Introduction to Natural Language Processing

Chapter 6

Outline

Administration

- recall errata

Review

Next Word Prediction

Probabilistic Models

The Probability of a Sequence

Training and Testing

The Problem of Zeroes

Revaluating Zero and Low Probability N-grams

Smoothing

Backoff

Review

While FSAs can model many things in NLP, there are limitations (no advice when rejection, all or one solution when accept condition is ambiguous).

So far we've concentrated on individual words in isolation.

Now we're going to start looking at words in *context*.

We'll start with what looks like an artificial task: predicting the next word in a sequence.

Word Prediction Example

From NY Times...

Stocks plunged this ...

Word Prediction Example (cont.)

From NY Times...

Stocks plunged this morning, despite a cut in interest ...

Word Prediction Example (cont.)

From NY Times...

Stocks plunged this morning, despite a cut in interest rates by the Federal Reserve, as Wall ...

Word Prediction Example (cont.)

From NY Times...

Stocks plunged this morning, despite a cut in interest rates by the Federal Reserve, as Wall Street began ...

Word Prediction Example (cont.)

From NY Times...

Stocks plunged this morning, despite a cut in interest rates by the Federal Reserve, as Wall Street began trading for the first time since last ...

Word Prediction Example (cont.)

From NY Times...

Stocks plunged this morning, despite a cut in interest rates by the Federal Reserve, as Wall Street began trading for the first time since last Tuesday's terrorist attacks.

Word Prediction

Clearly, at least some of us have the ability to predict future words in an utterance.

How?

- domain knowledge
- syntactic knowledge
- lexical knowledge

Claim

A useful part of the knowledge needed to allow Word Prediction (guessing the next word) can be captured using simple statistical techniques.

In particular, we'll rely on the notion of the *probability* of a sequence (e.g., sentence).

Why do this?

Why would anyone want to predict a word?

You don't really. But if you say you can predict the next word, it means you can rank the likelihood of sequences containing various alternative words.

- you can assess the likelihood/goodness of a sentence

Statistical Methods Abound in NLP

Resolving part of speech ambiguity

- *fly* can be a noun or a verb
- find the most likely word class, given the surrounding words

Resolving word sense ambiguity

- the noun *bank* can be the side of a river or a financial instituion
- again, find the most likely word class, given the surrounding words

Speech recognition

- find the most likely word sequence given a signal

Corpus Analysis

Statistical methods use large corpora (or databases) of natural language, which have been marked up (or "annotated") with phenemena of interest (e.g., parts of speech, word sense, parse trees).

- annotation must frequently be done by humans

These corpora are then used to find statistics.

- e.g., in 95% of cases, if *fly* follows an article (e.g., *the*, *a*), it is a noun

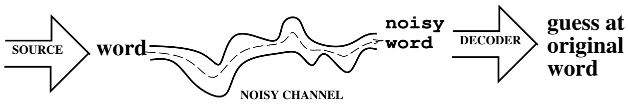
These statistics are then used to analyze new (unannotated) sentences.

- in the sentence *The fly flies*, the most likely part of speech tagging is *art n v*

Essentially we use probability theory to find the likelihood of one interpretation over another, and hence the maximally likely interpretation.

Noisy Channel Model

The Noisy Channel probabilistic model:



Many NLP problems can be modeled as mapping from one string of symbols to another.

In statistical language applications, knowledge of the source (e.g, a statistical model of word sequences) is referred to as a Language Model or a Grammar.

Example applications that employ language models

- speech recognition
- handwriting recognition
- real word spelling correction

Handwriting Recognition Example

Assume a note is given to a bank teller, which the teller reads as *I have a gub.* (example courtesy of a Woody Allen film)

NLP to the rescue ...

- *gub* is not a word
- - *gun* and *gull* are words, but *gun* has a higher probability in the context of a bank

Real Word Spelling Errors

They are leaving in about fifteen *minuets* to go to her house.
The study was conducted mainly *be* John Black.
The design *an* construction of the system will take more than a year.
Hopefully, all *with* continue smoothly in my absence.
Can they *lave* him my messages?
I need to *notified* the bank of [this problem.]
He is trying to *fine* out.

Real Word Spelling Correction Ex.

Collect a list of commonly substituted words

- piece/peace, whether/weather, their/there ...

