Class-Based n-gram Models of Natural Language

Peter F. Brown*
Peter V. deSouza*
Robert L. Mercer*
IBM T. J. Watson Research Center

Vincent J. Della Pietra*
Jenifer C. Lai*

We address the problem of predicting a word from previous words in a sample of text. In particular, we discuss n-gram models based on classes of words. We also discuss several statistical algorithms for assigning words to classes based on the frequency of their co-occurrence with other words. We find that we are able to extract classes that have the flavor of either syntactically based groupings or semantically based groupings, depending on the nature of the underlying statistics.

1. Introduction

In a number of natural language processing tasks, we face the problem of recovering a string of English words after it has been garbled by passage through a noisy channel. To tackle this problem successfully, we must be able to estimate the probability with which any particular string of English words will be presented as input to the noisy channel. In this paper, we discuss a method for making such estimates. We also discuss the related topic of assigning words to classes according to statistical behavior in a large body of text.

In the next section, we review the concept of a language model and give a definition of n-gram models. In Section 3, we look at the subset of n-gram models in which the words are divided into classes. We show that for $n = 2$ the maximum likelihood assignment of words to classes is equivalent to the assignment for which the average mutual information of adjacent classes is greatest. Finding an optimal assignment of words to classes is computationally hard, but we describe two algorithms for finding a suboptimal assignment. In Section 4, we apply mutual information to two other forms of word clustering. First, we use it to find pairs of words that function together as a single lexical entity. Then, by examining the probability that two words will appear within a reasonable distance of one another, we use it to find classes that have some loose semantic coherence.

In describing our work, we draw freely on terminology and notation from the mathematical theory of communication. The reader who is unfamiliar with this field or who has allowed his or her facility with some of its concepts to fall into disrepair may profit from a brief perusal of Feller (1950) and Gallagher (1968). In the first of these, the reader should focus on conditional probabilities and on Markov chains; in the second, on entropy and mutual information.

* IBM T. J. Watson Research Center, Yorktown Heights, New York 10598.

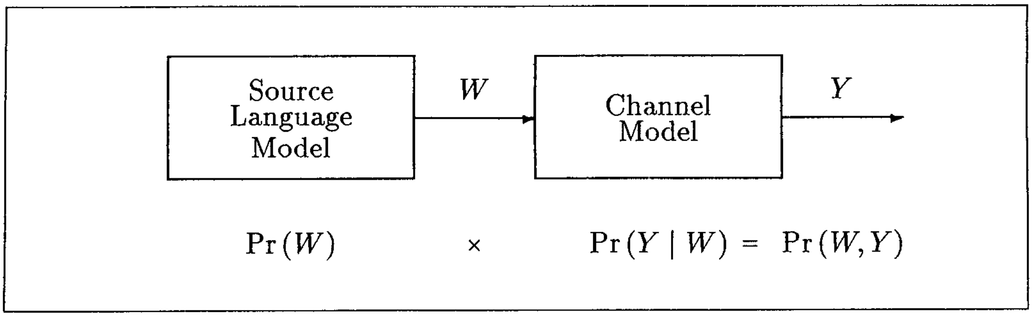

Figure 1
Source-channel setup.

2. Language Models

Figure 1 shows a model that has long been used in automatic speech recognition (Bahl, Jelinek, and Mercer 1983) and has recently been proposed for machine translation (Brown et al. 1990) and for automatic spelling correction (Mays, Demerau, and Mercer 1990). In automatic speech recognition, **y** is an acoustic signal; in machine translation, **y** is a sequence of words in another language; and in spelling correction, **y** is a sequence of characters produced by a possibly imperfect typist. In all three applications, given a signal **y**, we seek to determine the string of English words, **w**, which gave rise to it. In general, many different word strings can give rise to the same signal and so we cannot hope to recover **w** successfully in all cases. We can, however, minimize our probability of error by choosing as our estimate of **w** that string $\hat{\mathbf{w}}$ for which the a posteriori probability of $\hat{\mathbf{w}}$ given **y** is greatest. For a fixed choice of **y**, this probability is proportional to the joint probability of $\hat{\mathbf{w}}$ and **y** which, as shown in Figure 1, is the product of two terms: the a priori probability of $\hat{\mathbf{w}}$ and the probability that **y** will appear at the output of the channel when $\hat{\mathbf{w}}$ is placed at the input. The a priori probability of $\hat{\mathbf{w}}$, $\Pr(\hat{\mathbf{w}})$, is the probability that the string $\hat{\mathbf{w}}$ will arise in English. We do not attempt a formal definition of English or of the concept of arising in English. Rather, we blithely assume that the production of English text can be characterized by a set of conditional probabilities, $\Pr(w_k \mid w_1^{k-1})$, in terms of which the probability of a string of words, w_1^k, can be expressed as a product:

$$\Pr\left(w_1^k\right) = \Pr(w_1)\Pr\left(w_2 \mid w_1\right) \cdots \Pr\left(w_k \mid w_1^{k-1}\right). \tag{1}$$

Here, w_1^{k-1} represents the string $w_1 w_2 \cdots w_{k-1}$. In the conditional probability $\Pr(w_k \mid w_1^{k-1})$, we call w_1^{k-1} the history and w_k the prediction. We refer to a computational mechanism for obtaining these conditional probabilities as a language model.

