Grammatical Inference

(PURE AND SIMPLE)

Jason L. Hutchens
Michael D. Alder

November 12, 1997

THE LANGUAGE RACE

Over the last 30 years, the Artificial Intelligence and Computational Linguistics communities have developed sophisticated parsers to allow computers to "understand" English in a limited way.

During the same period, the Engineering community has developed speech recognition systems which utilise statistical language models for disambiguation.

Grammatical Inference aims to combine these two techniques by using statistical methods to learn a grammar for some language.

TERMINOLOGY

Terminology can cause some confusion due to the melding of techniques from various domains.

In this talk I call the lowest-level symbols *words*, represented by w_i.

I call a string of words a *text*, represented by $w_{1,N}$.

The *dictionary* contains all possible words, and is represented by Σ.

STATISTICAL MODELS

The only requirement of a statistical model is that it must assign a non-zero probability to *any* text $w_{1,N} \in \Sigma^+$.

The probability of such a text may be represented thusly:

$$P(w_{1,N}) = P(w_1)P(w_2|w_1)\ldots P(w_N|w_{1,N-1})$$

Statistical language models are often used for *prediction*.

INFORMATION THEORY

The *information* is a measure of surprise:

$$I(w_n|w_{1,n-1}) = -\log_2 P(w_n|w_{1,n-1})$$

The *entropy* is a measure of uncertainty:

$$H(w_{1,n-1}) = \sum_{w \in \Sigma} P(w|w_{1,n-1}) I(w|w_{1,n-1})$$

PERFORMANCE MEASURES

The data compression field use average bits-per-symbol:

$$\overline{I}(w_{1,N}) = -\frac{1}{N}\log_2 P(w_{1,N})$$

The speech recognition folk use *perplexity*:

$$PP(w_{1,N}) = P(w_{1,N})^{-\frac{1}{N}}$$

But they're both measuring the same thing:

$$PP(w_{1,N}) = 2^{\overline{I}(w_{1,N})}$$

MARKOV CHAINS

A Markov chain of order $n-1$ is referred to as an n-gram model. Such models assume that only the previous $n-1$ words have any influence on the probability of the next word.

These probabilities are estimated from frequency counts:

$$P(w_n|w_{1,n-1}) \approx f(w_n|w_{n-2,n-1})$$

where:

$$f(w_n|w_{n-2,n-1}) = \frac{C(w_{n-2,n})}{C(w_{n-2,n-1})}$$

SMOOTHING

Smoothing is used to ensure that every possible text is assigned a non-zero probability:

$$
\begin{aligned}
P(w_n|w_{1,n-1}) \approx & \\
& \lambda_1(w_{n-2,n-1})f(w_n) + \\
& \lambda_2(w_{n-2,n-1})f(w_n|w_{n-1}) + \\
& \lambda_3(w_{n-2,n-1})f(w_n|w_{n-2,n-1})
\end{aligned}
$$

Where the λ_i sum to one, and are functions of the context. The big question is. . . where do they come from?

HIDDEN MARKOV MODELS

The speech recognition people treat the smoothed model as an HMM in order to optimise the smoothing weights using the Baum-Welch algorithm.

This method is time-consuming, and only satisfactory if the model is non-adaptive.

Of course, the technique is sensitive to initial conditions, and finds a local maxima only.

TEXT COMPRESSION

Because the model begins with no knowledge of the language, an extra -1-order model is used, which assigns each symbol an equal probability.

Because the model is real-time and adaptive, the smoothing weights are adaptive functions of the context.

The best techniques perform well, but there is little justification for them.

GRAMMATICAL INFERENCE

In English, several words may constitute a higher level entity, such as a noun-phrase or a sentence.

Also, words may be grouped into classes. In English, one word can have several classes, such as verb, adjective and noun.

We can program the models to look for such structure–we are assuming its existence and usefulness.

ENTROPIC CHUNKING

The instantaneous entropy of a statistical model is a good indicator of boundaries between word sequences which constitute higher-level structure.

This is because prediction is easier within a related sequence of words than it is between two unrelated sequences.

We can find chunks by thresholding the instantaneous entropy. But how do we perform this thresholding, and how do we distinguish good chunks from bad?

FINDING WORD CLASSES

Each possible context has an associated prediction. These predictions may be viewed as points on a hyper-triangle thingy in $\Re^{|\Sigma|}$.

Clusters of predictions give evidence of word classes. The centroids of these clusters may be used as the classes themselves.

Words which make similar predictions may themselves belong to one class.

The trouble is, many clusters may represent the intersection of two or more "real" classes. So how do we find the kosher classes?

OUR GOALS

We would like to use a very basic language model to find structure in a text which may be used to bootstrap the model, making it more powerful.

We would like to make as few assumptions about the language as possible, to ensure that our technique is generic.

We would like to do this to test how much information about language may be gained from observation of examples only.

CONCLUSION

The field is still in its infancy; the various specialist fields are still not talking to each other, and the terminology is all over the place.

Different fields have different goals. Compression, recognition or understanding? Are they all the same thing?

Sparse data causes headaches; we need to be able to measure our confidence in the model for a particular context, based on the weight of training data.

The processing and memory requirements are enormous, which makes experimentation a pain in the arse.

BONUS HIDDEN SECTION

Game theory is an enjoyable diversion from your PhD.

How can we partition a set of numbers into two subsets of equal size in order to minimise the difference of their sums?

Can we write a computer program to solve a crossword type puzzle, and make an easy $2000...

...or is granny technology the best we're gonna get?