Similarities and differences between the statistical models for speech recognition and parts-of-speech tagging

Leif Grönqvist, leifg@ling.gu.se

Jan 3 2002, Course paper, TT1

Abstract

In this paper we will compare the statistical models for automatic speech recognition (ASR) and parts-of-speech tagging (POS-tagging) and try to come up with some hypotheses on how to improve the ASR model using techniques used in POS-tagging. Experiments with different smoothing algorithms and use of spoken data as a part of the language model are suggested.

1 Introduction

The two problems discussed in this paper - POS-tagging and ASR - are totally different and it is difficult to see anything in common between them at all. In spite of this, lots of researchers in the two fields are using the same basic model. They start from Shannon's [1] noisy channel model and use Bayesian inference to get a computationally sufficient statistical model. In the ASR case, the noisy data is the sound wave, and the source data is a sequence of words. For POS-tagging the source data is a sequence of parts-of-speech and the noisy data the corresponding sequence of words - a bit more unintuitive.

2 Background

In 1996 a research group started to work on a statistical POS-tagger for spoken Swedish. The work was initially presented in Nivre et al (1996) and the results were promising. The continuation went in two directions: tuning the tagger for spoken language [2], presented in Nivre & Grönqvist 2001, and exploring and evaluating different smoothing algorithms used in the statistical model, presented in Nivre 2000.

3 Parts-of-speech tagging

To start with, the POS-tagging problem may be formulated as:

$$argmax_{t_1,\ldots,t_n} P(t_1,\ldots,t_n|w_1,\ldots,w_n)$$

Given a sequence of words $w_1,\ldots,w_n$, calculate the most probable sequence of POS-tags $t_1,\ldots,t_n$. Bayesian inversion allow us to rewrite the formula as:

$$argmax_{t_1,\ldots,t_n} \frac{P(w_1,\ldots,w_n|t_1,\ldots,t_n)P(t_1,\ldots,t_n)}{P(w_1,\ldots,w_n)}$$

The probability is maximized over $t_1,\ldots,t_n$ so the formula may be simplified as follows:

$$argmax_{t_1,\ldots,t_n} P(w_1,\ldots,w_n|t_1,\ldots,t_n)P(t_1,\ldots,t_n)$$

[1] Claude Elwood Shannon was an electrical engineer and matematician. He died in February 2001.

[2] the only available tagged training data was written Swedish.

This is actually impossible to calculate because of the sparse data problem - if n is big, the sequence of words and POS-tags will not be found in the training data. The Markov assumption for trigram [3] and the assumption that the probability for a word only depends on its tag, may now be used to make an approximation of the formula that makes it possible to calculate with a reasonable amount of training data:

$$argmax_{t_1,\ldots,t_n} \prod_{i=1}^{n} P(w_i|t_i)P(t_i|t_{i-1}, t_{i-2})$$

Now, the probabilities may be estimated with relative frequencies (MLE [4]).

The Viterbi algorithm, initially presented i Viterbi 1967 (or a bit easier to read in Dugad 1996), may be used to do this, but there are still some problems to solve: a lot of data in new data are not present in the trainig data. We are going to use a trigram based HMM-tagger[5] in the following sections. The labels on each state in the HMM is in this case is a pair of POS-tags and the symbol alphabet is the set of all words.

3.1 Smoothing

Smoothing is needed because some events that we need probabilities for are missing in the training data. The two types of probabilities: $P(w_i|t_i)$ called observation likelihood or lexical probabilities for POS-tagging, and $P(t_i|t_{i-1}, t_{i-2})$ (prior probabilities or contextual probabilities for POS-tagging) are not equally difficult to estimate. The number of contextual probabilities are bounded by the number of parts-of-speech which is relatively few, but the lexical probabilities may be almost infinite. So the smoothing is much more important for the lexical probabilities but the contextual should also be smoothed. Here, some of the most common methods are briefly described. More details will be found in Nivre 2000.