Often we must choose which of two different language models is the better one. The performance of a language model in a complete system depends on a delicate interplay between the language model and other components of the system. One language model may surpass another as part of a speech recognition system but perform less well in a translation system. However, because it is expensive to evaluate a language model in the context of a complete system, we are led to seek an intrinsic measure of the quality of a language model. We might, for example, use each language model to compute the joint probability of some collection of strings and judge as better the language model that yields the greater probability. The *perplexity* of a language model with respect to a sample of text, S, is the reciprocal of the geometric average of the probabilities of the predictions in S. If S has $\mid S \mid$ words, then the perplexity is $\Pr(S)^{-1/|S|}$. Thus, the language model with the smaller perplexity will be the one that assigns the larger probability to S. Because the perplexity depends not only on the language model but also on the text with respect to which it is measured, it is important that the text be representative of that for which the language model is intended. Because perplexity is subject to sampling error, making fine distinctions between language models may require that the perplexity be measured with respect to a large sample.

In an *n-gram language model,* we treat two histories as equivalent if they end in the same $n - 1$ words, i.e., we assume that for $k \geq n$, $\Pr(w_k \mid w_1{}^{k-1})$ is equal to $\Pr(w_k \mid w_{k-n+1}^{k-1})$. For a vocabulary of size V, a 1-gram model has $V - 1$ independent parameters, one for each word minus one for the constraint that all of the probabilities add up to 1. A 2-gram model has $V(V - 1)$ independent parameters of the form $\Pr(w_2 \mid w_1)$ and $V - 1$ of the form $\Pr(w)$ for a total of $V^2 - 1$ independent parameters. In general, an n-gram model has $V^n - 1$ independent parameters: $V^{n-1}(V - 1)$ of the form $\Pr(w_n \mid w_1^{n-1})$, which we call the *order-n* parameters, plus the $V^{n-1} - 1$ parameters of an $(n - 1)$-gram model.

We estimate the parameters of an n-gram model by examining a sample of text, t_1^T, which we call the *training text*, in a process called *training*. If $C(\mathbf{w})$ is the number of times that the string $\mathbf{w}$ occurs in the string t_1^T, then for a 1-gram language model the maximum likelihood estimate for the parameter $\Pr(w)$ is $C(w)/T$. To estimate the parameters of an n-gram model, we estimate the parameters of the $(n-1)$-gram model that it contains and then choose the order-n parameters so as to maximize $\Pr(t_n^T \mid t_1^{n-1})$. Thus, the order-$n$ parameters are

$$\Pr(w_n \mid w_1^{n-1}) = \frac{C(w_1^{n-1} w_n)}{\sum_w C(w_1^{n-1} w)}. \tag{2}$$

We call this method of parameter estimation *sequential maximum likelihood estimation.*

We can think of the order-n parameters of an n-gram model as constituting the transition matrix of a Markov model the states of which are sequences of $n - 1$ words. Thus, the probability of a transition between the state $w_1 w_2 \cdots w_{n-1}$ and the state $w_2 w_3 \cdots w_n$ is $\Pr(w_n \mid w_1 w_2 \cdots w_{n-1})$. The steady-state distribution for this transition matrix assigns a probability to each $(n - 1)$-gram, which we denote $S(w_1^{n-1})$. We say that an n-gram language model is *consistent* if, for each string w_1^{n-1}, the probability that the model assigns to w_1^{n-1} is $S(w_1^{n-1})$. Sequential maximum likelihood estimation does not, in general, lead to a consistent model, although for large values of T, the model will be very nearly consistent. Maximum likelihood estimation of the parameters of a consistent n-gram language model is an interesting topic, but is beyond the scope of this paper.

