Statistical Identification of Language

Ted Dunning
Computing Research Laboratory
New Mexico State University

March 10, 1994

Abstract

A statistically based program has been written which learns to distinguish between languages. The amount of training text that such a program needs is surprisingly small, and the amount of text needed to make an identification is also quite small. The program incorporates no linguistic presuppositions other than the assumption that text can be encoded as a string of bytes. Such a program can be used to determine which language small bits of text are in. It also shows a potential for what might be called 'statistical philology' in that it may be applied directly to phonetic transcriptions to help elucidate family trees among language dialects.

A variant of this program has been shown to be useful as a quality control in biochemistry. In this application, genetic sequences are assumed to be expressions in a language peculiar to the organism from which the sequence is taken. Thus language identification becomes species identification.

Introduction

Given the following 20 character strings,

```
e pruebas bioquimica
man immunodeficiency
faits se sont produi
```

it is hardly surprising that a person can identify the languages as Spanish, English and French, respectively. It isn't even surprising that a person who speaks essentially no French or Spanish can do this correctly. Clearly, language understanding is not required for language identification. Furthermore, only the French string contains any closed class words, and none of the strings contain any accented characters which are unique to that language (erroneously so in the Spanish example).

Given this simple fact, it is a natural step to wonder just how simple a computer program could be which performs a comparable job of language identification. It is also interesting to ask whether such a program might be derived from relatively general mathematical principles, or whether it could be written in such a way that it could learn the characteristics of the languages to be distinguished. Another important question is how many characters are needed to reliably identify the language of a string. Finally, it is important to know just how broad the applicability of such a program actually is.

This article reports some answers to these questions. The program can be very simple. In fact, a few hundred lines of C suffice, of which the statistical classifier takes up considerably less than 100 lines of code. The algorithm used can be derived from basic statistical principles using only a few assumptions. The program is not based on hand-coded linguistic knowledge, but can learn from training data.

This program works moderately well at classifying strings as short as 10 characters, and works very well when given strings of 50 or more characters. The amount of training data is quite small; as little as a few thousand words of sample text is sufficient for good performance.

Further, the program exhibits interesting generalization behaviors. For instance, when it is trained on English, French and Spanish, the program tends strongly to classify German as English. This is provocative given the historical basis of English as a Germanic language. The program can be used on a wide variety of materials with the extreme case so far being the analysis of genetic sequences.

Furthermore, the algorithm used by the program can be derived mathematically. Bayesian methods can be used by assuming that language can be modeled by a low order Markov process which generates characters. This derivation of the classification methods from relatively basic statistical assumptions is more satisfying than writing a program based on ad hoc heuristics since ad hoc heuristics may not be available or apparent when changing to new languages or new domains.

Previous work

The problem of identification of the language of a text sample has been addressed a number of times. In many cases, these efforts are not reported in the literature, and in any case there has been no systematic comparison of approaches. Furthermore, most approaches taken so far have used some degree of a priori linguistic knowledge. This makes direct comparison with methods which are trained on a limited amount of data difficult. Furthermore, many approaches have been based on assumptions which may not hold for all languages of interest, or for all applications. For example, the amount of text available to be classified may be quite limited, or one or more of the languages in question may not be easy to tokenize into words. In the extreme case where the texts being analyzed are not actually human language at all, (e.g. genetic sequences) such assumptions may break down entirely, thus severely limiting the utility of a particular approach in some potential area of application.

While language identification is an extremely simple task when compared to other canonical computational linguistic tasks such as machine translation or information retrieval, it provides a perhaps surprising amount of depth in terms of performance tradeoffs and the variety of approaches which have been tried. The availability of a test set such as the one described in this paper will make it possible for new approaches to be tested with minimal effort on the part of the implementor and will also allow the results of such tests to be directly compared to the results obtained by other methods.

Unique letter combinations

There has been some commentary in electronic fora regarding the possibility of enumerating a number of short sequences which are unique to a particular language.

In particular, the following list of characteristic sequences was proposed by a variety of people [Chu94]

Language	**String**
Dutch	"vnd"
English	"ery "
French	"eux "
Gaelic	"mh"
German	" der "
Italian	"cchi "
Portuguese	" seu "
Serbo-croat	"lj"
Spanish	" ir "

Clearly these particular strings do not serve as reliable signposts. No one could claim that any text that discussed zucchini or Pinocchio would have to be Italian, only a killjoy would claim that Serbo-Croat had the only 'lj' strings or that Amherst, Elmhurst, farmhands or farmhouses could only be discussed in Gaelic. And borrowing of geographical terms such as Montreaux or the simple adoption of words such as milieux into English would also confound these tests.

While better strings could no doubt be devised, the problems with the unique string method are clear, and do not go away when more effort is made to be clever. The problems do not lie in lack of cleverness, they lie in the fact that such a unique strings approach depends on the presence of only a very few strings, and that it fails to weight the evidence that it does find. Adding all observed strings leads to the n-gram methods and picking an optimal weighting scheme gives the Bayesian decision rule system described later. Thus, the unique strings approach incorporates a valid intuitive germ, but this germ alone is not a strong enough foundation on which to build a usable decision rule.

Common words

Another approach which has been proposed a number of times is to devise lists of the words which commonly appear in a number of different languages. Since such common words make up substantial amounts of running text, this approach works relatively well, if enough text is available to be classified. This approach has been tested by the group at Leeds [Joh93] and is reported to have been tried by researchers with the Linguistic Data Consortium.

As the 20-character sequences given in the introduction show, however, as the strings to be classified become very short, there is a substantial chance that the strings will not contain any common words. This problem is exacerbated by the bursty nature of text when viewed on a small scale; the closed class words form a structure around pockets of less common words. Furthermore, the short strings which are interesting to identify are often exactly the strings which do not contain the relatively common closed class words. Rather, the strings of interest tend to be the comparatively longer rare words and proper names. For example, a machine translation program might be very interested in being able to identify English phrases in a primarily Spanish text, or to determine that an unknown company name was likely to be Japanese in origin. In neither case could the common word approach be expected work well.