Whenever you encounter one of these words in a sentence, construct the alternative sentence as well

Assess the goodness of each and choose the one (word) with the more likely sentence

Example

- blah blah blah the whether
- blah blah blah the weather

Basic Probability Background

Prior Probability (or unconditional probability)

- P(A)
- think of A as a proposition, as in some logic
- P(E) = 0.75 means that event E occurs 75 times out of a hundred

Examples

- coin toss
  - P(toss=heads) = 0.5
- lottery
  - P(reward=jackpot) = 0.000001
  - P(reward=nothing) = 0.9
  - P(reward=other) = 0.099999

Basic Probability (cont.)

Conditional Probability

- P(A|B)
- the probability of A given that we know B
- P(E1|E2)=0.6 means that if E2 is the case, then E1 occurs 6 times out of 10

Examples

- P(weather=rain | location=Pittsburgh) = 0.6
- P(reward=jackpot|zodiacsign=pisces)=0.000001

Axioms of Probability

For any propositions, A, B

1. $0 \leq P(A) \leq 1$
2. P(True) = 1 and P(False) = 0
3. P(A V B) = P(A) + P(B) - P(A ∧ B) (draw venn diagram)

If P(A|B) = P(A) (i.e., B doesn't effect the likelihood of A), we say that A and B are <u>independent</u>

- reward=jackpot and zodiacsign=pisces are independent

If two events are independent

- P(A ∧ B) = P(A) X P(B)

Relating Conditionals and Priors

Definition of conditional probability

$$P(A \mid B) = \frac{P(A \wedge B)}{P(B)}$$

Product rules gives an alternative formulation

$$(A \wedge B) = P(A \mid B)P(B)$$

You could also say

$$P(A \wedge B) = P(B \mid A)P(A)$$

More Probability

We can view the probability of a word sequence as the probability of a conjunctive event.

For example, the probability of "the big cat" is

- P(the ∧ big ∧ cat)

Chain Rule

Recall the definition of conditional probability

$$P(A \mid B) = \frac{P(A \wedge B)}{P(B)}$$

Rewriting

$$P(A \wedge B) = P(B \mid A)P(A)$$

Or

$$P(the \wedge dog) = P(dog \mid the)P(the)$$

So, even if "the" is more frequent than "dog", conditional rather than individual relative word frequencies will prefer "the dog" rather than "the the"

$$P(dog \mid the) > P(dog)$$

Similarly, relative word frequencies are better than equal probabilities for all words.

Chain Rule (continued)

In general, the probability of a complete string of words

$$P(w_1 \ldots w_n) =$$
$$P(w_1^n) =$$
$$P(w_1)P(w_2 \mid w_1)P(w_3 \mid w_1^2) \ldots P(w_n \mid w_1^{n-1}) =$$
$$\Pi_{k=1}^{n} P(w_k \mid w_1^{k-1})$$

Chain rule is derived by successive application of product rule

Approximating Things

Unfortunately, that's really not helpful in general.

Why?

Approximating Things (continued)

So we make a Markov Assumption...

$$P(w_n \mid w_1^{n-1}) \approx P(w_n \mid w_{n-N+1}^{n-1})$$

That is, the probability of a word given all previous words is approximated using N previous words.

In other words, we revert to a technique that allows us to collect statistics in *practice*.

For example, the assumption that the probability of a word depends only on the previous word (as in a bigram) is a Markov assumption.

Markov models are the class of probabilistic models that assume that we can predict the probability of some future unit without looking too far into the past.

Counting Words in Corpora

Probabilities are based on counting things, so ...

- what to count?
  - words (this chapter), word classes, word senses, speech acts ...
  - what is a word? (e.g., are *cat* and *cats* the same word?)
- where to find the things to count?
  - Corpora (online collections of text and speech)

Today's Outline

Paper Discussion

Review (word prediction, probability 101)

The Probability of a Sequence

Training and Testing

The Problem of Zeroes

Revaluating Zero and Low Probability N-grams

Smoothing

Backoff

Review

The probability of a sentence is represented as the probability of a given word sequence.

This can be computed using the chain rule.

Approximations using Markov models are less accurate, but it is easier to get reliable statistics.