The vocabulary of English is very large and so, even for small values of n, the number of parameters in an n-gram model is enormous. The IBM Tangora speech recognition system has a vocabulary of about 20,000 words and employs a 3-gram language model with over eight trillion parameters (Averbuch et al. 1987). We can illustrate the problems attendant to parameter estimation for a 3-gram language model with the data in Table 1. Here, we show the number of 1-, 2-, and 3-grams appearing with various frequencies in a sample of 365,893,263 words of English text from a variety of sources. The vocabulary consists of the 260,740 different words plus a special

Count	1-grams	2-grams	3-grams
1	36,789	8,045,024	53,737,350
2	20,269	2,065,469	9,229,958
3	13,123	970,434	3,653,791
> 3	135,335	3,413,290	8,728,789
> 0	205,516	14,494,217	75,349,888
≥ 0	260,741	6.799×10^{10}	1.773×10^{16}

Table 1
Number of n-grams with various frequencies in 365,893,263 words of running text.

unknown word into which all other words are mapped. Of the 6.799×10^{10} 2-grams that might have occurred in the data, only 14,494,217 actually did occur and of these, 8,045,024 occurred only once each. Similarly, of the 1.773×10^{16} 3-grams that might have occurred, only 75,349,888 actually did occur and of these, 53,737,350 occurred only once each. From these data and Turing's formula (Good 1953), we can expect that maximum likelihood estimates will be 0 for 14.7 percent of the 3-grams and for 2.2 percent of the 2-grams in a new sample of English text. We can be confident that any 3-gram that does not appear in our sample is, in fact, rare, but there are so many of them that their aggregate probability is substantial.

As n increases, the accuracy of an n-gram model increases, but the reliability of our parameter estimates, drawn as they must be from a limited training text, decreases. Jelinek and Mercer (1980) describe a technique called *interpolated estimation* that combines the estimates of several language models so as to use the estimates of the more accurate models where they are reliable and, where they are unreliable, to fall back on the more reliable estimates of less accurate models. If $Pr^{(j)}(w_i \mid w_1^{i-1})$ is the conditional probability as determined by the jth language model, then the interpolated estimate, $\hat{\Pr}(w_i \mid w_1^{i-1})$, is given by

$$\hat{\Pr}\left(w_i \mid w_1^{i-1}\right) = \sum_j \lambda_j(w_1^{i-1}) Pr^{(j)}(w_i \mid w_1^{i-1}). \tag{3}$$

Given values for $Pr^{(j)}(\cdot)$, the $\lambda_j(w_1^{i-1})$ are chosen, with the help of the EM algorithm, so as to maximize the probability of some additional sample of text called the *held-out data* (Baum 1972; Dempster, Laird, and Rubin 1977; Jelinek and Mercer 1980). When we use interpolated estimation to combine the estimates from 1-, 2-, and 3-gram models, we choose the λs to depend on the history, w_1^{i-1}, only through the count of the 2-gram, $w_{i-2}w_{i-1}$. We expect that where the count of the 2-gram is high, the 3-gram estimates will be reliable, and, where the count is low, the estimates will be unreliable. We have constructed an interpolated 3-gram model in which we have divided the λs into 1,782 different sets according to the 2-gram counts. We estimated these λs from a held-out sample of 4,630,934 words. We measure the performance of our model on the Brown corpus, which contains a variety of English text and is not included in either our training or held-out data (Kučera and Francis 1967). The Brown corpus contains 1,014,312 words and has a perplexity of 244 with respect to our interpolated model.

3. Word Classes

Clearly, some words are similar to other words in their meaning and syntactic function. We would not be surprised to learn that the probability distribution of words in the vicinity of *Thursday* is very much like that for words in the vicinity of *Friday*. Of course, they will not be identical: we rarely hear someone say *Thank God it's Thursday!* or worry about Thursday the 13th. If we can successfully assign words to classes, it may be possible to make more reasonable predictions for histories that we have not previously seen by assuming that they are similar to other histories that we have seen.

Suppose that we partition a vocabulary of V words into C classes using a function, π, which maps a word, w_i, into its class, c_i. We say that a language model is an *n-gram class model* if it is an n-gram language model and if, in addition, for $1 \leq k \leq n$, $\Pr\left(w_k \mid w_1^{k-1}\right) = \Pr\left(w_k \mid c_k\right) \Pr\left(c_k \mid c_1^{k-1}\right)$. An n-gram class model has $C^n - 1 + V - C$ independent parameters: $V - C$ of the form $\Pr\left(w_i \mid c_i\right)$, plus the $C^n - 1$ independent parameters of an n-gram language model for a vocabulary of size C. Thus, except in the trivial cases in which $C = V$ or $n = 1$, an n-gram class language model always has fewer independent parameters than a general n-gram language model.

Given training text, t_1^T, the maximum likelihood estimates of the parameters of a 1-gram class model are

$$\Pr\left(w \mid c\right) = \frac{C(w)}{C(c)}, \tag{4}$$

and

$$\Pr\left(c\right) = \frac{C(c)}{T}, \tag{5}$$

where by $C(c)$ we mean the number of words in t_1^T for which the class is c. From these equations, we see that, since $c = \pi(w)$, $\Pr(w) = \Pr\left(w \mid c\right) \Pr(c) = C(w)/T$. For a 1-gram class model, the choice of the mapping π has no effect. For a 2-gram class model, the sequential maximum likelihood estimates of the order-2 parameters maximize $\Pr\left(t_2^T \mid t_1\right)$ or, equivalently, $\log \Pr(t_2^T \mid t_1)$ and are given by

$$\Pr\left(c_2 \mid c_1\right) = \frac{C(c_1c_2)}{\sum_c C(c_1c)}. \tag{6}$$

By definition, $\Pr(c_1c_2) = \Pr(c_1) \Pr\left(c_2 \mid c_1\right)$, and so, for sequential maximum likelihood estimation, we have

$$\Pr(c_1c_2) = \frac{C(c_1c_2)}{T} \times \frac{C(c_1)}{\sum_c C(c_1c)}. \tag{7}$$