Additionally, since common words tend to be short, systems such as the n-gram which statistically weight the evidence provided by the short strings found in text will naturally incorporate what information is available from short words. They will also be able to incorporate the information provided by common suffixes and prefixes. Furthermore, the common word methods are somewhat limited in that they depend on the ability to define and recognize common words. If tokenization into words is difficult (as in Chinese), or if defining a set of common words is difficult (as in the case of genetic sequences) then the common word approach may be difficult or impossible to apply.

N-gram counting using ad hoc weighting

The group at Leeds has experimented with low order n-gram models as described in [Hay93] and [CHJS94]. They used an ad hoc statistical model to build their classifier which appears likely to introduce some brittleness into their system. Additionally, due to the design of their test corpus it is difficult to disentangle the different factors which affect performance. Their scoring method was based on directly using estimated probabilities which are combined linearly. Basic statistical considerations show that if linear combination is to be done, then the logarithms of the estimated probabilities should be used and some sort of smoothing should be done when these probabilities are estimated. This non-optimal combining method could well be responsible for some of the anomalous results reported by the Leeds group.

N-gram counting using rank order statistics

Cavnar and Trenkle [CT94] have reported the results of using an ad hoc rank order statistic to compare the prevalence of short letter sequences. They first tokenize the test and training texts in order to avoid sequences which straddle two words.

A rank ordered list of the most common short character sequences was derived from their training data after tokenization. Language identification was done by compiling a similar list of common short strings found in the string to be classified and comparing this list to the profiles for the different training corpora. This comparison was done using an adhoc rank ordering statistic which was designed to be relatively insensitive to omissions.

They were able to extract a test database from a variety of network news groups. This provided samples of 8 different languages (Dutch, French, German, Italian, Polish, Portuguese and Spanish). These samples ranged in size from 22Kbytes for Dutch to 150Kbytes for Portugese.

This system was able to achieve very good performance when classifying long strings (4K bytes or about 700 words of English or Spanish). They report that their system was relatively insensitive to the length of the string to be classified, but the shortest strings that they reported classifying were 300 bytes. This would be about 50 words in English.

Some weaknesses in this work are that it requires the input text to be tokenized and that the statistical characteristics of their rank order comparison are not known and appear to be difficult to derive rigorously. It may well be that their tokenization step is not necessary, which would allow their method to be applied in cases where tokenization is difficult. This issue was not explored in their reported work. It is possible that removing character sequences which straddle word boundaries from consideration would eliminate any word sequence information form consideration which might be harmful to performance. Other than the tokenization step and the particular statistic used, their work is similar to the work described in this paper.

A highly significant aspect of Cavnar and Trenkle's work is that they have agreed to make their test and training data available to other researchers by donating it to the Consortium for Lexical Resources. This means that it will be possible to compare the accuracy of different approaches under comparable conditions. At the time that this paper was written, it also appeared that they might also be willing to make their language classifier program available so that it could be run on other test and training corpora such as the one used in the work reported in this paper.

Classification Methods

The basic method used in the work described in this paper is to develop a set of character level language models from the training data and then to use these language models to estimate the likelihood that a particular test string might have been generated by each of the language models. Since the language models are derived by assuming only that the training and test data are sequences of bytes, no language specific pre-processing is required. Furthermore, the statistical analysis used comes with some guarantees of optimality under certain limited conditions.

Markov models

A Markov model is a random process in which the probability distribution of the next state depends solely on the current state. More formally, the Markov model defines a random variable whose values are strings from an alphabet X, and where the probability of a particular string S is

$$p(S) = p(s_1 \dots s_n) = p(s_1) \prod_{i=2}^{n} p(s_i \mid s_{i-1})$$

Such a Markov model is completely characterized by the initial state distribution $p(s_1)$ and the transition probabilities $p(s_i \mid s_{i-1})$. Markov models have several characteristics which make them attractive: they are easy to deal with mathematically, they are causal, and they can be described relatively succinctly.

A random process where the distribution of the next state depends only on the last k states can be relabeled so that it is also a Markov model. This is done by using the last k states as the label for the current state. The transition probabilities for this relabeled model are simply

$$p(s_{i+1} \ldots s_{i+k} | s_i \ldots s_{i+k-1}) = p(s_{i+k} \mid s_i \ldots s_{i+k-1})$$

It is customary to broaden the definition of a Markov model to include these limited history models which are then known as Markov models of order k. The evolution of a Markov model is described completely by the distribution of initial states $p(s_1 \ldots s_k)$ and the transition probabilities $p(s_{i+k} \mid s_i \ldots s_{i+k-1})$.

We can produce a rudimentary model of natural language by assuming that the distributions of strings in the language satisfy the distribution for a Markov model of order k where k is relatively small and the alphabet X is the set of characters in the language (say the ISO-Latin-1 character set).

It cannot be claimed that a low order Markov model captures very much of the structure of a language. Higher order models capture this structure much better, but a higher order model is not necessarily better for the purposes of language identification. This is because the amount of training data needed to accurately estimate the parameters of the Markov model is roughly exponential in the order of the model. With an insufficient training set, a higher order model will behave in a brittle manner; only texts very similar to those used for training will have reasonable probabilities, texts dissimilar to the training texts will have significantly underestimated probabilities. If used for generation, the strings generated by a high order model will consist primarily of relatively long segments taken directly from the training texts.

Given this trade-off, there is an optimum where the model captures enough language structure to be useful as a categorizer, but where the order is low enough that only a reasonable amount of training text is needed to attain satisfactory performance.

The following are samples of text generated by Markov models with order from 0 to 6 which were trained using a 6000 word article on computational linguistics:

```
0  hm 1 imuandno~doc ni leotLs Aiqe1pdt6cf tlc.teontctrrdsxo~es loo oil3s
1  ~ a meston s  oflas n,~ 2 nikexihiomanotrmo s,~125 0 3 1 35  fo there
2  s ist  des anat p sup sures Alihows raiial on terliketicany of prelly
3  approduction where.  If the lineral wate probability the or likelihood
4 sumed normal of the normal distribution.  Church, Gale, Willings. This
5 ~k sub 1} sup {n-k}  .EN  where than roughly 5.  This agreemented by th
6 these mean is not words can be said to specify appear.  McDonald.  1989
```

While these are clearly *not* valid English, it is also clear that some essence of the language in the original article has been captured by the higher order models. In fact, entire chunks of equations and parts of the original text are reproduced verbatim. This verbatim reproduction is an artifact of using a maximum likelihood estimation technique with only a very limited amount of training data. Unless some sort of smoothing is applied to models derived from the necessarily sparse data available, even a relatively low order Markov model is highly subject to such overtraining.