3.1.1 Additive smoothing

Additive smoothing is the easiest method to implement and probably to understand too. Instead of using the raw frequencies as with MLE, a constant number of observations are added to the frequency. The reasoning behind this is that we add unseen events that should have been seen in a bigger set of training data. The smoothed probability is calculated as:

$P(x) = \frac{f(x)+k}{N+kn}$, where N is the total number of observations in the training set and n the number of possible different observations (the size of the sample space), instead of the unsmoothed $P(x) = \frac{f(x)}{N}$. The kn in the formula has to be added to make sure that the sum of the probabilities are still 1. The most popular choice for the constant is $k = 0.5$ which is known as Lidstone's Law [6] or Expected Likelihood Estimation (ELE).

3.1.2 Back-off smoothing

For the events not present in the training data, this method "backs off" to a more general event. Unseen trigram probabilities may for example be estimated by bigram probabilities.

3.1.3 Linear interpolation

The probabilities will be estimated as a weighted sum of the probabilities for the actual event and more general events. Trigram probabilities may for example be estimated as a combination of trigram, bigram and unigram frequencies. The more general events should contribute less, but make sure that all probabilities are nonzero.

[3] The probability for a tag-value will only depend on the two tags before, and this probability is also independent of time.

[4] Maximum Likelihood Estimation, presented in Nivre & Grönqvist 2001

[5] Hidden Markov Model, see Merialdo 1994

[6] Lidstone wrote about this in 1920(!) This kind of statistical models must be used for a lot of other things as well.

3.1.4 Good-Turing smoothing

This method is much more sophisticated. With the frequency distribution for the different events as input, the expected number of unseen events are calculated and added to the seen frequency. This can be done using expectation values (Good 1953).

4 Speech recognition

Basically the model is the same for ASR as for POS-tagging, but the statistical model is much more difficult to estimate, so even more tricks are needed. As for POS-tagging, the prior probabilities (called language model in ASR) are the easiest to estimate, but the set of them are not bounded. Instead of being based on sequences of parts-of-speech they are sequences of words and the probabilities are on the form: $P(w_i|w_{i-1}, w_{i-2})$. Huge text corpora (hundreds of millions of running words) are used to make it possible to find enough of the word triples [7] in the training data but still, most of the triples are missing, and the smoothing has to be really effective. The observation likelihood (called acoustic model) are probabilities for a sound waveform segment depending on a phone and of course this can not be done with MLE - exactly the same sound segments as recorded are never present in the training data. Instead this has to be done by solving several problems (Jurafsky & Martin p 241):

- The sound wave has to be segmented. This may be done by slicing it up into frames of 10-20 milliseconds.
- Each segment is transformed into spectral features (energy levels in different frequency intervals)
- A neural net will give probabilities for each phone given the spectral features.

Together with prior probabilities for sequences of phones it is now possible to build a Markov chain.

Now it is possible to use a HMM with phones both in the states and as observed symbols, and together with the Viterbi algorithm we will get probabilities for different phone sequences the most probable sequence of phones given a sound waveform. Not exactly what we wanted but a step in the right direction. The problem is reduced to: find the most probable word sequence given a sequence of phones which is much easier to solve.

4.1 Decoding

One way to do this is to calculate pronunciation networks for every possible input word, and connect them to each other using bigram probabilities for words. This will give an automaton with states for each phone. There will be several states for each phone because they are present in several pronunciation networks. The best path through this big automaton may be found by the Viterbi algorithm, but there is a problem: it will find the best sequence of phones - not the best sequence of words! Maybe this does not matter, the best sequence of phones is probably at least a good sequence of words, but there is another big disadvantage, much more important for ASR than for POS-tagging:[8] Viterbi will not give an N-best list or a lattice. In Jurafsky & Martin p 251 two ways to solve this are presented below. Maybe a third way to solve it could be to build more complicated HMM:s with more than one phone in each state, to be able to get rid of the limitation that the Markov assumption gives us.

4.1.1 Modifying the Viterbi algorithm

One way to be able to find the best word sequence instead of the best sequence of phones is to modify the Viterbi algorith to give an N-best list of phone sequences, keeping the probability, and then use a word

[7] Triples or pairs are used depending on the size of the training data

[8] If you get the wrong POS sequence, most of the tags are probably right, but in the ASR case the meanings other sequences often are totally different, and hopefully the right word sequence may be found among the N best when comparing with the expected utterances. Especially if the domain is complex, so called word lattices are useful. They may contain a lot more different possible paths than N-best lists.

based trigram language model to assign new probabilities. Then the acoustic probability and the language model based may be combined to get a new score for each word sequence, giving a new N-best list.

