NLP Lunch Tutorial: Smoothing

Bill MacCartney

21 April 2005

Preface

- Everything is from this great paper by Stanley F. Chen and Joshua Goodman (1998), "An Empirical Study of Smoothing Techniques for Language Modeling", which I read yesterday.
- Everything is presented in the context of n-gram language models, but smoothing is needed in many problem contexts, and most of the smoothing methods we'll look at generalize without difficulty.

The Plan

- Motivation
  - the problem
  - an example

- All the smoothing methods
  - formula after formula
  - intuitions for each

- So which one is the best?
  - (answer: modified Kneser-Ney)

- Excel "demo" for absolute discounting and Good-Turing?

Probabilistic modeling

- You have some kind of probabilistic model, which is a distribution $p(e)$ over an event space E.
- You want to estimate the parameters of your model distribution p from data.
- In principle, you might to like to use maximum likelihood (ML) estimates, so that your model is

$$p_{ML}(x) = \frac{c(x)}{\sum_e c(e)}$$

But...

Problem: data sparsity

- But, you have insufficient data: there are many events x such that $c(x) = 0$, so that the ML estimate is $p_{ML}(x) = 0$.
- In problem settings where the event space E is unbounded (e.g. most NLP problems), this is generally undesirable.
- Ex: a language model which gives probability 0 to unseen words.
- Just because an event has never been observed in training data does not mean it cannot occur in test data.
- So if $c(x) = 0$, what should $p(x)$ be?
- *If data sparsity isn't a problem for you, your model is too simple!*

"Whenever data sparsity is an issue, smoothing can help performance, and data sparsity is almost always an issue in statistical modeling. In the extreme case where there is so much training data that all parameters can be accurately trained without smoothing, one can almost always expand the model, such as by moving to a higher n-gram model, to achieve improved performance. With more parameters data sparsity becomes an issue again, but with proper smoothing the models are usually more accurate than the original models. Thus, no matter how much data one has, smoothing can almost always help performace, and for a relatively small effort."

Chen & Goodman (1998)

Example: bigram model

JOHN READ MOBY DICK
MARY READ A DIFFERENT BOOK
SHE READ A BOOK BY CHER

$$p(w_i|w_{i-1}) = \frac{c(w_{i-1}w_i)}{\sum_{w_i} c(w_{i-1}w_i)}$$

$$p(s) = \prod_{i=1}^{l+1} p(w_i|w_{i-1})$$

JOHN READ MOBY DICK
MARY READ A DIFFERENT BOOK
SHE READ A BOOK BY CHER

$p(\text{JOHN READ A BOOK})$

$$
\begin{aligned}
&= p(\text{JOHN}|\bullet)\; p(\text{READ}|\text{JOHN})\; p(\text{A}|\text{READ})\; p(\text{BOOK}|\text{A})\; p(\bullet|\text{BOOK}) \\
&= \frac{c(\bullet\ \text{JOHN})}{\sum_w c(\bullet\ w)}\; \frac{c(\text{JOHN READ})}{\sum_w c(\text{JOHN}\ w)}\; \frac{c(\text{READ A})}{\sum_w c(\text{READ}\ w)}\; \frac{c(\text{A BOOK})}{\sum_w c(\text{A}\ w)}\; \frac{c(\text{BOOK}\ \bullet)}{\sum_w c(\text{BOOK}\ w)} \\
&= \frac{1}{3}\; \frac{1}{1}\; \frac{2}{3}\; \frac{1}{2}\; \frac{1}{2} \\
&\approx 0.06
\end{aligned}
$$

JOHN READ MOBY DICK
MARY READ A DIFFERENT BOOK
SHE READ A BOOK BY CHER

$p(\text{CHER READ A BOOK})$

$$= p(\text{CHER}|\bullet)\ p(\text{READ}|\text{CHER})\ p(\text{A}|\text{READ})\ p(\text{BOOK}|\text{A})\ p(\bullet|\text{BOOK})$$

$$= \frac{c(\bullet\ \text{CHER})}{\sum_w c(\bullet\ w)}\ \frac{c(\text{CHER READ})}{\sum_w c(\text{CHER}\ w)}\ \frac{c(\text{READ A})}{\sum_w c(\text{READ}\ w)}\ \frac{c(\text{A BOOK})}{\sum_w c(\text{A}\ w)}\ \frac{c(\text{BOOK}\ \bullet)}{\sum_w c(\text{BOOK}\ w)}$$

$$= \frac{0}{3}\ \frac{0}{1}\ \frac{2}{3}\ \frac{1}{2}\ \frac{1}{2}$$

$$= 0$$

Add-one smoothing

$$p(w_i|w_{i-1}) = \frac{1 + c(w_{i-1}w_i)}{\sum_{w_i}[1 + c(w_{i-1}w_i)]} = \frac{1 + c(w_{i-1}w_i)}{|V| + \sum_{w_i} c(w_{i-1}w_i)}$$

- Originally due to Laplace.
- Typically, we assume $V = \{w : c(w) > 0\} \cup \{\text{UNK}\}$
- Add-one smoothing is generally a horrible choice.