Bayesian Decision Rules

Given the task of deciding which of two possible phenomena have caused a particular observation, we can minimize our probability of error by computing which is most likely to have given rise to the observation.

Thus if we are choosing between A and B given an observation X, where we feel that we know how A or B might affect the distribution of X, we can use Bayes' Theorem:

$$p(A, X) = p(A \mid X)p(X) = p(X \mid A)\, p(A)$$

But we know that $p(X) = 1$ since X was actually observed, so

$$p(A \mid X) = p(X \mid A)\, p(A)$$

similarly

$$p(B, X) = p(B \mid X) = p(X \mid B)\, p(B)$$

Our overall probability of error is minimized if we pick whichever A or B has a larger probability given the observation X. Strictly speaking, this requires knowledge of the prior probabilities $p(A)$ and $p(B)$. In practice, if these are truly unknown, good results can often be had by assuming that they are equal.

This method of picking the most likely cause is called a Bayesian decision rule. The method can be applied in cases where there are more than two phenomena, or even in cases involving the estimation of a continuous variable.

In the case of strings which are assumed to have been generated by one of a number of Markov models, the probability of observing string S given a particular Markov model A is

$$p(S \mid A) = p(s_1 \ldots s_k \mid A) \prod_{i=k+1}^{N} p(s_i \mid s_{i-k} \ldots s_{i-1} \mid A)$$

We can make this a bit simpler by assuming that all strings start with a sequence of k special symbols so that the leading term is always 1. We can force this to be true by padding all strings with these special symbols before estimating the model parameters. In practice, the leading term is just dropped without justification.

This product can be further simplified by rearranging it so that all like terms are grouped together to get

$$p(S \mid A) = \prod_{w_1 \ldots w_{k+1} \in S} p(w_{k+1} \mid w_1 \ldots w_k \mid A)\, T(w_1 \ldots w_{k+1}, S)$$

Here we have written $T(w_1 \ldots w_{k+1}, S)$ to mean the number of times that the $k+1$ gram $w_1 \ldots w_{k+1}$ occurs in string S. The product is done over all strings found in the string S (i.e. for all $w_1 \ldots w_{k+1} \in S$).

Computationally, it is much more convenient to compare logarithms of these conditional probabilities since this avoids problems of numeric underflow. This gives us

$$\log p(S \mid A) = \sum_{w_1 \ldots w_{k+1} \in S} T(w_1 \ldots w_{k+1}, S) \log p(w_{k+1} \mid w_1 \ldots w_k \mid A)$$

By computing this value for each of our language models (A, B, etc.) and picking the largest (assuming equal prior probabilities), we can pick the language model which is most likely to have generated the observed string. We can use any available information about the prior probability of seeing text generated by each of the language models, or we can assume equal prior probabilities depending on the particularities of our situation. If the language models are accurate models of the languages, then this selection should be our best bet for deciding which language S is from.

But how should we derive the model parameters (the p's)?

Estimating Model Parameters

The most obvious method for estimating the values of our model parameters is to use the ratios of counts derived from the training data. If training data for language model A consists of the string T_A, this method gives

$$p(w_{k+1} \mid w_1 \dots w_k \mid A) = \frac{T(w_1 \dots w_{k+1}, T_A)}{T(w_1 \dots w_k, T_A)}$$

This estimate maximizes $p(T_A \mid A)$ and is known as the maximum likelihood estimator. Such estimates are intuitively appealing, and additionally, they have good asymptotic properties and are often simple to derive.

For our task of classifying strings using limited training data, however, direct use of the maximum likelihood estimator is not at all satisfactory. The problem is that $k+1$ grams which are not observed in the training data are given zero probabilities. This causes havoc in the real world since it is not uncommon for a substring to be missing from our (limited) training data, but present in the actual experimental data. Using the maximum likelihood estimator in such a case will give $p(S \mid A) = 0$. Furthermore, if a particular $k+1$ gram is relatively rare for all the languages involved, but actually happens to appear in the training data for A, then any test string containing this deadly substring will be judged to be from language A. All the other language models will give probabilities of 0.

This proclivity toward disaster due to overtraining makes maximum likelihood estimators unsatisfactory for our classification task. Happily, we can do much better.

One approach which works well is to find an estimator which minimizes the mean squared error of our expression for $p(S \mid A)$ or for $\log p(S \mid A)$. A well known result is that with a uniform prior distribution this minimum error is achieved by the Bayesian estimator. Given a uniform prior, to estimate a function f of the model parameters θ with the minimum squared error, given an observed S, the optimal estimate is

$$\hat{f} = \frac{\int p(S \mid \theta) f(\theta)\, d\theta}{\int p(S \mid \theta)\, d\theta}$$

In the case of the log of the probability that a string was generated by a particular Markov model, f is the sum of terms like $T(w_1 \ldots w_{k+1}, S) \log p(w_{k+1} \mid w_1 \ldots w_k)$. By linearity, the Bayesian estimator can be decomposed so that what we only need to find is the estimators for $\log p(w_{k+1} \mid w_1 \ldots w_k)$ and reconstruct the estimator of the larger expression from those.

The derivation of this estimator is done using a sequence of change of variables, Laplace transforms and application of the convolution theorem. A similar derivation is given in [WW92] and a summary of the derivation is given in a later section of this paper. The desired estimator is

$$\hat{f} = \psi(T(w_1 \ldots w_{k+1}, T_A) + 2) - \psi(T(w_1 \ldots w_k, T_A) + m + 1)$$

where ψ is the digamma function $\psi(x) = \frac{d \log \Gamma(x)}{dx}$ and $m = |A|$ is the size of the alphabet.

This estimator is a bit unpleasant to implement. Instead, the fact that $\psi(x) \approx \log(x-1)$ can be used to obtain the approximation

$$\hat{f} = \log \frac{T(w_1 \ldots w_{k+1}, T_A) + 1}{T(w_1 \ldots w_k, T_A) + m}$$

which can be used without undue loss of accuracy. This last form is interesting since the Bayesian estimator for $p(w_{k+1} \mid w_1 \ldots w_k)$ itself is

$$\hat{p} = \frac{T(w_1 \ldots w_{k+1}, T_A) + 1}{T(w_1 \ldots w_k, T_A) + m}$$

This expression was derived originally by Laplace and is sometimes called Laplace's sample size correction.

