Grammatical Inference
What I Think

Jason L Hutchens

May 11, 1996

Abstract

Natural Language Grammatical Inference is really hard work. Human beings have enough trouble working out how to use language, so getting a computer to perform anywhere near as well as a human being is difficult, if not impossible.

That said, it is still a useful area of research. In this technical report, written mainly to crystallize my ideas and let my supervisor know that I am still alive, I simply step through all of my thoughts on the subject and speculate about what I will be doing in the near future.

1 Introduction

This paper is what you get when you sit at the keyboard and pour your brain out. I have basically ad-libbed it all the way, without too much concern for organisation. That means that this paper has followed my line of though quite closely. I hope that this doesn't confuse the reader too much. There are no nasty equations or anything like that, as I don't think they're necessary to get my thoughts and ideas across.

2 The Bootstrapping Principle

Whenever we try to write a computer program to solve any sort of problem, we must create a model of that problem first. Models are simple enough for us to understand, while retaining enough information about the original problem so as not to render them worthless.

To recognise Kanji characters, for example, we create a model of each of the characters that we wish to recognise. Such models are often nothing more than feature vectors, but they needn't be. To recognise a new character, we simply compare its model to all the ones we know about, and find the one which is in some sense the most similar.

Using feature vectors to distinguish between Kanji characters often works surprisingly well. The features often used are rather naive, and include such things as counting the number of border-crossings on a horizontal scan of the character. We would like to believe that better results could be attained if we could somehow write a program which would find the model automatically.

One way of doing this is to develop some sort of model for the image, and then use parameter estimation to optimise it for the image. For example, we may decide that a n^{th}-order polynomial is an adequate model. Parameter estimation would then be used to find the most appropriate n, and the coefficients of the model. A similar method would be used if we were modelling the data with n^{th}-order moments.

Perhaps a better way is to use *the bootstrapping principle*. We begin by choosing the simplest model we can dream up for the data we have got, and we don't worry too much about how well it models the data. Once we have the model, we can use it to find structure within the data. This structure may then be used to augment the model, creating a new model. This process may be iterated until some stopping criteria is satisfied.

If we are trying to find a grammar for Natural Language text, a good model to begin with would be the n-gram model operating on the character level. This models the language by estimating probability distributions over the alphabet of ASCII characters which are conditional on the preceding n characters. Such a model can be used to find the three most common forms of structure in language.

The model needn't be stochastic, though. I use stochastic language models for the following reasons.

1. Most work in Grammatical Inference is done with such models, and well defined performance measures exist for such models.

2. Being able to predict symbols in the language as a probability distribution over all symbols is useful in some applications, such as speech recognition.

3. It is easier to use such models to find structure within the language, as I will describe later in this report.

The process described is generally known as the *upwrite* process, as lower-level structure is "upwritten" to higher-level structure. Each time we perform an upwrite, we are sacrificing some information in order to generalise our model. Hopefully most of the information we lose is due to noise or aberrations within the data.

The upwrite process does create a hierarchical model of the data, though. When we use this model, we can select the granularity that we desire. For example, we may use the highest-level model (ie: the most abstract model) when performing Kanji character recognition. However, if we are unable to make a confident decision, we may drop down a level and try again, taking more information about the data into account in the process.

Another method is to make decisions at all levels of the model and amalgamate the results by taking a weighted average. This means that we have to do parameter estimation to find the best amalgamation weights, which seems a bit silly to me. The advantage is that the weights will be optimised to favour the level of the model which makes the best decisions *on average*. The disadvantage is that this means that worse decisions will invariably be made.

If we can do an upwrite, we can surely do the inverse operation known as the *downwrite*. That is, we can take our highest-level description of the data, and downwrite it level-by-level to get the lowest-level description. This will not be an exact copy of the data we used to create the model, but we would be worried if it was nothing like it. In fact, the downwrite produces a family of data, one of which will be an exact copy of our original data, and all of which will produce similar models when upwritten. Each data in the family will have a probability of occurence associated with it if the model is stochastic, and these probabilities may be used to take a weighted average of each data in the family.

