Language Acquisition and Data Compression

Jason L. Hutchens*and Michael D. Alder†

January 31, 1997

Abstract

Statistical data compression requires a stochastic language model which must rapidly adapt to new data as it is encountered. A grammatical inference engine is introduced which satisfies this requirement; it is able to discover structure in arbitrary data using nothing more than the predictions of a simple trigram model. We show that compression may be used as an alternative to perplexity for language model evaluation, and that the information processing techniques employed by our system may reflect what happens in the human brain.

1 Introduction

Grammatical inference is the process of programming a computer to automatically infer a grammar for a language [8]. We consider a grammar to be nothing more than a model for some data. Applications such as speech recognition and data compression require a stochastic language model, and well-defined performance measures exist for such models.

It is easy to get caught in the trap of building complicated models which utilise various *ad hoc* techniques in order to gain modest improvements in performance. Although this is excusable in many cases, we believe that simple models may offer insights into human language acquisition, and may be applicable to a wider range of problems due to their generality.

In this paper we introduce a new performance measure for stochastic language models, and then proceed to describe the upwrite predictor. Preliminary results are given, and we conclude with speculations as to where

*Jason L. Hutchens is with the Centre for Intelligent Information Processing Systems, The University of Western Australia, Nedlands W.A. 6907, Australia. E-mail: `hutch@ciips.ee.uwa.edu.au`

†Michael D. Alder is with the Department of Mathematics, The University of Western Australia, Nedlands W.A. 6907, Australia. E-mail: `mike@maths.uwa.edu.au`

this work will lead. Readers are advised to consult Charniak's book for a good introduction to grammatical inference [5], and the book by Bell, Cleary and Witten for an excellent overview of statistical text compression [4].

2 Performance Measures

We will concern ourselves with stochastic language models only; we define a stochastic language model as a model which, when presented with a string of symbols, is able to make a prediction about the next symbol in the form of a probability distribution over the alphabet of all symbols.

Information theory is used to evaluate such language models [14]. Historically, the perplexity of the model has been used as a performance measure. As we will show, this measure has some disadvantages, and for some applications an alternative may be useful.

2.1 Perplexity

When the IBM Speech Recognition group were developing a language model for their Tangora speech recognition system, the perplexity of the model with respect to the text was used as a performance measure [3]. This perplexity is defined in Equation 1, where the w_i are symbols in a string of length n, and it gives the average number of equi-probable symbols that the language model must choose between when making a prediction [5].

$$PP = P(w_1, w_2, \ldots, w_n)^{-\frac{1}{n}} \tag{1}$$

As Equation 2 shows, the perplexity may be rewritten as a monotone function of $\overline{I}$, the average information that the language model receives. This average information, given in Equation 3, is a measure of how well the language model is able to compress the text.

$$PP = 2^{\overline{I}} \tag{2}$$

$$\overline{I} = -\frac{1}{n}\sum_{i=1}^{n} \log_2 p(w_i | w_1, w_2, \ldots, w_{i-1}) \tag{3}$$

IBM's perplexity measure is therefore an indicator of how much compression the language model can achieve. This maximum likelihood measure disregards the complexity of the language model itself.[1] In accordance with

[1]For example, it neglects the overhead of compressing the alphabet, which may be large if the symbols are English words.

Occam's razor, we would like a measure that favours simple models which give good compression [9], and such a measure is easily obtained by using the language model as part of a real statistical compression system.

2.2 Statistical Data Compression

A data compression system maps a sequence of symbols into another representation in order to decrease the amount of storage space they require. To restore the original sequence, a decompresser is used to perform the inverse mapping.[2] Statistical compression systems use a stochastic language model when performing the mapping, and they come in three flavours; static, semi-adaptive and adaptive [4].