More refined estimates of the model parameters can be made using assumptions about the relationship between the best order k Markov model and a lower order Markov model. This is often done using held out data for smoothing. For the purposes of language identification, it does not appear that refining the accuracy of the estimator for the smaller model parameters will have much impact on the performance since it is the larger parameters which are responsible for most of the identification power. These larger parameters are well estimated by relatively simple techniques. In practice, we have found that simply using the Laplace sample size correction gives results indistinguishable in practice from other estimators.

The results described in this paper were obtained using a program which applied Laplace's correction to directly estimate the stationary $k+1$-gram probabilities from the training data. This estimator is not particularly defensible, but it does have the advantage of being easy to implement. Furthermore, if the $k+1$-gram probabilities are expressed in terms of conditional probabilities, we see that

$$\sum_{w_1 \ldots w_{k+1} \in S} T(w_1 \ldots w_{k+1}) \log p(w_{k+1} \mid w_1 \ldots w_k) = \sum_{i=1}^{k} l_i(S \mid A)$$

where the l_i are log-probabilities computed using only i-grams:

$$l_1(S|A) = \sum T(w_1, T_A) \log p(w_1, T_A)$$

$$l_2(S|A) = \sum T(w_1 w_2, T_A) \log p(w_1 w_2, T_A)$$

and so on,

$$l_k(S|A) = \sum T(w_1 \ldots w_k, T_A) \log p(w_1 \ldots w_k, T_A)$$

Thus using the logarithms of the $k+1$-gram probabilities gives us a decision rule which incorporates the evidence from Markov models with order from 0 to k. Based on these heuristic arguments, we estimated the stationary $k+1$-gram probabilities and used them to do the classification.

Practical Results

The performance of our n-gram language classifier was evaluated using a specially constructed test corpus. Versions of the classifier using different size n-grams and slightly different approximations of the pertinent test statistic were compared.

Construction of the bilingual test corpus

We identified the following conditions as relevant to the performance of a language classifier

1. how the test strings are picked,
2. the amount of training material available,
3. the size of the strings to be identified,
4. the number of languages to be identified, and
5. whether there is a correlation between domain and language.

It should be noted that language identification programs sometimes detect the domain of a text. For instance, Cavnar and Trenkle's n-gram classifier was used with some success as the basic engine in an information retrieval application [CT94]. This means that it may be difficult to separate the tasks of language identification and domain (or subject) identification. This is especially true if a language classifier is trained and tested on material in which the domain of discourse is strongly correlated with the language used.

We have avoided this problem of domain dependence and addressed all the pertinent variables except the number of languages by constructing a test corpus from a parallel translated corpus in English and Spanish. This test corpus provides 10 training texts with lengths of 1000, 2000, 5000, 10,000 and 50,000 bytes respectively, and provides 100 test texts with lengths of 10, 20, 50, 100, 200 and 500 bytes. In total, 50 training texts and 600 test texts were selected initially in each of the 2 languages. The training and test texts were selected by concatenating together all of the documents in a particular language and then selecting a contiguous sequence of bytes with a uniformly distributed starting point.

All of the the test strings were examined manually and any strings which were not clearly Spanish or English were excluded from the test set and replaced with newly generated strings. These excluded strings were kept for possible later examination. This was particularly a problem with the very short test strings since a substantial number of the short test strings consisted almost entirely of white space, numbers or punctuation. There were also a few situations where a string from the Spanish side of the parallel corpus was actually in English or visa versa. These cases were also excluded from the final corpus. All excluded texts were preserved. This procedure allows estimates to be made of the performance of a language classifier under a number of different operating assumptions.

None of the training texts were excluded from the test corpus, although one of the shorter training texts consisted primarily of upper case text from a table of contents and another consisted primarily of numbers taken from a table. The effect of these problematic training texts was limited by using the median instead of the mean when plotting performance figures. Best and worst performance figures were also plotted to indicate the range of performance which can be expected with this method.

In order to facilitate comparisons between different approaches, this test corpus will be made available via the Consortium for Lexical Research. Questions regarding how to obtain the corpus should be addressed to `lexical@nmsu.edu` or to the author of this report.

Construction of the genetic sequence test corpus

In addition to the bilingual corpus, we have prepared an additional test corpus containing genetic sequences from three organisms. We view the problem of identifying which organism a DNA sequence comes from as a biological parallel to the problem of language identification. While we do not make strong claims about parallels between the processes which ultimately lead to the differentiation of either languages or genomes, it is provocative to think that some of the same methods can be used to manipulate both.

The data in the genetic corpus have been prepared in a manner similar to that used to prepare the bilingual test corpus except for a few minor differences. These differences include the following

- the test sequences could not be checked manually,
- only one training text is given for each organism,
- the sizes of the test strings are much longer, in general, than the test strings from the bilingual corpus, and
- the test strings include sequences known to be from human, E. Coli or yeast, as well as some sequences whose origin is in doubt due to experimental errors, and

- the concept of using a parallel corpus for constructing a test corpus has no simple analog in the problem of distinguishing yeast from human sequences

The most important difference between the bilingual and genetic test corpora is related to the intended application of a statistical classifier in the two different areas. In the case of text, there are a wide variety of potential operational settings for a language classifier which will present very different problems. In the case of the genetic sequence classifier, the primary application to date has been the detection of yeast contamination of human expressed sequence tags (EST's). As such, the genetic test corpus is designed to test the applicability of software to exactly that problem. For example, the yeast training sequences include both introns and exons while the human training sequences consist only of known EST's which are all exonic in nature. Since the practical problem our classifier was intended to solve was the differentiation of arbitrary yeast sequence from human EST's, the current design of the genetic test corpus reflects the intended operational setting for the software and any confounding effects will have parallel effects in practice.

In previous work [WDS+92] an n-gram statistical method was shown to be effective in distinguishing short sequences of yeast, bacterial and human DNA. This software was used to demonstrate that a serious procedural error had resulted in the contamination of a large number of supposedly human sequences with sequences which actually came from some other organism [edi93]. The practical result was the withdrawal of the entire set of questionable sequences from the international sequence databases. Currently, the inclusion of n-gram based quality control software is being considered by several groups involved in the creation of sequence authoring software.