I believe that we should write programs to do these downwrites as a way of evaluating the integrity of our models. For example, I do not think that the highest-level description of a Kanji character would downwrite to anything recognisable to a native writer of Kanji characters. This would mean that too much information was discarded in the modelling process. One way of automatically verifying the model would be to use it to calculate the entropy of the data it is modelling. High entropy indicates that the too much information was discarded in the name of generalisation. This sort of calculation is easy with stochastic models - we simply need to find the probability of each element of our data being produced by the model.

The main problem with the hierarchical upwrite method is that the upwrite which is performed is chosen to be the one which provides the best local improvement in the model. It is doubtful that this will result in the best possible model, as the best upwrite at any stage depends on which upwrites were performed previously. This problem is unavoidable, however, as to find the optimal model we would have to explore the advantages of every possible sequence of upwrites which would be time intensive.

3 Syntactic Structure

I previously mentioned that natural language text contains three kinds of structure which may be found using stochastic language models. These structures are given below.

1. Sequences of symbols that occur quite frequently and in some sense represent a single concept. In language, for example, sequences of strings form words.
2. Several symbols may represent the same or similar concepts. At a high level in English, for example, words may be grouped into their parts-of-speech.
3. The same symbol may be repeated several times in a row. We see this type of recursion in English where an adjective can appear multiple times before a noun.

In the following sections, I attempt to demonstrate how each of these structures may be found in natural language text using a stochastic language model.

3.1 Chunks

When we find words in a string of characters, we simply look for characters which we recognise as symbol separators. That is, we find words by looking for whitespace and punctuation. However, we can find words in a segment of text which has all whitespace and punctuation removed, so we are obviously doing something more clever than looking only for symbol separators.

Often there are many possible segmentations of a piece of text. We decide which one is correct by firstly trying a tentative segmentation of the next word, then reading ahead to see whether that segmentation makes sense in the current context. If it doesn't, we try a different segmentation. We are using our ability to predict where the symbol separators should be, and which words are likely to follow the ones we have segmented so far.

Language models may be used to predict which symbols are likely to come next, so they can be used to segment text in a similar way. Because we use the resulting segmentation to augment the language model via an upwrite, we cannot improve the segmentation by testing to see whether it is grammatical or not. Once we have a complete model, though, we can segment new texts to make them grammatical wrt the model.

One way of finding chunks in a text is to use the text to infer an n-gram model on the letter level. We can then scan through the text, using the model to make predictions about which letters are likely to come next. If our predictions are confirmed, then we are probably in the middle of a word. If, however, our predictions prove to be erroneous, we have probably encountered a word boundary, and we can insert a separator symbol at that point.

This is due to the fact that the intra-word statistics are less uniform than the inter-word statistics. As the prediction is in the form of a probability distribution over the alphabet, we may calculate the entropy of the predictor at each stage. The entropy is a measure of uncertainty, so higher uncertainties mean that the predictor is more confident that a symbol separator must be inserted.

The only problem is that we must specify a threshold entropy above which the predictor will insert a symbol separator. We would like the predictor to find this threshoild entropy automatically, and we do this by using parameter estimation techniques. I discuss these techniques in the next section.

3.2 Parts-of-Speech

We tend to group words depending on their meaning or the context in which they can appear. Linguists have discovered a handful of rather coarse groupings which they call the parts-of-speech. Such groups include verbs, nouns, adjectives and so on. It is likely, though, that we use much finer groupings. For example, we may group words which refer to small furry creatures. Such groupings would certainly make our job of understanding what a sentence means rather easier.

When we encounter new words which we have never werkled before, we tentatively assign them to a group which we know about, based upon the context in which they were used. If we only see this word a few times in our life, we would probably forget it quite quickly. If, however, we encountered the word regularly, we may either reinforce our conviction that it belongs in a certain group, or we may change the group if new evidence convinces us that our original assignment was wrong.