Since $C(c_1)$ and $\sum_c C(c_1c)$ are the numbers of words for which the class is c_1 in the strings t_1^T and t_1^{T-1} respectively, the final term in this equation tends to 1 as T tends to infinity. Thus, $\Pr(c_1c_2)$ tends to the relative frequency of c_1c_2 as consecutive classes in the training text.

Let $L(\pi) = (T-1)^{-1} \log \Pr\left(t_2^T \mid t_1\right)$. Then

$$\begin{aligned} L(\pi) &= \sum_{w_1w_2} \frac{C(w_1w_2)}{T-1} \log \Pr\left(c_2 \mid c_1\right) \Pr\left(w_2 \mid c_2\right) \\ &= \sum_{c_1c_2} \frac{C(c_1c_2)}{T-1} \log \frac{\Pr\left(c_2 \mid c_1\right)}{\Pr\left(c_2\right)} + \sum_{w_2} \frac{\sum_w C(ww_2)}{T-1} \log \underbrace{\Pr\left(w_2 \mid c_2\right) \Pr\left(c_2\right)}_{\Pr(w_2)}. \end{aligned} \tag{8}$$

Therefore, since $\sum_w C(ww_2)/(T-1)$ tends to the relative frequency of w_2 in the training text, and hence to $\Pr(w_2)$, we must have, in the limit,

$$\begin{aligned} L(\pi) &= \sum_w \Pr(w) \log \Pr(w) + \sum_{c_1c_2} \Pr(c_1c_2) \log \frac{\Pr\left(c_2 \mid c_1\right)}{\Pr\left(c_2\right)} \\ &= -H(w) + I(c_1, c_2), \end{aligned} \tag{9}$$

where $H(w)$ is the entropy of the 1-gram word distribution and $I(c_1, c_2)$ is the average mutual information of adjacent classes. Because $L(\pi)$ depends on π only through this average mutual information, the partition that maximizes $L(\pi)$ is, in the limit, also the partition that maximizes the average mutual information of adjacent classes.

We know of no practical method for finding one of the partitions that maximize the average mutual information. Indeed, given such a partition, we know of no practical method for demonstrating that it does, in fact, maximize the average mutual information. We have, however, obtained interesting results using a greedy algorithm. Initially, we assign each word to a distinct class and compute the average mutual information between adjacent classes. We then merge that pair of classes for which the loss in average mutual information is least. After $V - C$ of these merges, C classes remain. Often, we find that for classes obtained in this way the average mutual information can be made larger by moving some words from one class to another. Therefore, after having derived a set of classes from successive merges, we cycle through the vocabulary moving each word to the class for which the resulting partition has the greatest average mutual information. Eventually no potential reassignment of a word leads to a partition with greater average mutual information. At this point, we stop. It may be possible to find a partition with higher average mutual information by simultaneously reassigning two or more words, but we regard such a search as too costly to be feasible.

To make even this suboptimal algorithm practical one must exercise a certain care in implementation. There are approximately $(V-i)^2/2$ merges that we must investigate to carry out the i^{th} step. The average mutual information remaining after any one of them is the sum of $(V-i)^2$ terms, each of which involves a logarithm. Since altogether we must make $V - C$ merges, this straightforward approach to the computation is of order V^5. We cannot seriously contemplate such a calculation except for very small values of V. A more frugal organization of the computation must take advantage of the redundancy in this straightforward calculation. As we shall see, we can make the computation of the average mutual information remaining after a merge in constant time, independent of V.

Suppose that we have already made $V-k$ merges, resulting in classes $C_k(1), C_k(2), \ldots, C_k(k)$ and that we now wish to investigate the merge of $C_k(i)$ with $C_k(j)$ for $1 \leq i < j \leq k$. Let $p_k(l,m) = \Pr\left(C_k(l), C_k(m)\right)$, i.e., the probability that a word in class $C_k(m)$ follows a word in class $C_k(l)$. Let

$$pl_k(l) = \sum_m p_k(l,m), \tag{10}$$

let

$$pr_k(m) = \sum_l p_k(l,m), \tag{11}$$

and let

$$q_k(l,m) = p_k(l,m) \log \frac{p_k(l,m)}{pl_k(l)pr_k(m)}. \tag{12}$$

The average mutual information remaining after $V - k$ merges is

$$I_k = \sum_{l,m} q_k(l,m). \tag{13}$$

We use the notation $i + j$ to represent the cluster obtained by merging $C_k(i)$ and $C_k(j)$.