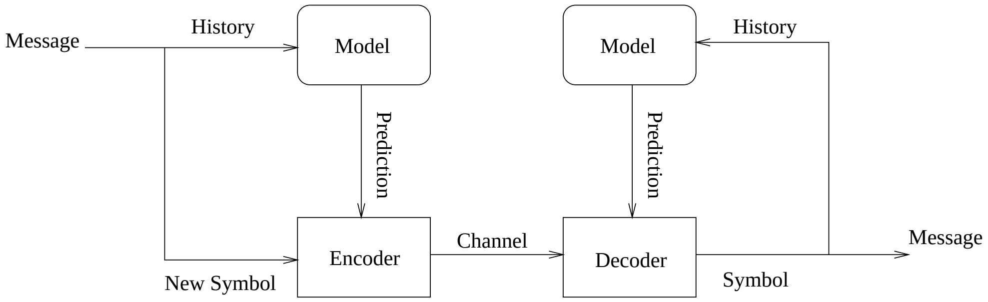

Figure 1: An adaptive statistical compression system

In an adaptive statistical compression system, the compressor is broken up into a model and an encoder, as in Figure 1. The data which is to be compressed is fed into the model one symbol at a time. The model, which is constructed adaptively from the history, is used to make a prediction about the current symbol. The encoder uses the probability thus assigned to the symbol to encode it for transmission. Arithmetic coding is employed to encode the data as optimally as possible [15].

Adaptive compression systems work because the decompresser is able to reconstruct the model from the encoded data in exactly the same way as it was constructed by the compressor. We have chosen to use an adaptive statistical compression scheme as a performance measure for our language model for the reasons outlined below:

1. The model is encoded implicitly in the compressed representation of the data, and this negates the need to encode and transmit it separately.

[2]Note that we consider non-lossy compression systems only in this paper.

2. No prior assumptions can be made about the data, unlike the perplexity measure which assumes the training and testing corpora will cover similar domains.

3. The model is required to be adaptive, which means the measure also takes into account how quickly the model adapts to changes in the data.

4. The perplexity of the model can be calculated easily, if required. The perplexity is a function of the average information, which may be approximated from the compression ratio.

3 Upwrite Predictor

The aim of our current project is to develop a stochastic language model which is generic enough to find structure in all sorts of one-dimensional data. To accomplish this, we need a model which satisfies the following requirements:

1. The model should make as few assumptions as possible about the data.

2. The only information available to the model should be the data itself.

3. The system should be able to incorporate any stochastic language model.

4. The predictions from the language model should be used to find structure within the data, and this structure should be used to bootstrap the model.

5. *Ad hoc* methods should be avoided at all costs.

The system we decided upon is shown in Figure 2. Raw binary data is fed into the system, and is broken up into discrete symbols by the parser. The parser uses a dictionary[3] which may be altered during processing, but which must always contain appropriate symbols to guarantee that any subsequent data can be parsed. By default, the dictionary should contain the set of ASCII characters, or at least the two binary digits.[4]

The parsed sequence of symbols is fed into the language model, and the model's prediction is used to encode the symbol stream. The power of the

[3]More correctly, an alphabet. We are mixing terms from two different fields, text compression and grammatical inference, here.

[4]The overhead of compressing the default dictionary should be considered when evaluating models.

