AUTOMATIC EXTRACTION OF SYNTAX APPLIED TO SPEECH RECOGNITION.

M.D. Alder

Department of Mathematics
University of Western Australia

ABSTRACT - The successful recognition of speech depends heavily upon the use of contextual information. Moreover the contextual information is too extensive to be put in by hand; this has led to attempts to automate the process. At the level of phonemic data, *hidden markov models* are extensively used, while at the lexical level the principal method is that of *n-grams*. (F. Jelinek , 1985)
The n-gram method has two related problems associated with it. The first is that there are a very large number of n-grams of consecutive words in English text for n greater than one, and the number goes up very quickly with n. The second is that even this number is sparse in real data, so that even with 'training sets' of millions of words of text, any new text contains a large fraction of n-grams never seen before. And this fraction also increases rapidly with n. The question of what to do in this case is a central problem for this and other classes of stochastic grammars.

In this paper I describe algorithms which address both problems. The first issue, that of storing large numbers of n-grams, is treated by storing not the n-grams themselves but classes of n-grams which are 'close together'. The second issue, that of sparseness of data, is solved by a derived method: we average over a neighbourhood of n-grams.

The algorithms are computationally intensive, but are amenable to parallelisation. There are implications for layered neural networks.

INTRODUCTION

The IBM *Tangora* Automatic Speech Recognition system can correctly transcribe from speech to English text a sentence such as *Mrs. Wright has no right to write about the sacred rites.* The mechanism for accomplishing the recognition of four distinct homophones is the application of a stochastic grammar, an algorithm which can consider a string of written English words and estimate the probability of its occurrence in text. Recurrence of four homophones in the same sentence is rare, but the necessity for using syntactic information has been clear since HARPY. Stochastic Grammars are used at several different levels: they are applied at the level of strings of phonemes, to strings of words, and lately to strings of symbols representing prosodic information. The same considerations indeed apply to related problems: OCR systems need to use syntactic information to discriminate between characters which, particularly in the present font, are optically indistinguishable. In general, we adopt an essentially Bayesian approach: we suppose that the syntax of the object language imposes *a priori* constraints on the classification of the acoustic signal, we articulate these constraints as a prior probability distribution on the strings, and we attempt to maximise the *a posteriori* probability of a transcription of a particular acoustic stream. This approach is elegantly described in F. Jelinek, 1985, where the two currently used stochastic grammars are described: Hidden Markov Models which are applied to the case where the symbols are phonemic elements of Speech, and n-grams which are applied to the case where the symbols are words in the object language.

The problem then becomes one of extracting this syntactic information automatically. This is the problem of stochastic grammatical inference; it is intimately related to the problem of data compression, and indeed may also be regarded as a paradigm of learning. The attraction of the two grammar classes which have been mentioned is that the inference problem is relatively straightforward; there are significant problems however. In this paper I shall consider only the n-gram grammars. I shall discuss the practical difficulties of inference and describe algorithms for overcoming these difficulties. Finally, the observation that human beings perform the process of grammatical inference naturally leads to considerations of implementation of these algorithms by neural nets.

THE PROBLEMS WITH N-GRAMS.

The n-gram methods used so far in application to the case where the symbols are words of the (written) English Language restrict n to the value 3. The trigram model for English text is precisely the third order Markov model approximation to English which is described by Shannon and Weaver (1949) and it may be explored by hand to a limited extent. It is not very good. The procedure is simple: we go through some set of English text, the *training set*, and we list all trigrams, i.e. triples of consecutive words. When two trigrams have the same first two words, we store them together and keep a count of the final words. I shall call the resulting objects (3,3) contexts, where the first digit tells you that we are dealing with trigrams, and the second digit tells you that we have a 'wildcard' in the third place, and we store the *distribution* of symbols which are associated with that wildcard in that context. The distribution is merely the set of possible final words together with their counts.

