Speech Processing II

Miriam Butt
October 2002

Recap

Last Time: Basics of what one needs to worry about when dealing with spoken language.

The most successful speech recognition systems to date have been based on statistical methods.

We will now try to understand the various components which make up a typical statistical approach.

Bayes Rule

The statistical likelihood that a given word stored in the database can be matched to a given piece of an acoustic string is generally calculated via *Bayes Rule*.

This method was propagated by Fred Jelinek at IBM and is a special instance of the more general idea of *Bayesian inference* or *Bayesian classification* (Bayes 1763).

Task: given an observation, find out which of a set of classes it belongs to.

Authorship Attribution

Task: given a set of anonymous papers, find out who the author is based on observations about their non-anonymous writings.

Example: The Federalist papers were 85 essays written in 1787-1788 by Alexander Hamilton, John Jay and James Madison. Of these, the authorship is unclear for 12: was it Hamilton or Madison?

Mosteller and Wallace (1964) trained a probabilistic model on the writing of Hamilton and Madison, respectively and computed the maximum-likelihood author.

[ni] and the Switchboard Corpus

In the Switchboard corpus, a total of 7 words could be pronouced [ni].

neat, need, new, knee, to, you, the

We will pretend we have received the acoustic signal corresponding to [ni] and are now trying to figure out which of these words it could be.

Bayes Rule

$$\hat{w} = \underset{w \in V}{\text{argmax}}\ P(w|O)$$

O is the observation: [ni], *w* is the word

The word *w* has to be a member of a given vocabulary V (in our case V= *neat, need, new, knee, to, you, the).*

$\hat{w}$ is the estimate of the correct word *w*.

$argmax_x f(x)$ means "the *x* such that *f(x)* is maximized"

P(w|O) is the probability of a word given the observation.

P(w|O) is impossible to compute.

Bayes Rule

$$\hat{w} = \underset{w \in V}{\text{argmax}}\ P(w|O)$$

Instead: compute *P(w|O)* indirectly via three other probabilities:

$$P(w|O) = \frac{P(O|w)P(w)}{P(O)}$$

P(w) is the probability of a word (frequency), this is easy, given a corpus.

P(O) is the probability of the observation. This is very difficult to estimate, but one doesn't need it because *P(O)* doesn't change for each word

Bayes Rule

$$\hat{w} = \underset{w \in V}{\text{argmax}}\ P(O|w)P(w)$$

- *P(O|w)* is called the *likelihood*
- *P(w)* is called the *prior probability*

Computing the Likelihood

One Way: Compute the likelihood for a corpus that a certain phoneme will be realized by a given sound.

For example: check what the probability is that *th* is realized as [n] in a given corpus when it follows a nasal.

$$\text{Probability} = \frac{\text{ncount}}{\text{envcount}}$$

envcount is the number of time *th* occurs

ncount is the number of times *th* is realized as [n] when it follows a nasal.

Computing the Likelihood

See handout (J&M pp. 166-167).

Our probability calculation predicts *new* to be the best choice. However, for [ni] as representing *need* in *I need*, this is clearly the wrong choice: information about the larger linguistic context is missing: *I new* is not English...

Weighted FSA

Speech recognizers depend on a database (or lexicon) of precompiled pronunciations (relationship between a word and its pronunciation).

No use of *lemmas* is made.

Most speech recognizers use Hidden Markov Models (HMM) to arrive at these databases.

A Markov Chain (used by HMMs) can be shown to be equivalent to a weighted FSA.

See handout (J&M Fig. 5.18 and 5.20).

Improving the Guess

The use of only Bayes Rule to estimate the probability that a given (stored) word in a vocabulary matches a sound observation is not sufficient.

Solution: Find techniques that make the statistical estimation more sophisticated.

N-Grams provide a way to look at the surrounding context.

N-Grams

Word Prediction: Our task would be easier if we could predict what the next most likely word would be, if we are in the middle of an utterance.

I'd like to make a collect ...

Applications: Speech Recognition, hand-writing recognition, faster mobile typing, spelling correction, communication software for disabled people.

N-Grams

An N-Gram takes the previous N-1 words to predict the next word.

In Speech Recognition, the most common model is a *bigram*.

Ideally, one would like to look as far back in the sequence/utterance as possible. But again, one doesn't know how to compute the probability of a word given a long sequence of preceding words.

The Markov Assumption

Solution: approximate the probability of a word given all the previous words by only using the single previous word.

Assumption: P(rabbit|I just saw a)
can be approximated by P(rabbit|a)

The bigram model thus approximates the probabibility of a word given all the previous words by the conditional probability of the preceding word.

The Markov Chain

A basic bigram model can be thought of as a Markov chain (weighted FSA) which has one state for each word.

Can also have a trigram, quadigram, etc. model, but the cost of coputing goes up the more we look in the past.

Playing with N-Grams

The further in the past you look, the better you do.

The bigger and more complete your training corpus is, the better you do.

Note: Names are a huge problem. One newspaper corpus had 21% names. These mostly cannot be stored and compiled ahead of time (company names, etc).

See handout (J&M pp. 203 and 205-206).

Back to [ni]

The use of a bigram model now gives us the correct result for the recognition of [ni] in the vocabulary of the Switchboard corpus.

See handout (J&M p. 223).

Word Lattices

To further improve the results for long utterances, applications often build *word lattices* using N-Grams (bigrams or trigrams).

Word Lattice: a directed graph of words (states) and links (arcs) which encodes a large number of sentences (an extremely poor man's syntax).

Each word carries its observation likelihood, so a particular path in the lattice can be combined with another bigram estimation to make for an even better guess.

Training and Testing

To train a speech recognizer one needs an extremely large corpus (1 billion words maybe).

This corpus must have at least three parts:

1. A subset which is the *training set*.
2. A subset which has not been seen by the recognizer and which is the *test set*.
3. A subset which has been hand annotated as the *gold standard* and against which the performance can be checked.

Word Error Rate

See handout (J&M p. 271).

Next time: speech recognition with the help of linguistic principles and no statistics?