Simple N-Grams

An N-gram model uses the previous N-1 words to predict the next one:

- $P(w_n \mid w_{n-1})$

We'll pretty much always be dealing with $P(word \mid some\ prefix)$

- unigrams (n=1): $P(dog)$
- bigrams (n=2): $P(dog \mid big)$
- trigrams (n=3): $P(dog \mid the\ big)$
- quadrigrams (n=4): $P(dog \mid chasing\ the\ big)$

Computing a String's Probability

The probability of a string of words such as "I want to eat Chinese food", that is P(I want to eat Chinese food)

- P(I,want,to,eat,Chinese,food)

We can decompose this using the chain rule

- P(I) P(want | I) P(to | I want) ... P(food | I want to eat Chinese)

So how do we estimate probabilities like P(to | I want)

Using N-Grams

Approximate the probability of a word given all the previous words, by just using some of them

Recall

$$P(w_n \mid w_1^{n-1}) \approx P(w_n \mid w_{n-N+1}^{n-1})$$

For a bigram grammar

$$P(w_1^n) \approx \Pi_{k=1}^n P(w_k \mid w_{k-1})$$

Thus, P(sentence) can be computed by multiplying bigram probabilities

- P(I want to each Chinese food) = P(I | "sentence start")P(want | I)P(to | want)P(eat | to)P(Chinese | eat)P(food | Chinese)

A 2nd order Markov model (trigrams) gets us (omitting sentence start)

- P(I)P(want | I)P(to | I want)P(eat | want to)P(Chinese | to eat)P(food | eat Chinese)

BERP: Berkely Restaurant Project

Example bigram grammar fragment:

eat on	.16	eat Thai	.03
eat some	.06	eat breakfast	.03
eat lunch	.06	eat in	.02
eat dinner	.05	eat Chinese	.02
eat at	.04	eat Mexican	.02
eat a	.04	eat tomorrow	.01
eat Indian	.04	eat dessert	.007
eat today	.03	eat British	.001

Probabilities seem to capture "syntactic" facts, "world knowledge"

- "eat" is often followed by a NP
- British food is not too popular

N-gram models can be trained by Counting and Normalization

BERP Bigram Counts

	I	want	to	eat	Chinese	food	lunch
I	8	1087	0	13	0	0	0
want	3	0	786	0	6	8	6
to	3	0	10	860	3	0	12
eat	0	0	2	0	19	2	52
Chinese	2	0	0	0	0	120	1
food	19	0	17	0	0	0	0
lunch	4	0	0	0	0	1	0

Note all the zeroes even with a coherent word set.

To get probabilities, we must normalize to get the values between 0 and 1.

BERP Bigram Probabilities

Normalization: divide each row's counts by appropriate unigram counts (that is, divide each bigram count by the number of times the first word of that bigram appears)

	I	want	to	eat	Chinese	food	lunch
I	.0023	.32	0	.0038	0	0	0
want	.0025	0	.65	0	.0049	.0066	.0049
to	.00092	0	.0031	.26	.00092	0	.0037
eat	0	0	.0021	0	.020	.0021	.055
Chinese	.0094	0	0	0	0	.56	.0047
food	.013	0	.011	0	0	0	0
lunch	.0087	0	0	0	0	.0022	0

Computing the probability of "I I"

- "I" unigram count = 8 + 1087 + 13 ... = 3437
- "I I" bigram count = 8
- p(I | I) = 8 / 3437 = .0023

A bigram grammar can be represented as an N-by N matrix of probabilities, where N is the vocabulary size.

Observations on these Probabilities

What's being captured with ...

- P(want | I) = .32
- P(to | want) = .65
- P(eat | to) = .26
- P(food | Chinese) = .56
- P(lunch | eat) = .055

What about...

- P(I | I) = .0023
- P(I | want) = .0025
- P(I | food) = .013

Observations (continued)

P(I | I) = .0023

- I I I I want ...

P(I | want) = .0025

- I want I want ...

P(I | food) = .013

- the kind of food I want is ...

Approximating Shakespeare

As we increase the value of N, the accuracy of the n-gram model increases.

Unigrams

- *Every enter now severally so, let*
- *Hill he late speaks; or! a more to leg less first you enter*

Bigrams

- *What means, sir. I confess she? then all sorts, he is trim, captain.*
- *Why dost stand forth thy canopy, forsooth; he is this palpable hit the King Henry.*

Approximating Shakespeare (cont.)

Trigrams

- *Sweet prince, Falstaff shall die.*
- *This shall forbid it should be branded, if renown made it empty.*

Quadrigrams

- *What! I will go seek the traitor Gloucester.*
- *Will you not tell me who I am?*

Approximating Shakespeare (cont.)

There are 884,647 tokens, with 29,066 word form types, in about a one million word Shakespeare corpus.