JOHN READ MOBY DICK
MARY READ A DIFFERENT BOOK
SHE READ A BOOK BY CHER

$p(\text{JOHN READ A BOOK})$

$$= \frac{1+1}{11+3} \quad \frac{1+1}{11+1} \quad \frac{1+2}{11+3} \quad \frac{1+1}{11+2} \quad \frac{1+1}{11+2}$$

$$\approx 0.0001$$

$p(\text{CHER READ A BOOK})$

$$= \frac{1+0}{11+3} \quad \frac{1+0}{11+1} \quad \frac{1+2}{11+3} \quad \frac{1+1}{11+2} \quad \frac{1+1}{11+2}$$

$$\approx 0.00003$$

Smoothing methods

- Additive smoothing
- Good-Turing estimate
- Jelinek-Mercer smoothing (interpolation)
- Katz smoothing (backoff)
- Witten-Bell smoothing
- Absolute discounting
- Kneser-Ney smoothing

Additive smoothing

$$p_{add}(w_i|w_{i-n+1}^{i-1}) = \frac{\delta + c(w_{i-n+1}^{i})}{\delta|V| + \sum_{w_i} c(w_{i-n+1}^{i})}$$

- Idea: pretend we've seen each n-gram δ times more than we have.
- Typically, $0 < \delta \leq 1$.
- Lidstone and Jeffreys advocate $\delta = 1$.
- Gale & Church (1994) argue that this method performs poorly.

Good-Turing estimation

- Idea: reallocate the probability mass of n-grams that occur $r+1$ times in the training data to the n-grams that occur r times.
- In particular, reallocate the probability mass of n-grams that were seen once to the n-grams that were never seen.
- For each count r, we compute an adjusted count r^*:

$$r^* = (r+1)\frac{n_{r+1}}{n_r}$$

  where n_r is the number of n-grams seen exactly r times.
- Then we have:

$$p_{GT}(x : c(x) = r) = \frac{r^*}{N}$$

  where $N = \sum_{r=0}^{\infty} r^* n_r = \sum_{r=1}^{\infty} r n_r$.

Good-Turing problems

- Problem: what if $n_{r+1} = 0$? This is common for high r: there are "holes" in the counts of counts.
- Problem: even if we're not just below a hole, for high r, the n_r are quite noisy.
- Really, we should think of r^* as:

$$r^* = (r+1)\frac{E[n_{r+1}]}{E[n_r]}$$

- But how do we estimate that expectation? (The original formula amounts to using the ML estimate.)
- Good-Turing thus requires elaboration to be useful. It forms a foundation on which other smoothing methods build.

Jelinek-Mercer smoothing (interpolation)

- Observation: If $c(\text{BURNISH THE}) = 0$ and $c(\text{BURNISH THOU}) = 0$, then under both additive smoothing and Good-Turing:

$$p(\text{THE}|\text{BURNISH}) = p(\text{THOU}|\text{BURNISH})$$

- This seems wrong: we should have

$$p(\text{THE}|\text{BURNISH}) > p(\text{THOU}|\text{BURNISH})$$

  because THE is much more common than THOU

- Solution: *interpolate* between bigram and unigram models.

