Language Models

Martin Kay

September 17, 2006

In statistically based natural language processing a language model is used in the generation of output strings to assess likelihood that a given string of words is a sentence in a particular language. If the information about the ordering of the words in the sentences in a texts were discarded so that each sentence was regarded simply as a bag of words, then the better of two language models would be able to do a better job of restoring their order. According to the simplest view of statistical machine translation, this is just what the language model is called upon to do. For each source sentence, a translation model proposes a bag of words to be used in its translation and it is the job of the language model to order them in the best way. There is, in fact, more to it than this, but it at least gives a setting for the use of language models.

Language models are based on so-called n-grams. An n-gram is simply a sequence of n words. The likelihood that a permutation of the given bag of words is a sentence of the language is estimated by taking the product of the probabilities of all the n-grams that it contains. Thus, a sentence of k words contains $k - n + 1$ n-grams, and the probability of each of them is estimated on the basis of a body of training data. Various tricks are used to allow for permutations that did not occur at all in the training data.

The value of n is fixed at a fairly low value—say 3, 4, or 5—because, though larger values generally give better results, the amount of training data required to permit a useful estimate of the probability of longer strings grows at an overwhelming rate with value of n.

This note results from my reflections on the possibility of language models based on sequences of variable size. If a given set of training data of fairly modest size contains some three or four instances of strings consisting of eight or ten words, then this is surely a remarkable fact, and one from which it should be possible to derive important advantages.

Observe, first of all, that it is a relatively straightforward matter to catalog the repeated sequences of whatever length in a corpus of text. They

can be read out directly from a suffix tree which can be constructed from the text at a cost in time and space that is linearly related to the length of the text. Given an arbitrary bag of words, it is also straightforward, and not overwhelmingly expensive in computational resources, to draw up a list of all the strings constructable from it, of whatever length, that were seen in the training corpus. Given an arbitrary string that uses all the words in the bag, it is therefore simple to determine just which of these it contains.

Suppose that each of the strings from the training data is assigned a figure of merit (*fom*, plural *fom's* (sic.)) based on a variety of properties including, most notably, its length and the number of its occurrences in the training data. It remains to derive from the fom's the constituent strings, a figure of merit for a string covering all the words. This is apparently difficult for two reasons, especially if fom's are based on probabilities. First, as we have already observed, reasonable sized corpora provide little more than anecdotal information about long substrings and, second, there is no obvious way of combining the fom's of strings of different sizes into a single fom for the complete string. The probabilities of strings of different lengths are not commensurate: a longer string is generally speaking altogether less probable than a shorter one. I will return to this question shortly.

First, let us assume that a solution to this problem is at hand, and that it has some simple properties that would be desirable if it were to be useful in the kind of process we have in mind. In particular, let us assume that it has a reasonable monotonicity property so that, if a pair of strings is concatenated, the fom of the result will be readily calculable from those of the parts and it will either never be greater than, or never less than, the fom of either constituent.

The following procedure composes a list of strings from the members of a given bag, all of which occur in the training data. Walk the suffix tree constructed from the training data, increasing in depth only when all the characters in the path to the new node are in the given bag. When this ceases to be the case, record the string constructed from the current path as a new member of the set unless the potential new member is a proper substring of some existing member.

Using this set of strings, the set of strings each of which consumes the whole of the initial bag can be constructed as follows:

1. Draw up a list of strings, composed only of words in the initial bag, that occurred in the training data, in decreasing order according to their fom's. The strings are *maximal* in the sense that if one string is a substring of another, only the latter is included, though the existence

of the shorter string will, in general, be reflected in the calculation of the fom of the other. Call this list the initial *agenda*.

2. Create a second list, initially empty, called the *chart*.

3. Repeat the following until a *complete string* has been created, that is, a string that consumes all the words in the initial bag.

   (a) Take a string, α from the head of the agenda together with each string β in the chart in turn. If the bag of words required to compose both strings is contained in the initial bag, create the strings $\alpha\beta$ and $\beta\alpha$ and put them into the agenda list in the place dictated by their fom's.

Notice that the concatenation of, say, α with a following β implicitly concatenates all suffixes of α with all prefixes of β and any of these that occurred in the training data will, in general, have to be considered in computing the fom of the new string. It is not hard to see that, if fom's are well behaved, this procedure produces the *best* complete string—the one with the highest fom—before any others. In other words, it constitutes a *best-first* algorithm.

Outlining these procedures first makes it clearer what it means for fom's to be well behaved. Most importantly, only if the fom of a string $\alpha\beta$ is no less than those of α and β will the procedure just outlined constitute a best-first algorithm.

The trouble with many best-first algorithms, including this one, is that they are almost breadth-first algorithms because the best complete result emerges only after almost *all* almost partial results have been computed. This presumably provided the primary motivation for the development of the $A*$ algorithm, which takes into account the minimum possible cost of developing a partial result into a complete result. There are various ways in which the present algorithm might be turned into a version of $A*$, but the details depend of the particular way in which fom's are to be computed. This remains the principal outstanding question.

As we said at the outset, the canonical problem for a language model is to restore to the extent possible the original order to a bag of the words found in a sentence. The standard approach is to seek the order that is most probable in the language as a whole, estimated on the basis of a training corpus. A major problem with this, as we also noted, is that the probability estimates of substrings of different lengths are incommensurate.

But, could it be that the standard statistic is not only difficult to use, but also not the best one for the job? A very different one, which may be

easier to use is the probability the substring, *given the bag of the words that make it up*. In other words, it is the number of instances of the substring in the training data, divided by the number of sentences that contain all the words in it.

Consider a string S that contains substrings $s_1, s_2...s_k$, all of which are also substrings of the training corpus, that are maximal with respect to S. Might the fom's of the substrings, $f_1, f_2...f_k$, be assigned in such a way that the fom of S would be $f_1 + f_2... + f_k$, or something comparably simple?