rare POS's will be more spread out in time, so that the predictive power of the cache component will be weakened.

The 19 POS's that survived this selection process are listed in Table II.

D. Testing the Combined Model

As described in Section IV-B, two parts of the LOB Corpus were used to find the best-fit parameters for the pure *3g-gram model* and the *combined model*, made up of the *3g-gram 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. However, it is more convenient to calculate the logarithm of this estimated overall probability, equal to the sum of the logs of the individual word estimated probabilities:

$$LO = \log_2 \hat{P}(w_1, w_2, \cdots, w_n) = \log_2 \hat{P}(w_1) + \log_2 \hat{P}(w_2) + \cdots + \log_2 \hat{P}(w_n).$$

The perplexity is then calculated as

$$PP = 2^{-LO/n},$$

where n is the number of words (230,598 in this case).

Recall that we also tested a simpler version of the *combined model*, in which the cache component has the same weight for all POS's. The weights tried were 0.0, 0.1, 0.2, $\cdots$, 0.9; the 3g-gram 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 POS's of the two preceding words. Thus every word encountered must be assigned a POS. There are three cases:

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

2) the word was in the training text, and had the same tag whenever it occurred;

3) 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:

1) in this case, the two previous POS's 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.

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

3) 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 POS's. A separate part of the program compared actual tags with guessed ones in order to collect statistics on the performance of these heuristics.

As one of the referees of this paper pointed out, the assignment of a tag to an unknown word would benefit from the use of information about one or more of the words that succeed it. In fact, an even simpler improvement could have been made. We did not take into account the fact that rare words are more likely to occur in some

POS classes than in others. For instance, a word we do not recognize is unlikely to be a connective or a preposition. Recall that we counted the unique words in the training text; it would have been possible to record their distribution among POS categories, and modify heuristic 1) above so as to take this information into account. Either approach would probably have reduced the number of misassigned unknown words.

V. Results

A. 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 $f(g_i | g_{i-2}, g_{i-1})$ and the POS doublet frequency $f(g_i | g_{i-1})$ in the estimation of $P(g_i = g_j | g_{i-2}, g_{i-1})$. 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 [1] 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.

B. 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 3g-gram 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 optimal weights, calculated by means of the deleted interpolation method in Table III, decisively refute this hypothesis.

The pattern in Table III is just the opposite of what we had expected, with function POS's having significantly higher optimal weights for the cache component of the *combined model* than content POS's. This intriguing result is discussed in the conclusion.

TABLE III
Optimal Weights by POS

POS	Description	Cache Component	3g-gram Component
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 noun	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

C. Performance of Both Models on the Testing Text

We calculated the performance of the various models on the testing text of 100 samples from the LOB Corpus (230,598 words); the most important results will be given first. The pure 3g-gram model gives perplexity equal to 332 (average probability per word is 0.003008). On the other hand, the combined model gives perplexity equal to 107 (average probability per word is 0.009341). This dramatic, more than threefold, 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 are shown in Table IV.

Thus the lowest perplexity, 118, was obtained when the cache component weight was 0.7 and 3g-gram component weight was 0.3. It is difficult to be sure without using deleted interpolation 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 3g-gram 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 recognized). Of the remaining 216,162 words that had occurred at least once in the training text, 202,882

TABLE IV
RESULTS FOR MODEL WITH EQUALLY WEIGHTED CACHES

Cache Weight	3g-gram Weight	Perplexity	Average Probability
0.1	0.9	180	0.005551
0.2	0.8	152	0.006570
0.3	0.7	137	0.007277
0.4	0.6	128	0.007790
0.5	0.5	122	0.008150
0.6	0.4	119	0.008363
0.7	0.3	118	0.008411
0.8	0.2	121	0.008232
0.9	0.1	131	0.007620

(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 POS's 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 seem to strongly confirm our hypothesis that recently used words have a higher probability of occurrence than the 3g-gram model would predict. When a *3g-gram 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 3. How representative are our results?