Test Methods

The results presented in this section were obtained by training the Bayesian classifier described previously on each training set from the English/Spanish test corpus, classifying all test strings and then recording the result. In order to clearly highlight when the classifier has insufficient data to make a valid decision, the classifier was run in such a way that the default answer in the absence of data was incorrect. In some extreme cases, it was therefore possible to get error rates well in excess of 50%. In normal operation, error rates this high are extremely unlikely, but for comparison purposes, rigging the test against the program in this way helps highlight failures.

Figure 1 shows the variation in error rate depending on the order of the underlying Markov model, the size of the test string and the amount of training data used. As can be seen, with small amounts of training data, lower order models work best, while with more training data, higher order models become practical. There is an optimum in performance using the bigram and trigram models ($k = 1$ and $k = 2$, respectively). This pattern holds for most combinations of test string and training data sizes, although with longer strings (200 characters or more) and large amounts of training data, the error rate is essentially zero for any $k > 0$.

Figures 2 and 3 provide a closer view of the variation in error rate with small (2Kbytes) and large (50Kbytes) training sets. They also illustrate the large variability in error rate when small training sets are used with higher order models.

Discussion of results

Accuracy of about 92% can be achieved with only 20 bytes of test text and 50K of training, this improves to about 99.9% accuracy when 500 bytes are examined. With only 5K of training text, examining 500 bytes gives an accuracy of roughly 97%. All of these estimates of classification accuracy are subject to statistical variation, but for longer test strings (>100 bytes) and larger training sets (50Kbytes or more) we can have roughly 90% confidence that the accuracy will be >99%.

Classification accuracy is good and improves with longer test strings and with more test data. With shorter test strings or with less training data, bigram models appear to perform best. With large amounts of training data, and longer test strings, only the unigram model has high enough error rates to measure easily with our experimental design.

Idiosyncratic training data can be a problem, especially if only a limited amount is available. These effects can be minimized by using bigram models.

Since there are literally billions of 4 and 5-grams, it is clear that 50 Kbytes of training data is extremely unlikely to be enough data to accurately estimate all model parameters. On the other hand, good accuracy is obtained with only this much training data. Apparently, the model parameters which are responsible for discrimination performance converge to sufficient accuracy long before all parameters can be well estimated.

Because this experiment was designed to avoid accidentally correct classification, and to hide confounding sources of information from the classifier, it seems likely that performance in practice will be even better than the results obtained in these experiments.

Data and software availability

The program described in this report is available from the Consortium for Lexical Research. Subject to final permission for distribution, the test corpus described in this paper will also be made available via the Consortium.

Future and related work

Simple classifiers such as those described in this report have a number of applications. For instance, a machine translation system might be able to use such a system to determine whether short strings which are difficult to translate should be carried verbatim, or which language a source file is in.

It may be that such systems could also be used in conjunction with a speaker independent phoneme recognizer to identify spoken as well as written languages. This would have clear utility in certain intelligence applications. The input data would be much noisier than the textual data used to produce the results described in this paper, but the success of this system when used to classify short bits of genetic sequences indicates that noise may not be a problem with this sort of classifier. Clearly, techniques which depend on tokenization or which are sensitive to noise would not be suitable candidates for this task.

It is also possible that this method could be applied to hand generated phonetic transcriptions of speech to provide an objective measure of similarity between different spoken dialects. This process would be subject to bias introduced by the hand coding process. In some sense, this measure of relatedness would be similar to the language clustering work described in [BKP92], but while they classified languages according to the editing distance between corresponding renditions of 15 common words, applying an n-gram metric would allow the comparison of language similarities based on realistic texts rather than a few specially chosen words. While the work of Batagelj et al. clearly shows that orthographic normalization is not strictly necessary, using a phonetic transcription would avoid some of the tendency to do comparisons of orthographic families rather than linguistic or dialectic families. Unlike much corpus based work, only a relatively modest amount of transcribed material would be needed (a few thousand words).

Further research into the application of these methods as a quality control method in genetic applications is also important. If a simple classifier such as this could be integrated into sequence authoring or database submission tools, it might be possible that major procedural errors could be detected much earlier in the sequencing process, with the effect of improving the cost effectiveness of sequencing projects. In fact, it is even conceivable that these statistical classifiers could be integrated into fragment assembly software to improve the probabilistic estimates used in the assembly process and reduce the problem of incorporating foreign fragments into an assembled sequence.

Since the techniques necessary to produce good results are so simple, source identification can also serve as a useful pedagogical tool. The availability of standard data sets makes it possible to have students implement their own algorithms and compare them directly to standardized results. Problems in computational linguistics at this level of complexity are relatively rare and standardized test cases are essentially unavailable.

Sample Derivations of Bayesian estimators

The Bayesian estimator of a function f of model parameters $\theta = (p_1, p_2, \ldots, p_n)$ given an observation S is the expectation of this function given the (known) prior distribution $g(\theta)$. That is,

$$\hat{f} = \frac{\int f(\theta) P(S \mid \theta) g(\theta) d\theta}{\int P(S \mid \theta) g(\theta) d\theta}$$

This estimator minimizes the mean squared error risk associated with estimating f. Choosing a uniform distribution for g is often done to simplify the form of this estimator. Even when this choice is not justified, the resulting estimator has desirable properties.