Shakespeare produced 300,000 bigram types out of 844 million possible bigrams. Thus, 99.96 % of the possible bigrams were never seen (have zero entries in the table).

The quadrigrams are worse. What's coming out looks like Shakespeare because it is Shakespeare.

All those zeroes are causing problems.

Some Useful Empirical Observations

A small number of events occur with high frequency.

A large number of events occur with low frequency.

You can quickly collect statistics on the high frequency events.

You might have to wait an arbitrarily long time to get valid statistics on low frequency events.

Some of the zeroes in the table are really zeroes. Some are simply low frequency events you haven't seen yet.

Another Example

Jim Martin did the following...

```
% deroff -w /usr/man/man1/csh.1 | sort |
uniq -c | sort -r
```

And got 1626 unique types out of about 11,000 tokens.

The top seven in terms of frequency:

- 709 the
- 304 is
- 275 to
- 250 of
- 192 and
- 188 command
- 184 csh

But 740 of the 1626 only occur once!

Example (continued)

Even if we add in more and more man pages...

- very few/no additional new high frequency types. But lots of additions to the counts for the existing high frequency types.
- more and more new single instance types

N-Gram Training Sensitivity

If we repeated the Shakespeare experiment but trained on a Wall Street Journal corpus, there would be little overlap in the output.

- implications for corpus design

Methodology: Training and Testing

Probabilities come from a Training portion of a corpus, which is used to design the model.

- overly narrow corpus: probabilities don't generalize
- overly general corpus: probabilities don't reflect task or domain

A separate Test portion of the corpus is used to *evaluate* the model, typically using standard *metrics.*

- held out test set
- cross validation
- evaluation differences should be statistically significant

The Problem of Zeroes

Zero counts (sparse matrices) are a problem.

In random generation, this limits the generator to output strings that actually occur in the training corpus. Boring, although not really important.

In analysis, when a system is asked to assess the probability of an input string that contains an as yet unseen event, it has to assign a probability of zero to it. This will lead to poor test results.

Overcoming Zeroes

Don't use higher order N-grams

Smoothing

- add-one
- Witten-Bell

Backoff

- Katz

Add-One Smoothing

Recall bigrams

$$P(w_n \mid w_{n-1}) = \frac{C(w_{n-1}w_n)}{C(w_{n-1})}$$

A simple idea: Add 1 to all the counts.

Rationale: Zero count events are just single count events you haven't seen yet because your corpus wasn't big enough.

$$P_i = \frac{c(i)}{N}$$

$$P_i^* = \frac{c(i)+1}{N+V}$$

N = total number of word tokens

V = total number of word types (vocabulary size)

For bigrams

$$P^*(w_n \mid w_{n-1}) = \frac{C(w_{n-1}w_n)+1}{C(w_{n-1})+V}$$

Original vs. Add-One Bigram Counts

	I	want	to	eat	Chinese	food	lunch
I	8	1087	0	13	0	0	0
want	3	0	786	0	6	8	6
to	3	0	10	860	3	0	12
eat	0	0	2	0	19	2	52
Chinese	2	0	0	0	0	120	1
food	19	0	17	0	0	0	0
lunch	4	0	0	0	0	1	0

	I	want	to	eat	Chinese	food	lunch
I	9	1088	1	14	1	1	1
want	4	1	787	1	7	9	7
to	4	1	11	861	4	1	13
eat	1	1	3	1	20	3	53
Chinese	3	1	1	1	1	121	2
food	20	1	18	1	1	1	1
lunch	5	1	1	1	1	2	1

Add-One Bigram Probabilities

	I	want	to	eat	Chinese	food	lunch
I	.0018	.22	.00020	.0028	.00020	.00020	.00020
want	.0014	.00035	.28	.00035	.0025	.0032	.0025
to	.00082	.00021	.0023	.18	.00082	.00021	.0027
eat	.00039	.00039	.0012	.00039	.0078	.0012	.021
Chinese	.0016	.00055	.00055	.00055	.00055	.066	.0011
food	.0064	.00032	.0058	.00032	.00032	.00032	.00032
lunch	.0024	.00048	.00048	.00048	.00048	.00096	.00048