We suspect that if we had employed a training text from one source (for instance, business correspondence) and a testing text drawn from another source (for instance, sports journalism) the advantages of including a cache component in the language model would have been even more apparent, since the 3g-gram component would not be as good a predictor in this case. By the same logic, if both the training text and the testing text came from the same narrowly defined area of discourse, the cache component would probably not bring about such a dramatic improvement as was observed by us, because the 3g-gram component would be a better predictor than it was with the highly diverse training and testing texts we employed. Similarly, an increase in the size of the training text would make the 3g-gram component of the model more reliable, and hence decrease the relative amount of improvement in perplexity contributed by the cache component.

Thus, the importance of our results is in the trend they show, not in the precise values we obtained; these depend on the size and origin of both the training text and the testing text. Nevertheless, the sheer magnitude of the improvement in perplexity over the pure 3g-gram model we achieved by incorporating a cache component—a factor of 3—indicates that we are not being misled by some random fluctuation in word frequencies. A time locality effect exists, and it is important.

Would the performance of the trigram model also be improved if a cache component were incorporated with it? Strictly speaking, the data presented here do not answer the question one way or the other. It is probably true that the two words preceding a given word often indicate the nature of the topic being discussed or the verbal habits of the speaker, so that the trigram model would perform better than the 3g-gram model on a diverse testing text such as the one we used. On the other hand, we think that a great deal of information about the lexical choices likely to be made by the speaker depends on what happened hundreds of words ago and will thus be lost by the trigram model, yet is easily obtained by means of a cache component. The only way to find out is to test a pure trigram model against a trigram plus cache model—we hope that those of our readers with sufficient resources to try the experiment will do so!

It came as a surprise to us that, in general, the best-fit weight of the cache component for function POS's was higher than the best-fit weight for the cache component for content POS's. We had naively assumed that the best-fit weight for the cache component would reflect the "burstiness" of the POS in question, and hence be larger for content POS's. Actually, another factor seems to be more powerful in determining the weighting for the cache component of a given POS: the less diverse a category is, the larger its best-fit cache component weight. For instance, the POS category PP3A contains only the words "he" and "she"; the corresponding cache of 200 words, containing these two words in different proportions, will contain a precise estimate of their probability distribution within the category. In this case, the best-fit cache weight was 1.000. On the other hand, consider a POS category like *NN*, containing singular nouns. By definition, all the singular nouns that were not among the last 200 singular nouns will be assigned a probability of zero, which is unrealistic; furthermore, even probability estimates for nouns within the cache are based on a very small sample. Thus diverse categories like this one need the information in the 3g-gram component to smooth out the probability estimates (for *NN*, the best-fit cache component weight is only 0.403).

If there is a moral to this, it is a point often made by Jelinek: whenever possible, do not make intuitive judgments about the parameters of your language model but train them from actual data. We might easily have decided in advance that building cache components for function POS's was pointless, thus impairing the performance of the *combined model*.

An interesting point is that the way we implemented our model was actually rather unfavorable to the cache component for content POS's. Recall that for training and testing purposes, we created two "texts" which were concatenations of 2000-word actual texts taken from the LOB Corpus. As there are 19 caches containing 200 words each, the caches at any time during training and testing (except the beginning) contain many words from previous actual texts. Even though actual texts drawn from the same general category were placed together, content words may differ so much in frequency from actual text to actual text that this implementation reduces the efficiency of the cache component. Perhaps all caches should be emptied at the end of each actual text, even though this would mean that there are often less than 200 words in each cache. In this case, we almost certainly would want the weighting of the cache component to depend on the number of words in the cache, in a more sophisticated way than our current step-function heuristic (which sets the cache contribution to 0 until the cache has 5 words in it, then leaves it constant).

Several other ideas for improvements have occurred to us. 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 behavior of the speaker or writer from the lexical to the syntactic component of the model. In other words, perhaps the relatively recent past (before the preceding two POS's) is a good guide to the POS's that will be employed, as well as to the words that will be uttered. Recently employed POS's would be assigned higher probabilities.