The process of deriving the Bayesian estimator for the general case of an order k Markov model is well illustrated by the simpler case of estimating the parameters of a trinomial distribution which generates a particular string $S = (s_1 \ldots s_n)$. In this case $\theta = (p_0, p_1, p_2)$ and

$$P(S \mid \theta) = \prod_{i=1}^{n} p_{s_i}$$

$$P(S \mid \theta) = p_0^{k_0} p_1^{k_1} p_2^{k_2}$$

Letting $\Delta = \delta(p_0 + p_1 + p_2 - 1)$ and

$$\Theta = \begin{cases} 1 & \text{if } p_0 \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

The Bayesian estimator of p_0 is S_1/S_0 where

$$S_0 = \int P(S \mid \theta) \Delta \Theta d\theta$$

Starting with the simpler S_0,

$$S_0 = \int_0^1 dp_0 \int_0^{1-p_0} dp_1 \int_0^{1-p_0-p_1} dp_2 \left[p_0^{k_0} p_1^{k_1} p_2^{k_2} \right]$$

$$S_0 = \int_0^1 dp_0 \left[p_0^{k_0} \int_0^{1-p_0} dp_1 \left[p_1^{k_1} \int_0^{1-p_0-p_1} dp_2 p_2^{k_2} \right] \right]$$

$$S_0 = \int_0^1 dp_0 \left[p_0^{k_0} \int_0^{1-p_0} dp_1 \frac{p_1^{k_1}(1-p_0-p_1)^{k_2+1}}{k_2+1} \right]$$

By using the change of variables $\tau_0 = 1 - p_0$, this becomes

$$S_0 = \frac{1}{k_2+1} \int_0^1 dp_0 \left[p_0^{k_0} \int_0^{\tau_0} dp_1 p_1^{k_1} (\tau_0 - p_1)^{k_2+1} \right]$$

This last expression is the convolution of $u(t)t^{k_1}$ and $u(t)t^{k_2+1}$. It can be evaluated by taking the Laplace transform of each of these functions and multiplying (see notes on short lemmas). This gives

$$S_0 = \frac{1}{k_2+1} \int_0^1 dp_0 p_0^{k_0} \left[\mathcal{L}^{-1} \left(\frac{\Gamma(k_1+1)}{s^{k_1+1}} \frac{\Gamma(k_2+2)}{s^{k_2+2}} \right) \right]_0^{1-p_0}$$

$$S_0 = \frac{1}{k_2+1} \int_0^1 dp_0 \left[p_0^{k_0} \mathcal{L}^{-1} \left[\frac{\Gamma(k_1+1)\,\Gamma(k_2+2)}{s^{k_1+k_2+2}} \right]_0^{1-p_0} \right]$$

$$S_0 = \frac{\Gamma(k_1+1)\Gamma(k_2+2)}{\Gamma(k_1+k_2+2)(k_2+1)} \int_0^1 dp_0 \left[p_0^{k_0} (1-p_0)^{k_1+k_2+2} \right]$$

Another similar convolution step gives

$$S_0 = \frac{\Gamma(k_1+1)\Gamma(k_2+2)\Gamma(k_0+1)\Gamma(k_1+k_2+2)}{\Gamma(k_0+k_1+k_2+3)\Gamma(k_1+k_2+2)(k_2+1)}$$

Cancelling,

$$S_0 = \frac{\Gamma(k_0+1)\Gamma(k_1+1)\Gamma(k_2+1)}{\Gamma(k_0+k_1+k_2+3)}$$

Similarly,

$$S_1 = \frac{\Gamma(k_0+2)\Gamma(k_1+1)\Gamma(k_2+1)}{\Gamma(k_0+k_1+k_2+4)}$$

Thus

$$\frac{S_1}{S_0} = \frac{k_0 + 1}{k_0 + k_1 + k_2 + 3}$$

Similar use of a change of variable, the Laplace transform, and the convolution theorem allow this method to be extended to the general case of the multinomial as well as the slightly more complex case of the Markov model of any order. Symmetry allows the evaluation of the integrals to obtain an estimator for any of the p_i.

The calculation of the estimator for $\log P(S)$ uses the fact that $\log\ P(S)$ can be decomposed into a sum with terms which involve logarithms of exactly one of the model parameters. By arranging the integrals to save the log term until the outermost integration, the same changes of variable and applications of the convolution theorem can be used to evaluate all but the outermost of the integrals. This last integral is of the form

$$\int_0^1 t^n (1-t)^m \log t \, dt$$

which can be integrated by observing that it, too, is a convolution which can be separated using the Laplace transform.

For the multinomial case, the Bayesian estimator for $f = \log P(S)$ with uniform prior is given by

$$\hat{f} = \sum_{i=1}^{n} k_i \left(\psi(k_i + 2) - \psi(N + n + 1)\right)$$

where $N = \sum_{i=1}^{n} k_i$.

It does not suffice, in general, when computing the Bayesian estimator for $f(p_0 \ldots p_n)$ to simply estimate each of the p_i and then compute f from these. For example, to estimate $p_0 p_1$ in the binomial case,

$$S_0 = \frac{\Gamma(k_0 + 1)\Gamma(k_1 + 1)}{\Gamma(k_0 + k_1 + 2)}$$

as before, and

$$S_1 = \int dp_0 \int dp_1 p_0^{k_0+1} p_1^{k_1+1}$$

$$S_1 = \frac{\Gamma(k_0+2)\Gamma(k_1+2)}{\Gamma(k_0+k_1+4)}$$

So,

$$\hat{f} = \frac{(k_0+1)(k_1+1)}{(k_0+k_1+2)(k_0+k_1+3)}$$

If we had just estimated p_0 and p_1 and multiplied them, the denominator here would have been $(k_0+k_1+2)^2$.

Unfortunately, it is often difficult to derive the exact form of the Bayesian estimator. In such cases, taking the easy way out by using a function of estimators which are simpler to derive may give relatively good practical results.

Short Lemmas

The following definitions and simple lemmas are useful in following the derivations of the formulae in this paper. For the most part, they are given with only a hint at their derivation.