Having stored all the (3,3) contexts which are to be found in the text together with their distributions, we have a stochastic grammar for the text. We apply it to the problem of predicting a new text, the test set, by starting with the first two symbols of the new text as given, and matching it with the first two symbols of a context stored in our stochastic grammar. Having made the match, we now have a distribution of possible values for the third symbol, and the best guess is evidently the most common of these. Having made our guess, we can then move along one symbol, confirm or refute our prediction, and try again on the next symbol.

The simplicity of the procedure as described is confronted immediately with a problem. What do we do if there is no match? If our first two symbols of the text have never been seen before in that conjunction, we have to estimate a conditional probability of the third given the preceding pair, and our information is that the preceding pair never happens. But it just did! The methods employed at Yorktown Heights are two: the fall back method and the smoothed fallback method. The first of these simply falls back from trigrams to bigrams. Instead of using the two preceding symbols, we use only the immediately preceding symbol. It may happen that this has never been seen before either- a new word, not in the vocabulary of the system. In this event we fall back even further and use the frequency of occurrence of words in the text as our only guide. The smoothed falback method always uses a linear combination of trigram, bigram and 'unigram' statistics to make the prediction. The two methods work about equally badly. There is a rationale for each approach which is ingenious but not altogether convincing.

The problem at first appears to be merely an irritation caused by an insufficiently large training set, but numerical investigations make it plain that this is not the case. For n= 3, trigrams, texts containing billions of words are readily seen to be necessary to contain the problem, and it is doubtful whether such texts are really statistically homogeneous. Moreover, n= 3 is unrealistically small and the problem is not containable for larger n even supposing we had a text containing all the English words ever written. Even the strong syntactic constraints of formal papers are not enough to guarantee that a single paper presented at this conference will not contain a string of thirty eight consecutive words which have never occurred before in history. For thirty eight, perhaps read six. Human beings however learn to extract syntax on very much smaller training sets, and it is of some interest to consider how they might do it.

There are two other problems which also occur. The first is how to store n-grams or contexts when there are such enormous numbers of them. The second is how to evaluate a predictor and decide when one is better than another. Both appear deceptively simple, and neither are so in fact. Both appear to be irritating accidents, peripheral to the main issue. All the problems mentioned here are related to each other and are deeper than they at first appear. Classical statistics offers no help; the best unbiassed estimator for the probabilities of a sequent to two given words is the simple counting procedure without fallback. Use of Stein estimators gives a rationale for the fallback methods which do not work too well. Human beings however manage to do the job on much smaller data sets. How?

SMOOTHING OF SYMBOLIC TIME SERIES

Suppose you are reading a piece of English text and you read a borrim which you have never seen before. What do you do? You conjecture that it is either a misprint or a new word outside your present vocabulary and you conjecture further that it is a synonym for some other word you do know, or that it is close to, in meaning, some collection of words you have met before. You therefore replace the new word with a synonym. How do you choose the synonym without a dictionary? You use the context to tell you what the synonym must be. In the first sentence of this page, you may or may not have been stopped by the peculiar *borrim*, but you almost certainly decided on contextual grounds that it could be replaced by *word* without significant loss of information. If you had to predict the word following *borrim* , your chances of predicting *which* are very much better than the trigram model which would predict *the* or *a* as the most likely choice.

We can reasonably refer to this process as *smoothing* the text, a term derived from the analogous problem for sequences of Real numbers, or *discrete time series* as they are known to Statisticians and Engineers. It is in fact productive to regard a text as a finite sample of a time series of symbols from a finite alphabet, and to look upon this whole task of extracting sytactic information as being formally related to the theory of time series. The point is made in Alder, 1986.

We can therefore proceed to modify the trigram predictor so as to improve on the fallback method for coping with a new word as follows: given the string of symbols ABCD and wishing to predict the successor to D which has never been seen before, which I shall refer to as computing the probability distribution for ABCD?, we first find all possible values for the wildcard, ?, in ABC?. If x occurs with probability p(x), I now compute ABCx? and weight the resulting distribution by p(x). Finally I sum over all such x in ABC? .