One might also consider combining the model considered here with the trigram model. There are several different ways of doing this, one of which was presented at the end of Section III (a weighted average of the trigram model and our combined model). Alternatively, one could use our combined model only when a bigram is not found—or rarely found—in the training text. Trigram purists who dislike the use of POS's might prefer to construct a more dynamic version of the original trigram model, in which trigrams and bigrams encountered during the recognition task would be assigned higher probabilities than if they only occurred in the training text. It seems 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 (often very idiosyncratic) noun phrases appearing again and again.

The line of research described in this paper has more general implications. The results above 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 pseudosemantic 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.

Acknowledgment

We would like to thank the two anonymous referees, whose comments helped to make this a better paper.

References

[1] 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, Nov. 1986.
[2] ——, "Language modeling at the syntactic level," in *Proc. 7th Int. Conf. Pattern Recognition*, vol. II, Montreal, Aug. 1984, pp. 1373–1375.
[3] F. Jelinek, "The development of an experimental discrete dictation recognizer," *Proc. IEEE*, vol. 73, no. 11, pp. 1616–1624, Nov. 1985.
[4] 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.
[5] F. Jelinek and R. L. Mercer, "Interpolated estimation of Markov source parameters from sparse data," in *Pattern Recognition in Practice*, E. S. Gelsema and L. H. Kanal, Eds. 1981, pp. 381–397.
[6] S. Johansson, E. Atwell, R. Garside, and G. Leech, *The Tagged LOB Corpus User's Manual*, Norwegian Computing Centre for the Humanities, Bergen, Norway, 1986.
[7] S. Johansson, "Some observations on word frequencies in three corpora of present-day English texts," *ITL Rev. Appl. Linguistics*, vol. 67–68, pp. 117–126, 1985.
[8] ——, "Word frequency and text type: Some observations based on the LOB corpus of British English texts," *Comput. Humanities*, vol. 19, pp. 23–36, 1985.
[9] S. Katz, "Recursive *M*-gram modeling via a smoothing of Turing's formula," forthcoming paper.
[10] E. M. Muckstein, "A natural language parser with statistical applications," IBM Res. Rep. RC7516 (38450), Mar. 1981.
[11] 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.
[12] J. Peterson and A. Silberschatz, *Operating System Concepts.* Reading, MA: Addison-Wesley, 1983.
[13] L. R. Rabiner and B. H. Juang, "An introduction to hidden Markov models," *IEEE ASSP Mag.*, pp. 4–16, Jan. 1986.

Roland Kuhn received the B.Sc. (Honours) degree in mathematics and biology from Trinity College, University of Toronto, Toronto, Ont., Canada, the S.M. degree in theoretical biology from the University of Chicago, Chicago, IL, and the M.Sc. degree in computer science from McGill University, Montreal, P.Q., Canada.

He is now a doctoral candidate in the McGill University School of Computer Science. His research interests include natural language, speech recognition, and robotics.

Renato De Mori was born in Milan, Italy, in 1941. He received the Doctorate degree in electronic engineering from Politecnico di Torino, Torino, Italy, in 1967.

Since January 1986, he has been Professor and the Director of the School of Computer Science at McGill University, Montreal, P.Q., Canada. Since March 1987 he has been Vice President and Director of research at the Centre de Recherche en Informatique de Montreal (CRIM), a research center involving seven universities and more than twenty companies. He has been a member of various committees in the U.S., Europe, and Canada. He is the author of 3 books and over 100 papers, published mostly in international journals and in the proceedings of international conferences devoted to computer systems, pattern recognition, and artificial intelligence.

Dr. De Mori is Associate Editor of the IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE and of the journals *Computer Speech and Language* (New York: Academic), *Signal Processing*, *Speech Communication*, and *Pattern Recognition Letters* (Amsterdam, The Netherlands: North-Holland). He is the Vice President of the Canadian Society for Computational Studies of Intelligence, Chairman of the Artificial Intelligence Associate Committee of the National Research Council of Canada, member of the Scientific Council of the Centre National d'Etudes de Telecommunications (CNET), Lannion-A, France, and of the Information Technology Research Centre of the Province of Ontario.