The Γ function is a continuous generalization of the factorial function. It is typically defined by the following

$$\Gamma(n) = \int_0^\infty t^{n-1} e^{-t} dt$$

A handy thing about the Γ function is that the derivative gives us a useful hook into an otherwise horrid integral

$$\Gamma'(n) = \frac{d \int_0^\infty t^{n-1} e^{-t} dt}{dn} = \int_0^\infty t^{n-1} e^{-t} \log t dt$$

Given the definition of the Γ function, the Laplace transform of t^n is straightforward

$$\mathcal{L}(t^n) = \frac{\Gamma(n+1)}{s^{n+1}}$$

And given the derivative trick above, the transforms of $\log t$ and $t^n \log t$ are also easy

$$\mathcal{L}(\log t) = \frac{(\gamma - \log s)}{s}$$

$$\mathcal{L}(t^n \log t) = \Gamma(n+1) \frac{(\psi(n+1) - \log s)}{s^{n+1}}$$

and the inverse transforms are then

$$\mathcal{L}^{-1}\left(\frac{\log s}{s}\right) = \gamma - \log t$$

$$\mathcal{L}\left(\frac{\log s}{s^n}\right) = t^{n+1}\left(\frac{\Gamma'(n)}{\Gamma(n)^2} - \log t\right)$$

The ψ function is a abbreviation

$$\psi(n) = \frac{d \log \Gamma(n)}{dn}$$

Largely, this is because this ratio pops up so often

$$\psi(n) = \frac{\Gamma'(n)}{\Gamma(n)}$$

The harmonic series, log, γ and ψ are closely related

$$\psi(n) = \psi(n-1) + \frac{1}{n-1}$$

Convolution is defined as a linear integral operator on two functions

$$(f * g)(t) = \int_{-\infty}^{\infty} f(x) g(t-x) dx$$

and Heaviside's unit impulse function,

$$u(t) = \begin{cases} 0 & \text{if } t < 0 \\ 1 & \text{if } t >= 0 \end{cases}$$

allows conversion of indefinite integrals into definite integrals

$$(uf * ug)(t) = \int_0^t f(x) g(t-x) dx$$

The most important aspect of convolution for this paper is its relation to the Laplace transform

$$\mathcal{L}(uf * ug) = \mathcal{L}(f)\mathcal{L}(g)$$

Bibliography

[BKP92] V. Batagelj, D. Kerzic, and T. Pisanski. Automatic clustering of languages. *Computational Linguistics*, 18(3), 1992.

[CHJS94] Gavin Churcher, Judith Hayes, Stephen Johnson, and Clive Souter. Bigraph and trigraph models for language identification and character recognition. In *Proceedings of 1994 AISB Workshop on Computational Linguistics for Speech and Handwriting Recognition, University of Leeds*, 1994.

[Chu94] Gavin Churcher. Distinctive character sequences, 1994. personal communication.

[CT94] William B. Cavnar and John M. Trenkle. N-gram-based text categorization. In *1994 Symposium on Document Analysis and Information Retrieval in Las Vegas*, 1994.

[edi93] editor. News and commentary. *Science*, March 19 1993.

[Hay93] Judith Hayes. Language recognition using two-and three-letter clusters. Technical report, School of Computer Studies, University of Leeds, 1993.

[Joh93] Stephen Johnson. Solving the problem of language recognition. Technical report, School of Computer Studies, University of Leeds, 1993.

[WDS+92] Owen White, Ted Dunning, Granger Sutton, Mark Adams, J.Craig Venter, and Chris Fields. A quality control algorithm for dna sequencing projects. *Nucleic Acids Research*, 21(16), 1992.

[WW92] D. H. Wolpert and D.R. Wolf. Estimating functions of probability distributions from a finite set of samples, part 1: Bayes estimators and the shannon entropy. Technical Report LA-UR-92-4369, Los Alamos National Laboratory, 1992.

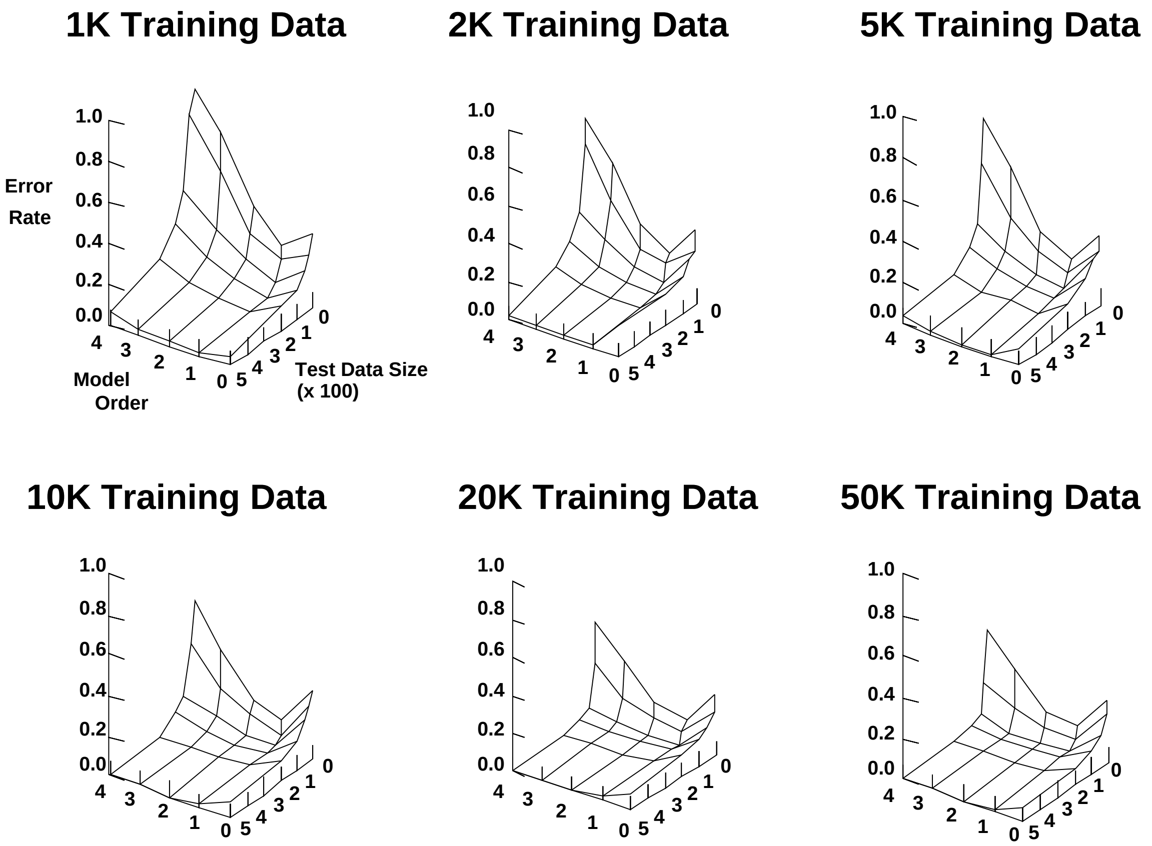

Figure 1

For any given combination of test string size and training set size, there is an optimum order for the language model. In all cases, longer test strings and more training data improve error performance.