I can do better than this; I can also consider the possible y in AB?C with probability p(y), and sum over all such weighted y the probabilities AByC?, and indeed also the probabilitiesAzBC? weighted by p(z) for z in A?BC. I might consider even broader neighbourhoods of the original ABCD by having more than one wildcard. It is easy to persuade oneself that this method gives better results than the fallback method when applied to English text. Numerical verifications of this impression are hard to come by because the problem is extremely intensive in computer time, and I am not well equipped in this regard. Confirmatory data in a rather restricted case will be published elsewhere.

It is clear that this method for coping with missing trigrams requires that we store contexts and their distributions which are (3,k) contexts for values of k other than 3, and moreover that the method would work better for (n,k) contetxs for k taking all values between 2 and n, with n as large as possible. Thus we have at first sight produced a natural seeming solution to the problem of missing trigrams but at the expense of even greater storage requirements. We appear to need to store a set of (n,k) contexts, and for each context an associated distribution of possible values for the wildcard in the k^{th} position. For trigrams this amounts to precisely twice as much storage required.

TOPOLOGICAL METHODS OF DATA COMPRESSION

Things are not as bad as they appear. First, it might be that the distributions associated with the contexts are not significantly different from each other, in which case it makes sense to pool them. Second, we are in effect replacing an n-gram by a context, which may be regarded as a set of neighbouring n-grams. It makes sense to try to store neighbourhoods instead of n-grams. To be persuaded of the soundness of the former strategy, consider the introductory sentence which *Tangora* can handle: *Mrs. Wright has no right to write about the sacred rites.* The second word is a proper name and proper names usually do follow titles like *Mrs.* and often precede verbs. It is easy to see that the practice of pooling symbols which occur after a designator like *Mrs.* and before a variety of verbs will increase our stock of proper names. Throwing in other designators will do the same. So pooling distributions appears to be a rational act given some obvious properties of natural language. In short, the algorithm I have suggested would be able to assign a high probability to *Wright* as the second word as a result of having met it only in the context

Professor Wright said.... or even *The Wright brothers were...* Again, this is something human beings seem to be able to accomplish. It is of some interest that this algorithm produces grammatical categories in natural language. For if one considers the possible substitutions of words in such a context as *The ? sat on the mat*, it is clear that the distribution contains precisely all the things that can sit on mats, together with their relative likelihood of doing so. If one pools with the similar distribution of all the things that, say, drink milk, arising from a distribution occurring in other contexts, then one builds up a distrbution which looks like the category of animals. Further somewhat indiscriminate pooling yields traditional grammatical categories such as nouns, at least to a statistical approximation. It appears, in short, that such categories arise naturally out of a particular algorithm for doing stochastic grammatical inference on natural language text, and the fact that we find such categories in natural language suggests that the algorithm employed by the brain to do the task is similar to the one described.

The second matter, of storing not n-grams, nor contexts, but neighbourhoods of contexts, further reduces the amount of storage, and further introduces topological ideas into an apparently non-topological setting. We arrive at the notion of a context distribution as an attempt to describe how human beings solve the problem of coping with sparse data. It leads to the representation of two n-grams which differ only in a single symbol as being neighbours of each other, and also of all the symbols that can occur in a distribution as being close- with respect to some contexts. I can further regard two contexts as being close together if their associated distribution are, even when the contexts themselves have no symbols in common. For example, *Mrs. ? said* and *Dr. ? gave* are neighbours by virtue of the discovery that the distribution of possible values for the wildcards are close together. There is of course a variety of senses in which two probability distributions can be said to be close (the Euclidean metric, χ^2, the Kolmogorov metric and others) but they are asymptotically equivalent with only modest hypotheses. If there were a practical way of storing the knowledge that the above contexts are neighbours in this sense, then we would not have to store anything like the same number of trigrams, and indeed would be able to store (n,k) context equivalence classes for larger n. It might seem impossible to do this at first sight; it seems that one has to be able to recognise that one has seen each context before, and that, it might be thought, requires that one keeps it in a list of 'contexts which have been encountered'. This is not, of course, the case; one needs only an adaptive hash-coding system. Such systems may be implemented by neural nets. It seems unlikely that this is a coincidence.