"I" unigram count = 3437

V = 1616,

Thus, $\frac{8+1}{3437+1616} = .0018$

Effect of Smoothing

	I	want	to	eat	Chinese	food	lunch
I	.0023	.32	0	.0038	0	0	0
want	.0025	0	.65	0	.0049	.0066	.0049
to	.00092	0	.0031	.26	.00092	0	.0037
eat	0	0	.0021	0	.020	.0021	.055
Chinese	.0094	0	0	0	0	.56	.0047
food	.013	0	.011	0	0	0	0
lunch	.0087	0	0	0	0	.0022	0

	I	want	to	eat	Chinese	food	lunch
I	.0018	.22	.00020	.0028	.00020	.00020	.00020
want	.0014	.00035	.28	.00035	.0025	.0032	.0025
to	.00082	.00021	.0023	.18	.00082	.00021	.0027
eat	.00039	.00039	.0012	.00039	.0078	.0012	.021
Chinese	.0016	.00055	.00055	.00055	.00055	.066	.0011
food	.0064	.00032	.0058	.00032	.00032	.00032	.00032
lunch	.0024	.00048	.00048	.00048	.00048	.00096	.00048

- too much probability mass is moved to zeros
- the count augmentation of 1 was arbitrary

Witten-Bell

Things Seen Once: Use the count of things you've seen once to help estimate the count of things you've never seen.

The basic idea of Witten-Bell is to use the frequency with which previously unseen (first-time) N-grams appear as a guess as to their actual proportion in the overall data.

In any given corpus the number of previously unseen N-grams is just the number of types, T.

So we'll guess as the total probability of future unseen events ...

$$\Sigma_{i:c_i=0} p_i^* = \frac{T}{N+T}$$

... by estimating the total probability mass of all zero N-grams, by taking the number of observed types divided by the number of tokens plus observed types.

Note that T (observed types) is different than our previous V (possible types).

Witten-Bell (continued)

But that's the total probability of future unseen events.

We need the probability to plug in for each currently zero entry in the table.

Just divide it up equally among all the zeros.

$$Z = \Sigma_{i:c_i=0} 1$$

$$p_i^* = \frac{T}{Z(N+T)}$$

Discounting

We can't just up all the zeroes to these counts. Things still have to sum to 1. So we have to discount the existing counts by the amount we gave to the zeroes.

If $c_i > 0$,

$$p_i^* = \frac{c_i}{N+T}$$

Witten-Bell for Bigrams

In the bigram case, everybody shouldn't get the same low estimate.

Some words are more likely to be precursors of new words than others. We can estimate the proclivity of a word to be followed by a new word by doing just we just did but condition on the preceding word.

In English, look at each word and see how many new words it brought into the corpus (the bigram type count for that word). Divide that by the number of total events that word participated in.

Thus, total probability mass for zeros...

$$\Sigma_{i:c(w_x w_i)=0} p^*(w_i \mid w_x) = \frac{T(w_x)}{N(w_x) + T(w_x)}$$

The amount available for each bigram is then distributed and discounted similarly to before.

Add-One vs. Witten-Bell Discounts

I .68 .97
want .42 .94
to .69 .96
eat .37 .88
Chinese .12 .91
food .48 .94
lunch .22 .91

Backoff Methods

Recall that we're trying to find numbers for the probability of some particular word in the context of some preceding words (N-grams), and have been worrying about what to do when we have zero counts.

Unless it's a word we've never seen before, there are other probabilities we can fall back on.

For example... if we've never seen $P(dog \mid smelly)$ why not use $P(dog)$.

But to make this work

- discount so it all adds up (can't replace a zero with something higher without reestimating the non-zeroes)

Katz Backoff

If we encounter a zero N-gram, back off to N-1.

If that's zero, backoff to N-2.

And so on.

For trigrams...

- $P(w_i \mid w_{i-2}w_{i-1})$, if $C(w_{i-2}w_{i-1}w_i) > 0$
- $\alpha_1 P(w_i \mid w_{i-1})$, if $C(w_{i-2}w_{i-1}w_i) = 0$ and $C(w_{i-1}w_i) > 0$
- $\alpha_2 P(w_i)$, otherwise

That is, we only "back off" to a lower order N-gram if we have zero evidence for a higher order.

Witten-Bell with Katz backoff is a popular combination for speech recognition language models.

Which Method is Best?

Each method is based on an observation about various facts about language and word frequencies. None of the observations are wrong but they lead to models that produce different results.

In the end it's all very empirical. The right method depends on the size and nature of the corpus, and the needs of the application. There's no "best" method for all situations.

For Next Time

Chapter 8