Thus, for example, $p_k(i+j,m) = p_k(i,m) + p_k(j,m)$ and

$$q_k(i+j,m) = p_k(i+j,m) \log \frac{p_k(i+j,m)}{pl_k(i+j)pr_k(m)}. \tag{14}$$

The average mutual information remaining after we merge $C_k(i)$ and $C_k(j)$ is then

$$\begin{aligned} I_k(i,j) &= I_k - s_k(i) - s_k(j) + q_k(i,j) + q_k(j,i) + q_k(i+j,i+j) \\ &\quad + \sum_{l \neq i,j} q_k(l,i+j) + \sum_{m \neq i,j} q_k(i+j,m), \end{aligned} \tag{15}$$

where

$$s_k(i) = \sum_l q_k(l,i) + \sum_m q_k(i,m) - q_k(i,i). \tag{16}$$

If we know I_k, $s_k(i)$, and $s_k(j)$, then the majority of the time involved in computing $I_k(i,j)$ is devoted to computing the sums on the second line of equation (15). Each of these sums has approximately $V - k$ terms and so we have reduced the problem of evaluating $I_k(i,j)$ from one of order V^2 to one of order V. We can improve this further by keeping track of those pairs l,m for which $p_k(l,m)$ is different from 0. We recall from Table 1, for example, that of the 6.799×10^{10} 2-grams that might have occurred in the training data, only 14,494,217 actually did occur. Thus, in this case, the sums required in equation (15) have, on average, only about 56 non-zero terms instead of 260,741, as we might expect from the size of the vocabulary.

By examining all pairs, we can find that pair, $i < j$, for which the loss in average mutual information, $L_k(i,j) \equiv I_k - I_k(i,j)$, is least. We complete the step by merging $C_k(i)$ and $C_k(j)$ to form a new cluster $C_{k-1}(i)$. If $j \neq k$, we rename $C_k(k)$ as $C_{k-1}(j)$ and for $l \neq i,j$, we set $C_{k-1}(l)$ to $C_k(l)$. Obviously, $I_{k-1} = I_k(i,j)$. The values of p_{k-1}, pl_{k-1}, pr_{k-1}, and q_{k-1} can be obtained easily from p_k, pl_k, pr_k, and q_k. If l and m both denote indices neither of which is equal to either i or j, then it is easy to establish that

$$\begin{aligned} s_{k-1}(l) &= s_k(l) - q_k(l,i) - q_k(i,l) - q_k(l,j) - q_k(j,l) + q_{k-1}(l,i) + q_{k-1}(i,l) \\ s_{k-1}(j) &= s_k(k) - q_k(k,i) - q_k(i,k) - q_k(k,j) - q_k(j,k) + q_{k-1}(j,i) + q_{k-1}(i,j) \\ L_{k-1}(l,m) &= L_k(l,m) - q_k(l+m,i) - q_k(i,l+m) - q_k(l+m,j) - q_k(j,l+m) \\ &\quad + q_{k-1}(l+m,i) + q_{k-1}(i,l+m) \\ L_{k-1}(l,j) &= L_k(l,k) - q_k(l+k,i) - q_k(i,l+k) - q_k(l+k,j) - q_k(j,l+k) \\ &\quad + q_{k-1}(l+j,i) + q_{k-1}(i,l+j) \\ L_{k-1}(j,l) &= L_{k-1}(l,j) \end{aligned} \tag{17}$$

Finally, we must evaluate $s_{k-1}(i)$ and $L_{k-1}(l,i)$ from equations 15 and 16. Thus, the entire update process requires something on the order of V^2 computations in the course of which we will determine the next pair of clusters to merge. The algorithm, then, is of order V^3.

Although we have described this algorithm as one for finding clusters, we actually determine much more. If we continue the algorithm for $V - 1$ merges, then we will have a single cluster which, of course, will be the entire vocabulary. The order in which clusters are merged, however, determines a binary tree the root of which corresponds