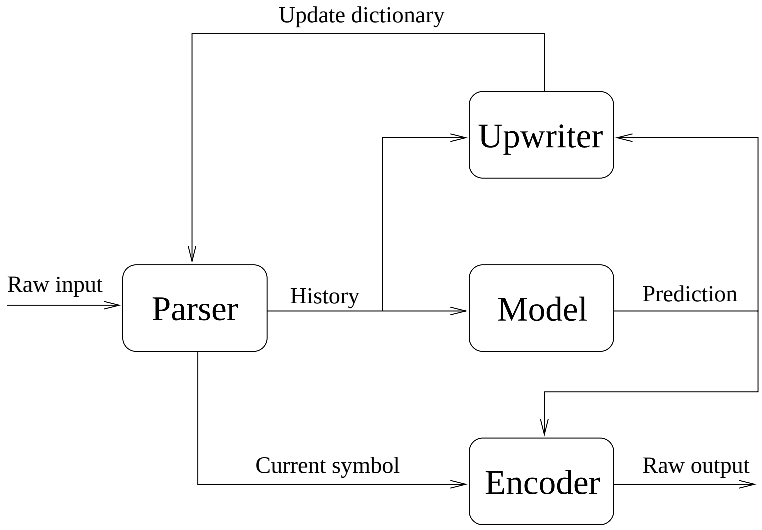

Figure 2: The upwrite predictor

system comes from the upwriter, which uses the symbol history and the prediction history to discover structure in the data. The upwriter can modify the dictionary by creating symbols which represent newly discovered structure, or by removing symbols which are deemed redundant. In this way, the model automatically incorporates new structure via the parser.

In the current implementation, the dictionary initially contains the set of ASCII characters, the parser uses greedy parsing, the encoder uses arithmetic coding, the language model is a simple trigram model as in Equation 4, and the upwriter uses entropic chunking and clustering to find deeper structure.

$$P(w_n|w_1, w_2, \ldots, w_{n-1}) \approx f(w_n|w_{n-2}, w_{n-1}) \tag{4}$$

The trigram language model makes the assumption that the probability of a symbol may be predicted from the preceding two symbols, and that this probability can be estimated from symbol frequencies obtained by counting trigram occurrences in a text [7]. To avoid the zero-frequency problem, the model begins by making the Bayesian assumption that everything is equiprobable.

This model has some disadvantages. Often, not enough context is available to make a good prediction. For example, although such models can reliably predict the symbol which follows the segment "th", they have difficulty predicting the symbol following ". ". Another disadvantage is that the model cannot group sets of symbols together. For example, if the symbol "9" appears in a new context, it cannot improve it's prediction by using statistics from other digits which have appeared in that context.

The upwriter may be used to counter these disadvantages. Using only the history of predictions and the history of symbols, we can find chunks of symbols which extend the context available to the model, and classes of symbols which allow the model to make better use of the statistics available.

3.1 Entropic Chunking

We can measure the entropy of the predictions that the trigram language model makes as in Equation 5, where Σ is the alphabet (that is, the dictionary). The entropy of the prediction may be considered to be a measure of how uncertain the model is about which symbol is coming next [2]. The entropy reaches a maximum when the model predicts that all symbols are equi-probable, and is minimum when the model assigns a probability of 1 to a particular symbol.

$$H = -\sum_{w \in \Sigma} p(w) \log_2 p(w) \tag{5}$$

If the entropy of the model becomes high as it is scanning across a text, it indicates that not enough context is available to make a reliable prediction. At that point, the upwriter may create a new symbol, or chunk, which consists of several symbols concatenated, in order to give the model more context for that particular case.

As Figure 3 shows, the entropy of the model fluctuates as it scans across a sequence of characters, and peaks at the boundaries between words, and at other logical places. If we consider points of high entropy to indicate chunk boundaries, chunks may be easily discovered and added to the dictionary by the upwriter [16].

`,"∧said∧`	`Holmes∧`	`Mr.∧`	`and∧`	`dis`
`er`	`ght`	`in∧the∧`	`ing`	`ion`
`of∧the∧`	`the`	`the∧`	`you∧`	`your∧`

Table 1: Some chunks found in natural language text

Table 1 lists some of the more common chunks found by the system on a single pass through a corpus of Sherlock Holmes stories, where $\wedge$ denotes the space symbol. The system has "discovered" many words and morphemes automatically. Even when all clues to word boundaries are removed, useful chunks are still found by the system. Note that both "the" and "the∧" are found. This allows the upwriter to create longer chunks beginning with "the", such are "there∧".