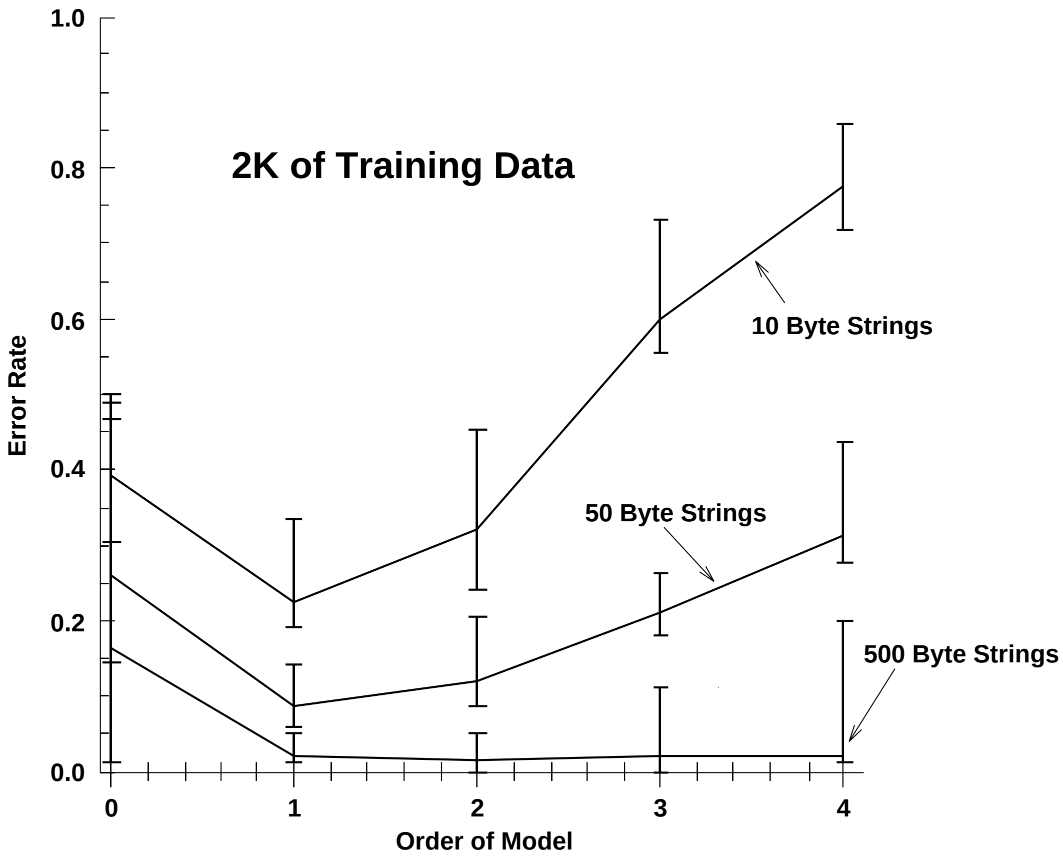

Figure 2

For 2K of training data, test strings which are 10 and 50 bytes long are best classified by an order 1 model, but 500 byte test strings can be classified by any model with order 1 or 2. Note that with such a short training set, even the optimum classifier does not work very well when classifying short strings. Also note that while the median performance of order 3, 4, and 5 models is about the same as the order 1 and 2 models, the bounds on performance are considerably looser. Error rates exceed 0.5 due to experimental design which penalizes guessing.

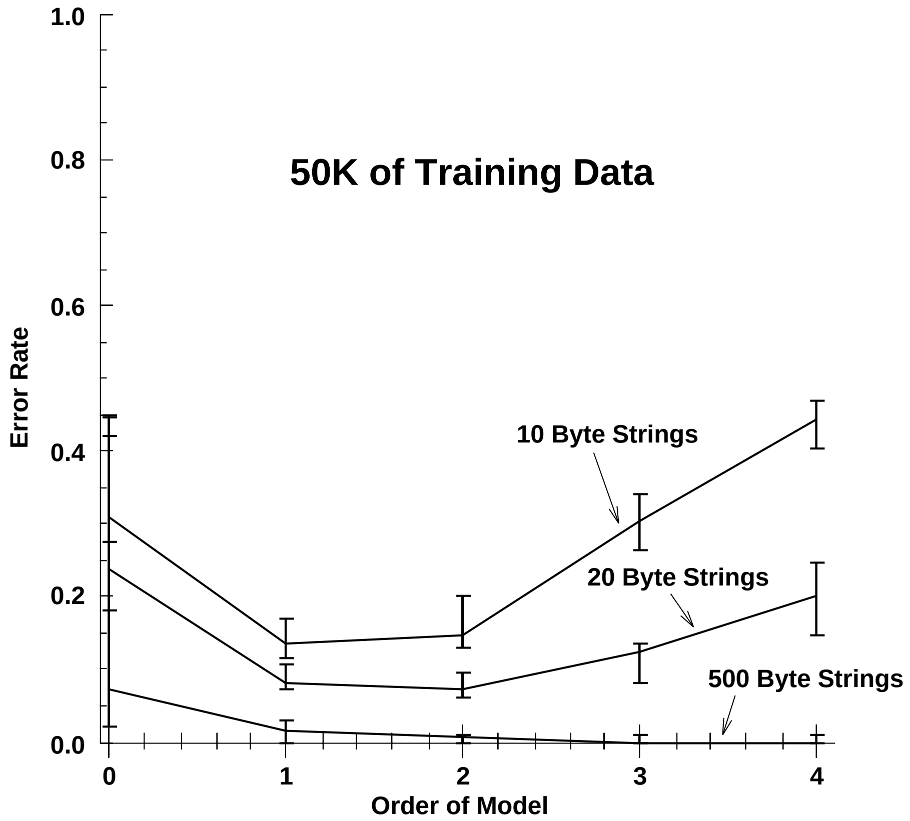

Figure 3

With 50K of training data, 10 and 20 byte test strings are well classified by models of order 1 and 2. Test strings of length 500 are classified extremely well with models of order 2 and above. Note the very tight bounds on performance for classifying 500 byte strings with the higher order models and the very wide bounds on error rate with the order 0 model even when applied to long test strings.