In natural language, words can have multiple meanings. This means that one word may belong to several groups. Such words are clearly quite different symbolically, even though they look the same when we write them down.

Getting a computer to find logical groupings of words or other symbols automatically is no easy task. The most obvious way of doing so is to use pre-Chomskian distributional linguistics. Basically, we find context vectors for each word that we know about and perform clustering on them, grouping the words accordingly. The main problem with this method is that it finds exactly one grouping for each word, which is usually the strongest grouping.

We need a technique to find parts-of-speech automatically, and to find multiple parts-pf-speech per word, depending on the context. This means that each word may have many context vectors, one for each part-of-speech that it belongs to. Such a technique does not exist at the moment. We know that the context vector we do have is some linear interpolation of these separate context vectors, but how do we find those vectors?

One way of finding context-sensitive groupings is to cluster the distribution vectors, which you get straight from the n-gram model. This is in fact what I have done as part of my project. It does assign words to multiple classes, as required, and each class not only contains the words, but also contains their relative probabilities of occurring in that class. However, the disadvantages are legion. Most importantly the data is sparse, so not many vectors go into making each cluster. This means that we can't be too confident about the clusters found, and that many vectors will simply remain unclustered.

So, we either make a context vector for each word in the text, or we make a word vector for each context in the text. There are a lot more contexts than words, so the corresponding vectors will be much sparser. One technique gives us good part-of-speech classes, the other provides context-dependant classes. We need to find some sort of compromise between the two techniques.

Note that the context that we use comes straight from the language model. In the n-gram model, for instance, the context is the n preceding words.

3.3 Recursive Structures

4 Parameter Estimation

Describe why we wish to use parameter estimatrion, and where we use it.

Describe the common techniques, their advantages and disadvantages.

Describe what we are optimising!

ML vs MDL vs Bayesian etc.

5 Learning as Compression

How to choose the best upwrite

How to compress data
Lossy vs lossless compression
Images vs text
Infer vs gzip

6 Implementation Issues

Size of data
Sparsity of data
Requirements of corpus
Assumptions made
Using frequent words only
Other possible optimisations

7 Applications

7.1 Speech Recognition and Transcription

7.2 Machine Translation

7.3 Data Compression

7.4 Natural Language Interfaces

7.5 Knowledge Acquisition and Search Engines

7.6 Prediction and Smoothing of Time Series

7.7 Text Generation, Understanding and Summarisation

7.8 Syntactic Pattern Recognition

7.9 The Loebner Contest

8 Biological Language Acquisition

Why Chomskian linguists and Pinker piss me off

9 Back to the Future

What I will do in the future

10 Literature Review

Write reviews of all papers and books

10.1 Syntactic Language Learning

11 Journals

Write reviews of all journals

11.1 Computational Linguistics

12 Conferences

Write on all conferences

12.1 ICGI

13 Conclusions

Conclusions

References

[1] Michael D. Alder. Principles of pattern classification: Statistical, neural net and syntactic methods of getting robots to see and hear. Available at ftp://ciips.ee.uwa.edu.au/pub/syntactic/book, February 1994.

[2] Mike Alder. Stochastic grammatical inference. Master's thesis, University of Western Australia, Nedlands, W.A. 6009, December 1988.

[3] D. Angluin and C.H. Smith. Inductive inference: Theory and methods. *ACM Computing Surveys*, 6:237–369, 1983.

[4] A. Averbuch, L. Bahl, R. Bakis, P. Brown, A. Cole, G. Daggett, S. Das, K. Davies, S. Degennaro, P. de Souza, E. Epstein, D. Fraleigh, F. Jelinek, S. Katz, B. Lewis, R. Mercer, A. Nadas, D. Nahamoo, M. Picheny, G. Schichman, and P. Spinelli. An IBM PC based large-vocabulary isolated-utterance speech recognizer. Technical report, IBM T.J. Watson Research Center, January 1986.

