Finding Chunks in Symbolic Time Series

Jason L. Hutchens
Michael D. Alder

September 4, 1998

THE PROBLEM

We wish to find sequences of symbols which constitute meta-symbols at some higher level of abstraction. We call such sequences *chunks*.

One example is finding words in English text, although we don't want to restrict ourselves to this domain.

This problem has arisen out of my work on data compression. Finding chunks is one way to improve the power of the model—by bootstrapping its alphabet, rather than by changing parameters of the model.

THE MODEL

We use a low-order Markov model. Ideally, we use a first-order model, as we wish longer contexts to be made available by chunking.

The model should begin by using the lowest level alphabet. Usually this will consist of bytes (although bits are even lower).

Chunking methods may be used by any predictive model. We choose the simplest to develop our methods.

THE CORPORA

We consider three corpora, which are all lower-level representations of the same zero-information source `ABABABAB...`

Corpus 1: `0001000010001000001...`

Corpus 2: `000111000111000111...`

Corpus 3: `ccxxddyyddwwxxyybb...`

Markov models of order 3 and 8 are sufficient to model the first two corpora. An upwrite to symbol classes is required to model the third, although that is beyond the scope of this talk.

METHOD 1: CHUNK THE CONTEXT

This requires a second-order Markov model or better. The idea is to create a chunk out of the context which the model has the most trouble with.

The entropy of the model's prediction is used to determine this.

Corpus 1: `00`

Corpus 2: `00` and `11`

Corpus 3: `cc`, `dd`, . . . , `zz`

METHOD 2: ENTROPIC CHUNKING

The model scans the text, and a chunk-boundary is inserted whenever the entropy exceeds some threshold value.

For the purposes of this example, this threshold is set to the lowest entropy, which is equivalent to agglutinating symbols into chunks whenever the entropy equals the threshold.

Corpus 1: 01

Corpus 2: none

Corpus 3: none

METHOD 3: INFORMATION

Similarly, we chunk whenever the information of a symbol exceeds some threshold value.

For the purposes of this example, the threshold is set to the lowest non-zero information value.

Corpus 1: `1000` and `10000`

Corpus 2: `111` and `000`

Corpus 3: `aaa`, `bbb`, . . . , `zzz`

METHOD 4: SEPARATOR SYMBOL 1

Some data sources put separator symbols between chunks. We postulate that such symbols precede large jumps in entropy.

Statistics about the size of the jump are collected for each symbol, and a single separator is chosen.

Corpus 1: `0`

Corpus 2: none

Corpus 3: none

METHOD 5: SEPARATOR SYMBOL 2

Similar to the previous method, but the information of the symbol is used.

Corpus 1: 1

Corpus 2: none

Corpus 3: none

DISCUSSION

For these corpora, method 3 works the best. In fact, it works perfectly, allowing the model to “discover” the data source after only one upwrite.

However, method 3 requires us to select a threshold, while methods 1, 4 and 5 don’t.

None of the three corpora have separator symbols, so methods 4 and 5 can’t be evaluated properly.

All of the methods have been found to work well on Natural Language data, and it is difficult to choose between them in this case.

CONCLUSION

We have presented some chunking methods. Each has advantages and disadvantages.

Natural Language data is a lot more “noisy”. The next step will be to adapt these methods for use on such data.

Results haven’t been fully explored.

What do we do when the data looks completely random to a low-order Markov model?