Kneser-Ney smoothing explained

18 January 2014

Language models are an essential element of natural language processing, central to tasks ranging from spellchecking to machine translation. Given an arbitrary piece of text, a language model determines whether that text belongs to a given language.

We can give a concrete example with a **probabilistic language model**, a specific construction which uses probabilities to estimate how likely any given string belongs to a language. Consider a probabilistic English language model P_E. We would expect the probability

$$P_E(\text{I went to the store})$$

to be quite high, since we can confirm this is valid English. On the other hand, we expect the probabilities

$$P_E(\text{store went to I the}), P_E(\text{Ich habe eine Katz})$$

to be very low, since these fragments do not constitute proper English text.

I don't aim to cover the entirety of language models at the moment — that would be an ambitious task for a single blog post. If you haven't encountered language models or *n*-grams before, I recommend the following resources:

- "Language model" on Wikipedia
- Chapter 4 of Jurafsky and Martin's *Speech and Language Processing*
- Chapter 7 of *Statistical Machine Translation* (see summary slides online)

I'd like to jump ahead to a trickier subject within language modeling known as **Kneser-Ney smoothing**. This smoothing method is most commonly applied in an *interpolated* form,[1] and this is the form that I'll present today.

Kneser-Ney evolved from **absolute-discounting interpolation**, which makes use of both higher-order (i.e., higher-*n*) and lower-order language models, reallocating some probability mass from 4-grams or 3-grams to simpler unigram models. The formula for absolute-discounting smoothing as applied to a bigram language model is presented below:

$$P_{abs}(w_i \mid w_{i-1}) = \frac{\max(c(w_{i-1}w_i) - \delta, 0)}{\sum_{w'} c(w_{i-1}w')} + \alpha\, p_{abs}(w_i)$$

Here δ refers to a fixed **discount** value, and α is a normalizing constant. The details of this smoothing are covered in Chen and Goodman (1999).

The essence of Kneser-Ney is in the clever observation that we can take advantage of this interpolation as a sort of backoff model. When the first term (in this case, the discounted relative bigram count) is near zero, the second term (the lower-order model) carries more weight. Inversely, when the higher-order model matches strongly, the second lower-order term has little weight.

The Kneser-Ney design retains the first term of absolute discounting interpolation, but rewrites the second term to take advantage of this relationship. Whereas absolute discounting interpolation in a bigram model would simply default to a unigram model in the second term, Kneser-Ney depends upon the idea of a *continuation probability* associated with each unigram.

This probability for a given token w_i is proportional to the **number of bigrams which it completes**:

$$P_{\text{continuation}}(w_i) \propto \; |\{w_{i-1} : c(w_{i-1}, w_i) > 0\}|$$

This quantity is normalized by dividing by the total number of bigram types (note that j is a free variable):

$$P_{\text{continuation}}(w_i) = \frac{|\{w_{i-1} : c(w_{i-1}, w_i) > 0\}|}{|\{w_{j-1} : c(w_{j-1}, w_j) > 0\}|}$$

The common example used to demonstrate the efficacy of Kneser-Ney is the phrase *San Francisco*. Suppose this phrase is abundant in a given training corpus. Then the unigram probability of *Francisco* will also be high. If we unwisely use something like absolute discounting interpolation in a context where our bigram model is weak, the unigram model portion may take over and lead to some strange results.

Dan Jurafsky gives the following example context:

> I can't see without my reading ______.

A fluent English speaker reading this sentence knows that the word *glasses* should fill in the blank. But since *San Francisco* is a common term, absolute-discounting interpolation might declare that *Francisco* is a better fit: $P_{abs}(\text{Francisco}) > P_{abs}(\text{glasses})$.

Kneser-Ney fixes this problem by asking a slightly harder question of our lower-order model. Whereas the unigram model simply provides how likely a word w_i is to appear, Kneser-Ney's second term determines how likely a word w_i is to appear in an unfamiliar bigram context.

Kneser-Ney in whole follows:

$$P_{KN}(w_i \mid w_{i-1}) = \frac{\max(c(w_{i-1}w_i) - \delta, 0)}{\sum_{w'} c(w_{i-1}w')} + \lambda \frac{|\{w_{i-1} : c(w_{i-1}, w_i) > 0\}|}{|\{w_{j-1} : c(w_{j-1}, w_j) > 0\}|}$$

λ is a normalizing constant

$$\lambda(w_{i-1}) = \frac{\delta}{c(w_{i-1})} |\{w' : c(w_{i-1}, w') > 0\}| .$$

Note that the denominator of the first term can be simplified to a unigram count. Here is the final interpolated Kneser-Ney smoothed bigram model, in all its glory:

$$P_{KN}(w_i \mid w_{i-1}) = \frac{\max(c(w_{i-1}w_i) - \delta, 0)}{c(w_{i-1})} + \lambda \frac{|\{w_{i-1} : c(w_{i-1}, w_i) > 0\}|}{|\{w_{j-1} : c(w_{j-1}, w_j) > 0\}|}$$

Further reading

If you enjoyed this post, here is some further reading on Kneser-Ney and other smoothing methods:

- Bill MacCartney's smoothing tutorial (very accessible)
- Chen and Goodman (1999)
- Section 4.9.1 in Jurafsky and Martin's *Speech and Language Processing*

Footnotes

1. For the canonical definition of interpolated Kneser-Ney smoothing, see S. F. Chen and J. Goodman, "An empirical study of smoothing techniques for language modeling," Computer Speech and Language, vol. 13, no. 4, pp. 359–394, 1999. ↩

4 Comments **foldl** **Disqus' Privacy Policy** 1

Favorite 3 **Sort**

Join the discussion…

LOG IN WITH

OR SIGN UP WITH DISQUS ?

Name

No name! • 6 years ago • edited

Could explain what difference does the δ make here?

In one of the terms we have: max(c(w_(i−1)w_i)−δ,0), but if δ lies 0 < δ < 1 and since c(w_(i−1)w be as big as hundreds, the difference is pretty low. So I am not sure if I understand it properly.

^ | ˅ • Reply •