[5] A. Averbuch, L. Bahl, R. Bakis, P. Brown, G. Daggett, S. Das, K. Davies, S. Degennaro, P. de Souza, E. Epstein, D. Fraleigh, F. Jelinek, B. Lewis, R. Mercer, J. Moorhead, A. Nadas, D. Nahamoo, M. Picheny, G. Schichman, P. Spinelli, D. Van Compernolle, and H. Wilkens. Experiments with the Tangora 20,000 word speech recognizer. Technical report, IBM T.J. Watson Research Center, 0000.

[6] L. Bahl, R. Bakis, J. Bellegarda, P. Brown, D. Burshtein, S. Das, P. de Souza, P. Gopalakrishnan, F. Jelinek, D. Kanevsky, R. Mercer, A. Nadas, D. Nahamoo, and M. Picheny. Large vocabulary natural language continuous speech recognition. Technical report, IBM T.J. Watson Research Center, 0000.

[7] L. Bahl, R. Bakis, P. de Souza, and R. Mercer. Polling: A quick way to obtain a short list of candidate words in speech recognition. Technical report, IBM T.J. Watson Research Center, September 1987.

[8] L. Bahl, P. Brown, P. de Souza, and R. Mercer. A tree-based statistical language model for natural language speech recognition. Technical report, IBM T.J. Watson Research Center, September 1987.

[9] L. Bahl, P. Brown, P. de Souza, R. Mercer, and M. Picheny. A method for the construction of acoustic Markov models for words. Technical report, IBM T.J. Watson Research Center, September 1987.

[10] L. Bahl, F. Jelinek, and R. Mercer. A maximum likelihood approach to continuous speech recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-5(2), March 1983.

[11] P. Brown, J. Cocke, S. Della Pietra, V. Della Pietra, F. Jelinek, J. Lafferty, R. Mercer, and P. Roossin. A statistical approach to machine translation. Technical report, IBM T.J. Watson Research Center, October 1988.

[12] P. Brown, J. Cocke, S. Della Pietra, V. Della Pietra, F. Jelinek, R. Mercer, and P. Roossin. A statistical approach to French/English translation. Technical report, IBM T.J. Watson Research Center, 0000.

[13] P. Brown, F. Jelenik, and R. Mercer. Basic methods of probabilistic context free grammars. Technical report, IBM T.J. Watson Research Center, 0000.

[14] Eugene Charniak. *Statistical Language Learning*. MIT Press, 1993.

[15] Nick Chater and Martin Redington Stephen Finch. Acquiring syntactic information from distributional statistics. Available at ftp://oliphant.cogsci.edinburgh.ac.uk/pub/frankred.

[16] Nick Chater and Martin Redington Stephen Finch. The guessing game: A paradigm for artificial grammar learning. Available at ftp://oliphant.cogsci.edinburgh.ac.uk/pub/frankred.

[17] Nick Chater and Martin Redington Stephen Finch. The potential contribution of distributional information to early syntactic category acquisition. Available at ftp://oliphant.cogsci.edinburgh.ac.uk/pub/frankred.

[18] Noam Chomsky. Aspects of the theory of syntax. Special Technical Report of the Research Laboratory of Electronics 11, The Massachusetts Institute of Technology, 1965.

[19] Noam Chomsky. *Syntactic Structures*. Mouton, 1975.

[20] Daniel I. A. Cohen. *Introduction to Computer Theory*. John Wiley & Sons, Inc., 1991.

[21] Jordan R. Cohen. Application of an auditory model to speech recognition. *Journal of the Acoustical Society of America*, 85(6), June 1989.

[22] Richard Timon Daly. *Applications of the Mathematical Theory of Linguistics*. Mouton & Co., 1974.

[23] Richard Dawkins. *The Blind Watchmaker*. Longman Scientific & Technical, 1986.

[24] Pierre A. Devijver. Statistical pattern recognition. In K.S. Fu, editor, *Applications of Pattern Recognition*, chapter 2, pages 15–36. CRC Press, Inc., 1982.