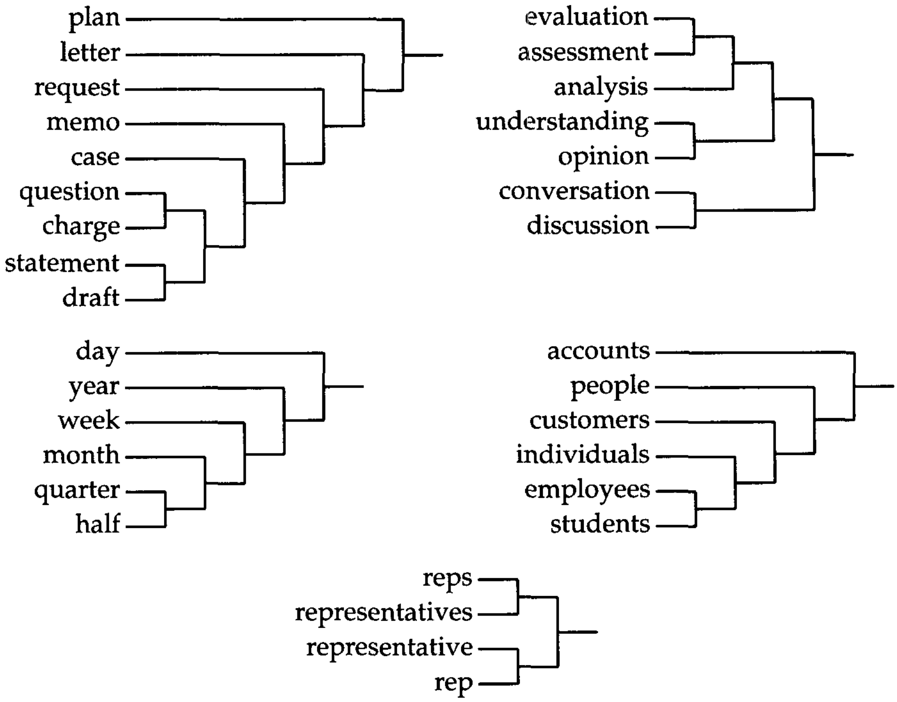

Figure 2
Sample subtrees from a 1,000-word mutual information tree.

to this single cluster and the leaves of which correspond to the words in the vocabulary. Intermediate nodes of the tree correspond to groupings of words intermediate between single words and the entire vocabulary. Words that are statistically similar with respect to their immediate neighbors in running text will be close together in the tree. We have applied this tree-building algorithm to vocabularies of up to 5,000 words. Figure 2 shows some of the substructures in a tree constructed in this manner for the 1,000 most frequent words in a collection of office correspondence.

Beyond 5,000 words this algorithm also fails of practicality. To obtain clusters for larger vocabularies, we proceed as follows. We arrange the words in the vocabulary in order of frequency with the most frequent words first and assign each of the first C words to its own, distinct class. At the first step of the algorithm, we assign the $(C + 1)^{st}$ most probable word to a new class and merge that pair among the resulting $C + 1$ classes for which the loss in average mutual information is least. At the k^{th} step of the algorithm, we assign the $(C + k)^{th}$ most probable word to a new class. This restores the number of classes to $C + 1$, and we again merge that pair for which the loss in average mutual information is least. After $V - C$ steps, each of the words in the vocabulary will have been assigned to one of C classes.

We have used this algorithm to divide the 260,741-word vocabulary of Table 1 into 1,000 classes. Table 2 contains examples of classes that we find particularly interesting. Table 3 contains examples that were selected at random. Each of the lines in the tables contains members of a different class. The average class has 260 words and so to make the table manageable, we include only words that occur at least ten times and

Friday Monday Thursday Wednesday Tuesday Saturday Sunday weekends Sundays Saturdays
June March July April January December October November September August
people guys folks fellows CEOs chaps doubters commies unfortunates blokes
down backwards ashore sideways southward northward overboard aloft downwards adrift
water gas coal liquid acid sand carbon steam shale iron
great big vast sudden mere sheer gigantic lifelong scant colossal
man woman boy girl lawyer doctor guy farmer teacher citizen
American Indian European Japanese German African Catholic Israeli Italian Arab
pressure temperature permeability density porosity stress velocity viscosity gravity tension
mother wife father son husband brother daughter sister boss uncle
machine device controller processor CPU printer spindle subsystem compiler plotter
John George James Bob Robert Paul William Jim David Mike
anyone someone anybody somebody
feet miles pounds degrees inches barrels tons acres meters bytes
director chief professor commissioner commander treasurer founder superintendent dean custodian
liberal conservative parliamentary royal progressive Tory provisional separatist federalist PQ
had hadn't hath would've could've should've must've might've
asking telling wondering instructing informing kidding reminding bothering thanking deposing
that tha theat
head body hands eyes voice arm seat eye hair mouth

Table 2
Classes from a 260,741-word vocabulary.

we include no more than the ten most frequent words of any class (the other two months would appear with the class of months if we extended this limit to twelve). The degree to which the classes capture both syntactic and semantic aspects of English is quite surprising given that they were constructed from nothing more than counts of bigrams. The class {*that tha theat*} is interesting because although *tha* and *theat* are not English words, the computer has discovered that in our data each of them is most often a mistyped *that.*