Jelinek-Mercer smoothing (interpolation)

- Unigram ML model:

$$p_{ML}(w_i) = \frac{c(w_i)}{\sum_{w_i} c(w_i)}$$

- Bigram interpolated model:

$$p_{interp}(w_i|w_{i-1}) = \lambda p_{ML}(w_i|w_{i-1}) + (1-\lambda) p_{ML}(w_i)$$

Jelinek-Mercer smoothing (interpolation)

- Recursive formulation: nth-order smoothed model is defined recursively as a linear interpolation between the nth-order ML model and the $(n-1)$th-order smoothed model.

$$p_{interp}(w_i|w_{i-n+1}^{i-1}) = \lambda_{w_{i-n+1}^{i-1}} p_{ML}(w_i|w_{i-n+1}^{i-1}) + (1 - \lambda_{w_{i-n+1}^{i-1}}) p_{interp}(w_i|w_{i-n+2}^{i-1})$$

- Can ground recursion with:
  - 1st-order model: ML (or otherwise smoothed) unigram model
  - 0th-order model: uniform model

$$p_{unif}(w_i) = \frac{1}{|V|}$$

Jelinek-Mercer smoothing (interpolation)

- The $\lambda_{w_{i-n+1}^{i-1}}$ can be estimated using EM on held-out data (*held-out interpolation*) or in cross-validation fashion (*deleted interpolation*).
- The optimal $\lambda_{w_{i-n+1}^{i-1}}$ depend on context: high-frequency contexts should get high λs.
- But, can't tune all λs separately: need to bucket them.
- Bucket by $\sum_{w_i} c(w_{i-n+1}^{i})$: total count in higher-order model.

Katz smoothing: bigrams

- As in Good-Turing, we compute adjusted counts.
- Bigrams with nonzero count r are discounted according to *discount ratio* d_r, which is approximately $\frac{r^*}{r}$, the discount predicted by Good-Turing. (Details below.)
- Count mass subtracted from nonzero counts is redistributed among the zero-count bigrams according to next lower-order distribution (i.e. the unigram model).

Katz smoothing: bigrams

- Katz adjusted counts:

$$c_{katz}(w_{i-1}^{i}) = \begin{cases} d_r r & \text{if } r > 0 \\ \alpha(w_{i-1}) p_{ML}(w_i) & \text{if } r = 0 \end{cases}$$

- $\alpha(w_{i-1})$ is chosen so that $\sum_{w_i} c_{katz}(w_{i-1}^{i}) = \sum_{w_i} c(w_{i-1}^{i})$:

$$\alpha(w_{i-1}) = \frac{1 - \sum_{w_i : c(w_{i-1}^{i}) > 0} p_{katz}(w_i | w_{i-1})}{1 - \sum_{w_i : c(w_{i-1}^{i}) > 0} p_{ML}(w_i)}$$

- Compute $p_{katz}(w_i | w_{i-1})$ from corrected count by normalizing:

$$p_{katz}(w_i | w_{i-1}) = \frac{c_{katz}(w_{i-1}^{i})}{\sum_{w_i} c_{katz}(w_{i-1}^{i})}$$

Katz smoothing

- What about d_r? Large counts are taken to be reliable, so $d_r = 1$ for $r > k$, where Katz suggests $k = 5$. For $r \leq k$...
- We want discounts to be proportional to Good-Turing discounts:

$$1 - d_r = \mu(1 - \frac{r^*}{r})$$

- We want the total count mass saved to equal the count mass which Good-Turing assigns to zero counts:

$$\sum_{r=1}^{k} n_r(1 - d_r)r = n_1$$

- The unique solution is:

$$d_r = \frac{\frac{r^*}{r} - \frac{(k+1)n_{k+1}}{n_1}}{1 - \frac{(k+1)n_{k+1}}{n_1}}$$

Katz smoothing

- Katz smoothing for higher-order n-grams is defined analogously.
- Like Jelinek-Mercer, can be given a recursive formulation: the Katz n-gram model is defined in terms of the Katz $(n-1)$-gram model.

Witten-Bell smoothing

- An instance of Jelinek-Mercer smoothing:

$$p_{WB}(w_i|w_{i-n+1}^{i-1}) = \lambda_{w_{i-n+1}^{i-1}} p_{ML}(w_i|w_{i-n+1}^{i-1}) + (1 - \lambda_{w_{i-n+1}^{i-1}}) p_{WB}(w_i|w_{i-n+2}^{i-1})$$

- Motivation: interpret $\lambda_{w_{i-n+1}^{i-1}}$ as the probability of using the higher-order model.
- We should use higher-order model if n-gram w_{i-n+1}^{i} was seen in training data, and back off to lower-order model otherwise.
- So $1 - \lambda_{w_{i-n+1}^{i-1}}$ should be the probability that a word not seen after w_{i-n+1}^{i-1} in training data occurs after that history in test data.
- Estimate this by the number of unique words that follow the history w_{i-n+1}^{i-1} in the training data.