4.1.2 The A* decored

Another way is to use the stack decoder, also called A* decoder because it is based on the A* search algorithm. It does work in a totally different way than the Viterbi algorithm and it will always give us the word sequence with the best probability. It works with a priority queue of utterance starts with probabilities, and a function that is able to estimate the probability for the rest of the utterance. A detailed description is found in Jurafsky & Martin p 254. The difficulty is to define this function (often called h*) and to make a good so called fast match function, that gives a list of the most probable next words. The A* decoder works in a best-first manner.

5 Possible improvements

There are a lot of weaknesses in the ASR systems that are not solved by changing the statistical models statically, for example the bad performance in noisy environments or the recognition of different speakers. Maybe more sophisticated neural nets for the phone recognition could improve this. If the neurons recognized the different phones on the basis of speaker independent features in some way, if possible, it could probably work better.

5.1 Prosodic data

Prosodic data, like questional intonation or stressed words for new information or answers, may be used to improve speech recognition in a dialogue context, but probably it should come in a later stage of the dialogue system. If the recognizer returns lattices or N-best lists it is possible to do so. On the other hand, if the different modules are implemented in a language with lazy behaviour (for example Haskell) a list of possible utterances may be passed to the next stage of the dialogue system, the different utterances will be considered by the interpreter module and when an utterance is good enough, the rest of the N-best list will not even be calculated at all.

5.2 Smoothing

The smoothing algorithms used in ASR are the same as in POS-tagging. We have not seen any comparisons between them for the ASR task, but in Nivre 2000 the best results for the contextual probabilities were reached by additive smoothing - the simplest available method. This could be the case for ASR too, but one can not be sure before trying it out.

5.3 Different training data

The language model for ASR uses a lot of written language data. Depending on what kind of speech the system is supposed to recognize, this may be bad. The training data should be of the same kind as the recognized data, so for dictating novels or newspaper articles the current training data used may be optimal, but if we want a recognizer for natural spontaneous spoken dialogue [9] , it would of course be better to have natural spontaneous spoken dialogues as training data. The amount needed is as mentioned huge, and too expensive to collect, but in combination with the written data, smoothed together with back-off or Linear interpolation smoothing, it may work. Hopefully the small vocabulary used in most parts of spoken dialogue in dialogue systems could be covered by existing spoken language corpora. It would definitely be worth a try.

[9] As we want for our dialogue systems

References

Dugad, R., Desai, U. B. (1996) A Tutorial on Hidden Markov Models. *Technical Report No SPANN-96.1*

Good, I. J. (1953) The Population Frequencies of Species and the Estimation of Population Parameters, *Biometrika 43, 45-63*

Jurafsky, D., Martin, J. H (2000) Speech and Language Processing - An introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition. 141-284 *Prentice Hall*

Lidstone, G. J. (1920) Note on the General Case of the Bayes-Laplace Formula for Inductive of A Posteriori Probabilities. *Transactions of the Faculty of Actualities 8, 182-192.*

Merialdo, B. (1994) Tagging English Test with a Probabilistic Model, *Computational Linguistics 20, 155-171*.

Nivre, J. (2000) Sparse Data and Smoothing in Statistical Part-of-Speech Tagging. *Journal of Quantitative Linguistics, 7(1), 1-17.*

Nivre, J. & Grönqvist, L. (2001) Tagging a Corpus of Spoken Swedish. To appear in *International Journal of Corpus Linguistics*.

Nivre, J., Grönqvist, L., Gustafsson, M., Lager, T. & Sofkova, S. (1996) Tagging Spoken Language Using Written Language Statistics. In Proceedings of the 16th *International Conference of Computational Linguistics*. Copenhagen: Center for Language Technology.

Rabiner, L. R. (1989) A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. *Proceedings of the IEEE, Vol 77-2*

Viterbi, A. J. (1967) Error Bounds for Convolutional Codes and an Asymptotically Optimal Decoding Algorithm. *IEEE Transactions on Information Theory* 13, 260-269