Table 4 shows the number of class 1-, 2-, and 3-grams occurring in the text with various frequencies. We can expect from these data that maximum likelihood estimates will assign a probability of 0 to about 3.8 percent of the class 3-grams and to about .02 percent of the class 2-grams in a new sample of English text. This is a substantial improvement over the corresponding numbers for a 3-gram language model, which are 14.7 percent for word 3-grams and 2.2 percent for word 2-grams, but we have achieved this at the expense of precision in the model. With a class model, we distinguish between two different words of the same class only according to their relative frequencies in the text as a whole. Looking at the classes in Tables 2 and 3, we feel that

little prima moment's trifle tad Litle minute's tinker's hornet's teammate's
6
ask remind instruct urge interrupt invite congratulate commend warn applaud
object apologize apologise avow whish
cost expense risk profitability deferral earmarks capstone cardinality mintage reseller
B dept. AA Whitey CL pi Namerow PA Mgr. LaRose
# Rel rel. #S Shree
S Gens nai Matsuzawa ow Kageyama Nishida Sumit Zollner Mallik
research training education science advertising arts medicine machinery Art AIDS
rise focus depend rely concentrate dwell capitalize embark intrude typewriting
Minister mover Sydneys Minster Miniter
3
running moving playing setting holding carrying passing cutting driving fighting
court judge jury slam Edelstein magistrate marshal Abella Scalia larceny
annual regular monthly daily weekly quarterly periodic Good yearly convertible
aware unaware unsure cognizant apprised mindful partakers
force ethic stoppage force's conditioner stoppages conditioners waybill forwarder Atonabee
systems magnetics loggers products' coupler Econ databanks Centre inscriber correctors
industry producers makers fishery Arabia growers addiction medalist inhalation addict
brought moved opened picked caught tied gathered cleared hung lifted

Table 3
Randomly selected word classes.

Count	1-gram	2-grams	3-grams
1	0	81,171	13,873,192
2	0	57,056	4,109,998
3	0	43,752	2,012,394
> 3	1,000	658,564	6,917,746
> 0	1,000	840,543	26,913,330
≥ 0	1,000	1,000,000	1.000×10^9

Table 4
Number of class n-grams with various frequencies in 365,893,263 words of running text.

this is reasonable for pairs like *John* and *George* or *liberal* and *conservative* but perhaps less so for pairs like *little* and *prima* or *Minister* and *mover.*

We used these classes to construct an interpolated 3-gram class model using the same training text and held-out data as we used for the word-based language model we discussed above. We measured the perplexity of the Brown corpus with respect to this model and found it to be 271. We then interpolated the class-based estimators with the word-based estimators and found the perplexity of the test data to be 236, which is a small improvement over the perplexity of 244 we obtained with the word-based model.

4. Sticky Pairs and Semantic Classes

In the previous section, we discussed some methods for grouping words together according to the statistical similarity of their surroundings. Here, we discuss two additional types of relations between words that can be discovered by examining various co-occurrence statistics.

The mutual information of the pair w_1 and w_2 as adjacent words is

$$\log \frac{\Pr(w_1 w_2)}{\Pr(w_1)\Pr(w_2)}. \tag{18}$$

If w_2 follows w_1 less often than we would expect on the basis of their independent frequencies, then the mutual information is negative. If w_2 follows w_1 more often than we would expect, then the mutual information is positive. We say that the pair $w_1 w_2$ is *sticky* if the mutual information for the pair is substantially greater than 0. In Table 5, we list the 20 stickiest pairs of words found in a 59,537,595-word sample of text from the Canadian parliament. The mutual information for each pair is given in bits, which corresponds to using 2 as the base of the logarithm in equation 18. Most of the pairs are proper names such as *Pontius Pilate* or foreign phrases that have been adopted into English such as *mutatis mutandis* and *avant garde*. The mutual information for *Humpty Dumpty*, 22.5 bits, means that the pair occurs roughly 6,000,000 times more than one would expect from the individual frequencies of *Humpty* and *Dumpty*. Notice that the property of being a sticky pair is not symmetric and so, while *Humpty Dumpty* forms a sticky pair, *Dumpty Humpty* does not.

Word pair	Mutual Information
Humpty Dumpty	22.5
Klux Klan	22.2
Ku Klux	22.2
Chah Nulth	22.2
Lao Bao	22.2
Nuu Chah	22.1
Tse Tung	22.1
avant garde	22.1
Carena Bancorp	22.0
gizzard shad	22.0
Bobby Orr	22.0
Warnock Hersey	22.0
mutatis mutandis	21.9
Taj Mahal	21.8
Pontius Pilate	21.7
ammonium nitrate	21.7
jiggery pokery	21.6
Pitney Bowes	21.6
Lubor Zink	21.5
anciens combattants	21.5
Abu Dhabi	21.4
Aldo Moro	21.4
fuddle duddle	21.4
helter skelter	21.4
mumbo jumbo	21.4

Table 5
Sticky word pairs.

we our us ourselves ours
question questions asking answer answers answering
performance performed perform performs performing
tie jacket suit
write writes writing written wrote pen
morning noon evening night nights midnight bed
attorney counsel trial court judge
problems problem solution solve analyzed solved solving
letter addressed enclosed letters correspondence
large size small larger smaller
operations operations operating operate operated
school classroom teaching grade math
street block avenue corner blocks
table tables dining chairs plate
published publication author publish writer titled
wall ceiling walls enclosure roof
sell buy selling buying sold

Table 6
Semantic clusters.