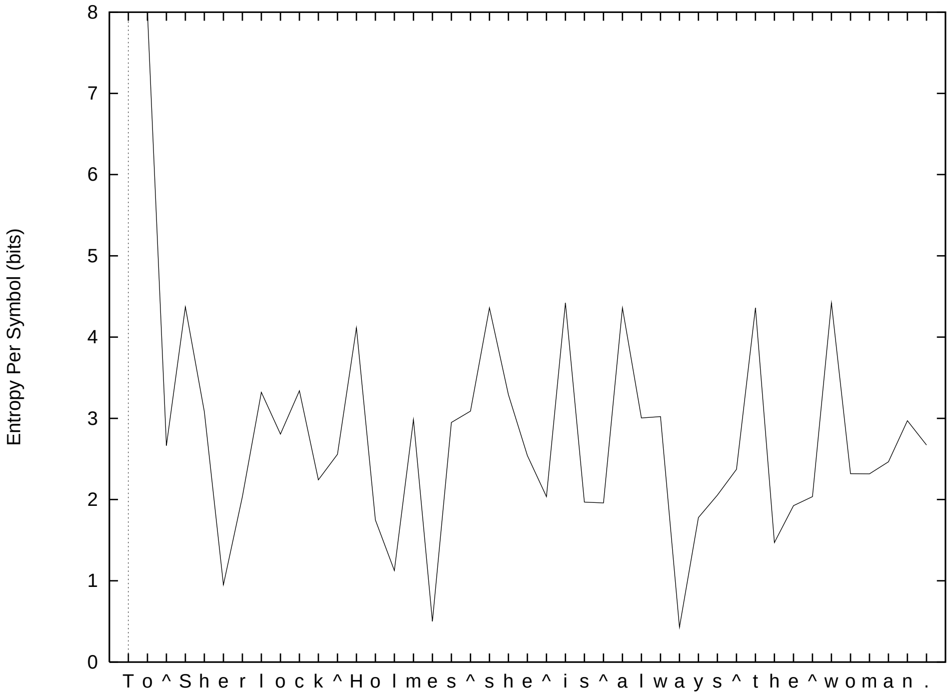

Figure 3: Entropy across a text segment

3.2 Clustering

Because several different symbols may represent similar "concepts", it may be useful to group them together into higher-level symbols. One reason for doing this is that it will allow more accurate statistics to be collected, as several trigrams will be mapped to a single, higher-level trigram. Symbol classes may be found by clustering prediction vectors from the language model [6, 13].

If the language model makes similar predictions from two different contexts, these contexts may be considered to be similar in some way. By clustering the history of prediction vectors, groups of equivalent contexts may be found. These groups correspond to symbol classes, and may be added to the dictionary as new symbols.

Table 2 lists some symbol classes discovered by a simple clustering algorithm using a corpus of Sherlock Holmes stories.[5] In this case, the dictionary was initialised to contain all words in the corpus. The table shows only the most frequent symbols from each class, and many classes have been omitted due to space restrictions. It can be seen that some very useful symbol classes have been found, including classes for punctuation and proper names. Note

[5]Each row of the table lists symbols belonging to a particular class. Note that the corpus used was a mere 200Kb in length.

!	.	?	and
,	-	:	;
is	was	were	are
find	have	get	take
beautiful	attractive	gorgeous	startling
carried	recounted	aided	fabricated
America	Chicago	England	
Douglas	Hudson	Larbey	
detective	inspector		
pistols	six-shooters		

Table 2: Some symbol classes found in natural language text

that these classes tend to be more context specific than the standard parts-of-speech, and may be considered to represent "concepts" to a small degree (consider the *beautiful-attractive-gorgeous-startling* class, for example).

4 Human Language Acquisition

The upwrite predictor uses simple information processing techniques to find complex structure in data. It does this by melding a *naïve* symbol-counting language model with a simple upwriter which looks for basic regularities in the statistics. The real power of the system stems from the feedback of new symbols into the language model. This "adaptive upwrite" process allows a hierarchy of symbols to be found quickly and easily, which is an ideal property for a natural language acquisition system.

We may speculate that the human brain works in a similar way; predictions may be made about the future, and compressing the past would increase the amount of information available to make these predictions [17]. Recent findings in psycholinguistics suggest that frequency based processing may play an important role in human language acquisition [10, 12, 11]. Additionally, the incremental upwrite process abstracts and generalises data, which makes recognition and comparison operations viable.