IMPLEMENTATION OF THE ALGORITHMS- THE DIRECT APPROACH

The first algorithm requires only that one reads a string of length n, replaces every symbol by a wildcard, stores the resulting (n,k) context and starts a set of distributions, one for each k, by inserting each symbol with a count of 1. One then proceeds to read the next symbol and repeats, augmenting the distribution counts accordingly. Each context points to its associated distribution which is kept in a separate dictionary of distributions. Having gone through the entire training text in this way, one then proceeds to pool the distributions, shrinking the size of the dictionary of distributions, and pointing now from each context to the enlarged distribution in the new dictionary. The second algorithm now treats the contexts as points in a space, and the addresses of the different distributions as a finite list of 'types' associated with them. This is now a standard pattern classification problem. We can take a neural net and train it so that when it receives a context, it computes the address of the cluster associated with it. The resulting object is a topological local stochastic grammar. (Local because n-grams give a local grammar). It is applied to new text in the way described above.

The description in the preceding paragraph is actually workable for small alphabets and has been implemented in the case where instead of words of English text, we deal only with letters. Even here, with a vocabulary of symbols of only a hundred or so, the computations are considerably beyond the convenient. It has been necessary to prune and oversimplify, and even so the pooling procedure alone takes weeks of runtime on a microVAX. The scaling up to a vocabulary of 20,000 words requires supercomputing powers and enormous amounts of memory and still takes very long times. Naive estimates of the computing power of the human brain are apt to be revised upward in spectacular fashion when confronted by this problem, a problem which the brain itself

solves relatively quickly.

IMPLEMENTATION OF THE ALGORITHMS VIA PARALLEL PROCESSING.

I have described elsewhere (Alder 1986,1988) how a neurally inspired algorithm can accomplish the same tasks as those described above, and which has the merit of being susceptible of implementation on a parallel processing system. The idea is to embed the space of contexts in a metric space and to model the process of automatic tuning of neurons known to occur in the visual cortex (Blakemore,1975).

It would be pleasing to be able to report that this has been implemented and found to work as expected, but a parallel program written in MODULA2 running on a microVAX runs even more slowly than the non-parallel version described above. For want of adequate computing power, it has not proved possible to test out the algorithms on other than essentially toy problems. It should be remarked that they do behave as expected in these cases, and that the scaling up process is feasible, merely expensive.

ACKNOWLEDGEMENTS

I am grateful to Fred Jelinek and the other members of the Continuous Speech Recognition Group for putting up with me for six months in 1985, also to IBM for the financial support.

REFERENCES

Alder, M.D. (1986) *Symbolic Time Series*, Mathematics Department Research Report, University of Western Australia , Nedlands.W.A. 6009.

Alder, M.D. (1988) *Stochastic Grammatical Inference Applied to Automatic Speech Recognition,* Mathematics Department Research Report, University of Western Australia, Nedlands,W.A. 6009.

Blakemore, C.(1975) *Developmental Factors in the Formation of Feature Extracting Neurons*, in *Feature Extraction by Neurons and Behaviour,* ed. Gerhard Werner, MIT 1975.pp 105,113.

Jelinek,F. (1985) *Self-Organised Language Modeling for Speech Recognition*, IBM Research Report, Continuous Speech Recognition Group, IBM T.J. Watson Research Center, Yorktown Heights, N.Y. 10598.

Shannon, C.E. and Weaver, W *(1949) The Mathematical Theory of Communication.* University of Illinois Press, Urbana, 1949.