[25] Jeffrey L. Elman. Distributed representations, simple recurrent networks, and grammatical structure. *Machine Learning*, 7, 1991.

[26] Ute Essen and Volker Steinbiss. Cooccurrence smoothing for stochastic language modeling. In *ICASSP 1990 International Conference on Acoustics, Speech and Signal Processing*, pages 161–164, 1990.

[27] King-Sun Fu and Taylor L. Booth. Grammatical inference: Introduction and survey - part I. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(3), May 1986.

[28] King-Sun Fu and Taylor L. Booth. Grammatical inference: Introduction and survey - part II. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(3), May 1986.

[29] K.S. Fu. Syntactic pattern recognition. In K.S. Fu, editor, *Applications of Pattern Recognition*, chapter 3, pages 37–63. CRC Press, Inc., 1982.

[30] E.M. Gold. Language identification in the limit. *Information and Control*, 10:447–474, 1967.

[31] I.J. Good. A.M. Turing's statistical work in World War II. In J.L. Britton, editor, *Collected works of A.M. Turing : Pure mathematics*, pages 393–396. Elsevier Science Publishers B.V., 1992.

[32] Ralph Grishman. *Computational linguistics: An introduction*. Cambridge University Press, 1986.

[33] Z.S. Harris. Distributional structure. *Word*, 10:140–162, 1954.

[34] John A. Hartigan. *Clustering Algorithms*. John Wiley & Sons, 1975.

[35] Douglas R. Hofstadter and Daniel C. Dennett. *The Mind's I*. The Harvester Press Limited, 1981.

[36] John E. Hopcroft and Jeffrey D. Ullman. *Introduction to automata theory, languages, and computation*. Addison-Wesley Publishing Company, Inc., 1979.

[37] Jason L Hutchens. Natural language grammatical inference. Available at http://ciips.ee.uwa.edu.au/ hutch/Research.html.

[38] David J. Hutches. *Data Structures and Algorithms for the Efficient Representation and Retrieval of Incremental Lexical Information*. PhD thesis, University of California, San Diego, 1993.

[39] Ajay N. Jain and Alex H. Waibel. Robust connectionist parsing of spoken language. In *ICASSP 1990 International Conference on Acoustics, Speech and Signal Processing*, pages 593–596, 1990.

[40] Frederick Jelinek. Markov source modeling of text generation. Technical report, IBM T.J. Watson Research Center, 0000.

[41] Frederick Jelinek. Principles of lexical language modeling for speech recognition. Technical report, IBM T.J. Watson Research Center, 0000.

[42] Frederick Jelinek. Self-organized language modeling for speech recognition. Technical report, IBM T.J. Watson Research Center, 0000.

[43] Frederick Jelinek. The development of an experimental discrete dictation recognizer. *Proceedings of the IEEE*, 73(11), November 1985.

[44] Frederick Jelinek. Computation of the probability of initial substring generation by stochastic context free grammars. Technical report, IBM T.J. Watson Research Center, April 1990.

[45] G.J.F Jones, H. Lloyd-Thomas, and J.H. Wright. Adaptive statistical and grammar models of language for application to speech recognition. In *IEE Colloquim on Grammatical Inference : Theory, Applications and Alternatives*, pages 25/1–8, April 1993.

[46] Julian Kupiec. Robust part-of-speech tagging using a hidden Markov model. *Computer Speech and Language*, 6(3), July 1992.

[47] M. Kutas. Reading senseless sentences. *Science*, 207:203–205, 1980.

[48] S.M. Lucas. New directions in grammatical inference. In *IEE Colloquim on Grammatical Inference : Theory, Applications and Alternatives*, pages 1/1–7, April 1993.

[49] Patricia McKenzie and Michael Alder. Initialising the EM algorithm for use in gaussian mixture modelling. Available at http://ciips.ee.uwa.edu.au/pub/syntactic/papers.

[50] Laurent Miclet. Grammatical inference. In Horst Bunke and Alberto Sanfeliu, editors, *Syntactic and Structural Pattern Recognition*, chapter 9, pages 237–290. World Scientific Publishing Co. Pte. Ltd., 1990.