Witten-Bell smoothing

- To compute the λs, we'll need the number of unique words that follow the history w_{i-n+1}^{i-1}:

$$N_{1+}(w_{i-n+1}^{i-1}\ \bullet) = |\{w_i : c(w_{i-n+1}^{i-1} w_i) > 0\}|$$

- Set λs such that

$$1 - \lambda_{w_{i-n+1}^{i-1}} = \frac{N_{1+}(w_{i-n+1}^{i-1}\ \bullet)}{N_{1+}(w_{i-n+1}^{i-1}\ \bullet) + \sum_{w_i} c(w_{i-n+1}^{i})}$$

Absolute discounting

- Like Jelinek-Mercer, involves interpolation of higher- and lower-order models.
- But instead of multiplying the higher-order p_{ML} by a λ, we subtract a fixed discount $\delta \in [0, 1]$ from each nonzero count:

$$p_{abs}(w_i|w_{i-n+1}^{i-1}) = \frac{\max\{c(w_{i-n+1}^{i}) - \delta, 0\}}{\sum_{w_i} c(w_{i-n+1}^{i})} + (1 - \lambda_{w_{i-n+1}^{i-1}}) p_{abs}(w_i|w_{i-n+2}^{i-1})$$

- To make it sum to 1:

$$1 - \lambda_{w_{i-n+1}^{i-1}} = \frac{\delta}{\sum_{w_i} c(w_{i-n+1}^{i})} N_{1+}(w_{i-n+1}^{i-1} \bullet)$$

- Choose δ using held-out estimation.

Kneser-Ney smoothing

- An extension of absolute discounting with a clever way of constructing the lower-order (backoff) model.
- Idea: the lower-order model is signficant only when count is small or zero in the higher-order model, and so should be optimized for that purpose.
- Example: suppose "San Francisco" is common, but "Francisco" occurs only after "San".
- "Francisco" will get a high unigram probability, and so absolute discounting will give a high probability to "Francisco" appearing after novel bigram histories.
- Better to give "Francisco" a low unigram probability, because the only time it occurs is after "San", in which case the bigram model fits well.

Kneser-Ney smoothing

- Let the count assigned to each unigram be the number of different words that it follows. Define:

$$N_{1+}(\bullet\ w_i) = |\{w_{i-1} : c(w_{i-1}w_i) > 0\}|$$

$$N_{1+}(\bullet\ \bullet) = \sum_{w_i} N_{1+}(\bullet\ w_i)$$

- Let lower-order distribution be:

$$p_{KN}(w_i) = \frac{N_{1+}(\bullet\ w_i)}{N_{1+}(\bullet\ \bullet)}$$

- Put it all together:

$$p_{KN}(w_i|w_{i-n+1}^{i-1}) =$$
$$\frac{\max\{c(w_{i-n+1}^{i}) - \delta, 0\}}{\sum_{w_i} c(w_{i-n+1}^{i})} + \frac{\delta}{\sum_{w_i} c(w_{i-n+1}^{i})} N_{1+}(w_{i-n+1}^{i-1}\ \bullet) p_{KN}(w_i|w_{i-n+2}^{i-1})$$

Interpolation vs. backoff

- Both interpolation (Jelinek-Mercer) and backoff (Katz) involve combining information from higher- and lower-order models.
- Key difference: in determining the probability of n-grams with nonzero counts, interpolated models use information from lower-order models while backoff models do not.
- (In both backoff and interpolated models, lower-order models are used in determining the probability of n-grams with zero counts.)
- It turns out that it's not hard to create a backoff version of an interpolated algorithm, and vice-versa. (Kneser-Ney was originally backoff; Chen & Goodman made interpolated version.)

Modified Kneser-Ney

- Chen and Goodman introduced *modified Kneser-Ney*:
  - Interpolation instead of backoff.
  - Single discount for all counts.
  - Uses a separate discount for one- and two-counts instead of a single discount for all counts.
  - Estimates discounts based on held-out data instead of using a formula based on training counts.
- Experiments show all three modifications improve performance.
- Modified Kneser-Ney consistently had best performance.

Conclusions

- The factor with the largest influence is the use of a modified backoff distribution as in Kneser-Ney smoothing.
- Jelinek-Mercer performs better on small training sets; Katz performs better on large training sets.
- Katz smoothing performs well on n-grams with large counts; Kneser-Ney is best for small counts.
- Absolute discounting is superior to linear discounting.
- Interpolated models are superior to backoff models for low (nonzero) counts.
- Adding free parameters to an algorithm and optimizing these parameters on held-out data can improve performance.

Adapted from Chen & Goodman (1998)

END