Instead of seeking pairs of words that occur next to one another more than we would expect, we can seek pairs of words that simply occur near one another more than we would expect. We avoid finding sticky pairs again by not considering pairs of words that occur too close to one another. To be precise, let $\Pr_{\text{near}}(w_1 w_2)$ be the probability that a word chosen at random from the text is w_1 and that a second word, chosen at random from a window of 1,001 words centered on w_1 but excluding the words in a window of 5 centered on w_1, is w_2. We say that w_1 and w_2 are *semantically sticky* if $\Pr_{\text{near}}(w_1 w_2)$ is much larger than $\Pr(w_1)\Pr(w_2)$. Unlike stickiness, semantic stickiness is symmetric so that if w_1 sticks semantically to w_2, then w_2 sticks semantically to w_1.

In Table 6, we show some interesting classes that we constructed, using $\Pr_{\text{near}}(w_1 w_2)$, in a manner similar to that described in the preceding section. Some classes group together words having the same morphological stem, such as *performance, performed, perform, performs,* and *performing*. Other classes contain words that are semantically related but have different stems, such as *attorney, counsel, trial, court,* and *judge.*

5. Discussion

We have described several methods here that we feel clearly demonstrate the value of simple statistical techniques as allies in the struggle to tease from words their linguistic secrets. However, we have not as yet demonstrated the full value of the secrets thus gleaned. At the expense of a slightly greater perplexity, the 3-gram model with word classes requires only about one-third as much storage as the 3-gram language model in which each word is treated as a unique individual (see Tables 1 and 4). Even when we combine the two models, we are not able to achieve much improvement in the perplexity. Nonetheless, we are confident that we will eventually be able to make significant improvements to 3-gram language models with the help of classes of the kind that we have described here.

Acknowledgment

The authors would like to thank John Lafferty for his assistance in constructing word classes described in this paper.

References

Averbuch, A.; Bahl, L.; Bakis, R.; Brown, P.; Cole, A.; Daggett, G.; Das, S.; Davies, K.; Gennaro, S. De.; de Souza, P.; Epstein, E.; Fraleigh, D.; Jelinek, F.; Moorhead, J.; Lewis, B.; Mercer, R.; Nadas, A.; Nahamoo, D.; Picheny, M.; Shichman, G.; Spinelli, P.; Van Compernolle, D.; and Wilkens, H. (1987). "Experiments with the Tangora 20,000 word speech recognizer." In *Proceedings, IEEE International Conference on Acoustics, Speech and Signal Processing.* Dallas, Texas, 701–704.

Bahl, L. R.; Jelinek, F.; and Mercer, R. L. (1983). "A maximum likelihood approach to continuous speech recognition." *IEEE Transactions on Pattern Analysis and Machine Intelligence,* PAMI-5(2), 179–190.

Baum, L. (1972). "An inequality and associated maximization technique in statistical estimation of probabilistic functions of a Markov process." *Inequalities,* 3, 1–8.

Brown, P. F.; Cocke, J.; DellaPietra, S. A.; DellaPietra, V. J.; Jelinek, F.; Lafferty, J. D.; Mercer, R. L.; and Roossin, P. S. (1990). "A statistical approach to machine translation." *Computational Linguistics,* 16(2), 79–85.

Dempster, A.; Laird, N.; and Rubin, D. (1977). "Maximum likelihood from incomplete data via the EM algorithm." *Journal of the Royal Statistical Society,* 39(B), 1–38.

Feller, W. (1950). *An Introduction to Probability Theory and its Applications, Volume I.* John Wiley & Sons, Inc.

Gallagher, R. G. (1968). *Information Theory and Reliable Communication.* John Wiley & Sons, Inc.

Good, I. (1953). "The population frequencies of species and the estimation of population parameters." *Biometrika,* 40(3–4), 237–264.

Jelinek, F., and Mercer, R. L. (1980). "Interpolated estimation of Markov source parameters from sparse data." In *Proceedings, Workshop on Pattern Recognition in Practice,* Amsterdam, The Netherlands, 381–397.

Kučera, H., and Francis, W. (1967). *Computational Analysis of Present Day American English.* Brown University Press.

Mays, E.; Damerau, F. J.; and Mercer, R. L. (1990). "Context-based spelling correction." In *Proceedings, IBM Natural Language ITL.* Paris, France, 517–522.