Natural Language Processing

Lecture 8

Today's Outline

Review

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

- P(I want to eat Chinese food)

The chain rule gets us

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

A 2nd order Markov model (trigrams) gets us

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

This is less accurate, but it is easier to get reliable statistics

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.

Adding Discounting

Why did we just go to all the trouble of coming up with numbers for zero events, if now we are ignoring them? Why not use Witten-Bell somehow?

We do... it's used to figure out the discounting. In particular, we use it to decide the probability mass to take away from the original model, and to make sure that the lower order N-grams get into the right range (sum up proportionally to the amount that we're taking away from the N-grams).

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

Backoff with Discounting

For trigrams...

- $P_{discounted}(w_i \mid w_{i-2}w_{i-1})$, if $C(w_{i-2}w_{i-1}w_i) > 0$
- $\alpha(w_{i-2}^{i-1})P_{discounted}(w_i \mid w_{i-1})$, if $C(w_{i-2}w_{i-1}w_i) = 0$ and $C(w_{i-1}w_i) > 0$
- $\alpha(w_{i-1})P_{discounted}(w_i)$, otherwise

(note this is wrong in book - see errata!)

α ensures that the probability mass from all the lower order N-grams sums up to the amount saved by discounting the higher order N-grams.

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

Acknowledgements

Some of this material was taken from the course web pages of James Martin.