At this point we should note that the upwrite concept has also been used successfully for quite different applications, such as speech and image processing [1]. That one simple concept can be applied to several areas of cognition is encouraging.

5 Results

Because this is very much a work in progress, few results are given at this point in time.[6] Suffice it to say that the upwrite predictor achieves compression ratios superior to those of gzip on text files, and comparable (or slightly inferior) on binary files. In all cases, it out-performs the interpolated IBM predictor when using ASCII characters as symbols.[7]

6 Future Work

We have only recently embarked on this line of investigation, and so our work is quite preliminary. In the future, we will tackle the following issues:

1. Different parsers and language models will be tested as components in the upwrite prediction system. This process should be easy, as the system is designed in a modular way so that one component can be entirely replaced without affecting others.

2. Experiments will be conducted on other sources of data, with the default dictionary containing the two binary symbols only. On textual data it will be interesting to see if the system discovers the ASCII characters automatically.

3. Some other promising methods of finding chunks of symbols, and classes of symbols, will be investigated.

4. An in-depth comparison of our system with state-of-the-art compression techniques will be performed.

5. Tests comparing popular language models, using both perplexity and compression as performance measures, will be conducted.

6. The system will be expanded to deal with higher-dimensional data sets, such as images.

[6] The only results we show have already been given in previous sections.

[7] The IBM predictor performs better when words are used as symbols, but it must be told what a word is, and it doesn't factor in the size of the dictionary when calculating the perplexity!

7 Conclusion

In this paper we have advocated the use of statistical compression systems as a performance measure for stochastic language models, and we have described the upwrite predictor, which is a generic language model. The upwrite predictor was shown to find useful structure in natural language text using simple techniques, and we hope that this lends some credence to the claim that human beings may use some statistical processing in language acquisition. We have also claimed that the upwrite predictor can find structure in all sorts of one-dimensional data, if it is there to be found. Experiments have yet to be conducted to substantiate this claim.

8 Acknowledgements

We would like to thank the Centre for Intelligent Information Processing Systems for making it possible for us to conduct this research, and the Microsoft Research Institute at Macquarie University for funding Jason Hutchens' trip to the Natural Language Processing Workshop.

References

[1] Michael D. Alder. Inference of syntax for point sets. In E. S. Gelsema and L. N. Kanal, editors, *Pattern Recognition in Practice IV: Multiple Paradigms, Comparative Studies and Hybrid Systems*, number 16 in Machine Intelligence and Pattern Recognition, pages 45–58. Elsevier Science B.V., Jun 1994. Available at: `ftp://ciips.ee.uwa.edu.au/pub/syntactic/papers/tr94-04.ps`.

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

[3] 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, 1986.

[4] Timothy C. Bell, John G. Cleary, and Ian H. Witten. *Text Compression*. Prentice Hall, 1990.

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

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

[7] Jason L Hutchens. Natural language grammatical inference. Honour's thesis, University of Western Australia, December 1994. Available at: `http://ciips.ee.uwa.edu.au/ hutch/Research.html`.

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

[9] David J.C. MacKay. *Bayesian Methods for Adaptive Models*. PhD thesis, California Institute of Technology, 1992.

[10] Don C. Mitchell, Fernando Cuetos, Martin M.B. Corley, and Marc Brysbaert. Exposure-based models of human parsing: Evidence for the use of coarse-grained (nonlexical) statistical records. *Journal of Psycholinguistic Research*, 24(6):469–489, 1995.

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

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

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

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

[15] Ross Neil Williams. *Adaptive Data Compression*. PhD thesis, University of Adelaide, 1989.

[16] J. G. Wolff. The discovery of segments in natural language. *British Journal of Psychology*, 68:97–106, 1977.

[17] J. Gerard Wolff. Language acquisition, data compression and generalization. *Language & Communication*, 2(1):57–89, 1982.