[51] G. Miller and N. Chomsky. Finitary models of language users. In R.D. Luce, R.R. Bush, and E. Galanter, editors, *Handbook of Mathematical Psychology II*, pages 421–464. Wiley, 1963.

[52] Marvin Minsky. *The Society of Mind*. William Heinemann Ltd., 1987.

[53] Steven Pinker. *The Language Instinct*. The Penguin Press, 1995.

[54] A. Radford. *Transformational Grammar, 2nd Ed.* Cambridge University Press, 1988.

[55] Allan Ramsay. Inference in language processing. In *IEE Colloquim on Grammatical Inference : Theory, Applications and Alternatives*, pages 5/1–3, April 1993.

[56] Hinrich Schütze. Part-of-speech induction from scratch. Available at ftp://csli.stanford.edu/pub/prosit/papers.

[57] Hinrich Schütze and Yoram Singer. Part-of-speech tagging using a variable memory Markov model. Available at ftp://csli.stanford.edu/pub/prosit/papers.

[58] C.E. Shannon. Prediction and entropy of printed English. *Bell Systems Technical Journal*, 30:50–64, 1951.

[59] Claude E. Shannon and Warren Weaver. *The Mathematical theory of Communication*. University of Illinois Press, 1949.

[60] John B. Smith. Computer criticism. In Jr. Walter A. Sedelow and Sally Yeates Sedelow, editors, *Computers in Language Research 2*, pages 25–59. Walter de Gruyter & Co., 1983.

[61] Raoul N. Smith. *Probabalistic Performance Models of Language*. Mouton & Co., 1973.

[62] V. Steinbiss, A. Noll, A. Paeseler, H. Ney, H. Bergmann, C. Dugast, H. H. Hamer, H. Piotrowski, H. Tomaschewski, and A. Zielinski. A 10,000 word continuous-speech recognition system. In *ICASSP 1990 International Conference on Acoustics, Speech and Signal Processing*, pages 57–60, 1990.

[63] Michael G. Thomason. Introduction and overview. In Horst Bunke and Alberto Sanfeliu, editors, *Syntactic and Structural Pattern Recognition*, chapter 1, pages 3–25. World Scientific Publishing Co. Pte. Ltd., 1990.

[64] Wen-Hsaing Tsai. Combining statistical and structural methods. In Horst Bunke and Alberto Sanfeliu, editors, *Syntactic and Structural Pattern Recognition*, chapter 12, pages 349–366. World Scientific Publishing Co. Pte. Ltd., 1990.

[65] J.P. Ueberla. Analysis of a simple bipos language model - attempt at a strategy to improve language models for speech recognition. In *IEE Colloquim on Grammatical Inference : Theory, Applications and Alternatives*, pages 10/1–8, April 1993.

[66] Julie A. van Dyke. A syntactic predictor to enhance communication for disabled users. Technical Report 92-03, University of Delaware, August 1991.

[67] Jr. Walter A. Sedelow and Sally Yeates Sedelow. Science and human language. In Jr. Walter A. Sedelow and Sally Yeates Sedelow, editors, *Computers in Language Research 2*, pages 1–23. Walter de Gruyter & Co., 1983.

[68] Satosi Watanabe. *Knowing and Guessing*. John Wiley & Sons, Inc., 1969.

[69] Warren Weaver. Translation. In William N. Locke and A. Donald Booth, editors, *Machine translation of languages : fourteen essays*. MIT Press, 1955.

[70] Yorick A. Wilks. Machine translation and the artificial intelligence paradigm of language processes. In Jr. Walter A. Sedelow and Sally Yeates Sedelow, editors, *Computers in Language Research 2*, pages 61–111. Walter de Gruyter & Co., 1983.

[71] Ave Wrigley. Parse tree n-grams for spoken language modeling. In *IEE Colloquim on Grammatical Inference : Theory, Applications and Alternatives*, pages 26/1–6, April 1993.