CMPT 825: Natural Language Processing **Spring 2008**

Lecture 8 — Feb 27, 2007

Lecturer: Anoop Sarkar *Scribe: Winona Wu*

8.1 Language Modeling

A *language model* is usually forumlated as a probability distribution $p(s)$ over string s that attempts to reflect how frequently a string s occurs as a sentense. The most widesly use lanugae models, by far, are n-gram language models. We notice that for a sentence s composed of words $w_1 \cdots w_l$, without loss of generality we can express $p(s)$ as

$$p(s) = p(w_1)p(w_2|w_1)p(w_3|w_1w_2)\cdots p(w_l|w_1\cdots w_{l-1}) = \prod_{i=1}^{l} p(w_i|w_1\cdots w_{i-1})$$

In bigram models, we make the approximation that the probability of a word depends only on the immediately preceding word, giving us

$$p(s) = \prod_{i=1}^{l} p(w_i|w_1\cdots w_{l-1}) \approx \prod_{i=1}^{l} p(w_i|w_{i-1}) \tag{8.1}$$

To estimate $p(w_i|w_{i-1})$, the frequency with which the word w_i occurs given that the last word is w_{i-1}, we can simply count how often the bigram $w_{i-1}w_i$ occurs in some text and normalize. Let $c(w_{i-1}w_i)$ denote the number of times the bigram $w_{i-1}w_i$ occurs in the given text. Then, we can take

$$p(w_i|w_{i-1}) = \frac{c(w_{i-1}w_i)}{\sum_{w_i} c(w_{i-1}w_i)} \tag{8.2}$$

The text available for building a model is called *training data*. The estimate for $p(w_i|w_{i-1})$ is called the *maximum likelihood* (ML) estimate of $p(w_i|w_{i-1})$ because this assignment of probilities yields the bigram model that assigns the highest probability to the training data of all possible bigram models. For n-gram models, where $n > 2$, instead of conditioning the probability of

a word on just the preceding word, we condition this probability on the last $n-1$ words. Generalizing (8.1) to $n > 2$, we get

$$p(s) = \prod_{i=1}^{l+1} p(w_i|w_{i-n+1}^{i-1}) \tag{8.3}$$

where the equation to estimate the probability of $p(w_i|w_{i-n+1}^{i})$

$$p(w_i|w_{i-n+1}^{i}) = \frac{c(w_{i-n+1}^{i})}{\sum_{w_i} c(w_{i-n+1}^{i})} \tag{8.4}$$

8.2 Performance Evaluation

The most common metric for evaluating a language model is the probability that the model assigns to test data, or the derivative measures of *cross-entropy* and *perplexity*. Since we can calculate the probability of a sentense $p(s)$ using equation (8.3), for a test set T composed of the sentences $(t_1, ..., t_{l_T})$ we can calculate the probability of the test set $p(T)$ as the product of the probabilities of all sentences in the set :

$$p(T) = \prod_{i=1}^{l_T} p(t_i)$$

The measure of cross-entropy can be motivated using the well-known relation between prediction and compression. We can derive a compression algorithm that encodes the test T using $-\log_2 p(T)$bits. The cross-entropy $H_p(T)$ of a model $p(w_i|w_{i-n+1}^{i-1})$ on data T is defined as

$$H_p(T) = -\frac{1}{W_T} \log_2 p(T) \tag{8.5}$$

where W_T is the length of the text T measured in words. This value can be interpreted as the average number of bits needed to encode each of the W_T in the test data using the compression algorithm.

The perplexity of $PP_p(T)$ of a model p is the average probability assigned by the model to each word in the test set T

$$PP_p(T) = P(T)^{\frac{1}{W_T}} = \sqrt[W_T]{\frac{1}{W_T}}$$

and also related to cross entropy by

$$PP_p(T) = 2^{H_p(T)}$$

Clearly, lower cross entropy values and perplexity values are better. Lower values means that the model is *better*. Typical perplexity yielded by n-gram models on English text range from about 50 to almost 1000 (corresponding to cross-entropies from about 6 to 10 bits/word).

8.3 Smoothing

Now, consider on unseen data in bigram model which is estimated by equation (8.2)

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

giving us $c(w_{i-1}w_i)$ or worse $\sum_{w_i} c(w_{i-1}w_i) = 0$. Obviously, this is an underestimate for the probability of unseen data as there is some probability that these data occurs. *Smoothing* deals with events that have been observed *zero* times by adjusting low probabilities such as zero probabilities upward and high probabilities downward to make distribution more uniform.

8.3.1 Add-one Smoothing

One simple smoothing technicque is to pretend each bigram occurs once more than it actually does, yielding

$$p(w_i|w_{i-1}) = \frac{1 + c(w_{i-1}w_i)}{\sum_{w_i} 1 + c(w_{i-1}w_i)} = \frac{1 + c(w_{i-1}w_i)}{|V| + \sum_{w_i} c(w_{i-1}w_i)} \tag{8.6}$$

where V is the vocabulary, the set of all words being considered.

8.3.2 Additive Smoothing

Additive smoothing is a generalization of the Add-one smothing method. Instead of pretending each n-gram occurs once more than it does, we pretend it occurs δ times more than it does, where typically $0 < \delta < 1$,

$$p(w_i|w_{i-1}) == \frac{\delta + c(w_{i-1}w_i)}{\delta|V| + \sum_{w_i} c(w_{i-1}w_i)} \tag{8.7}$$

This method generally performs poorly, but better than add-one smoothing.

8.3.3 Good- Turing Smoothing

The *Good-Turing* smoothing is central to many smoothing techniques. The Good-Turing estimate states that for any n-gram that occurs r times, we pretend that it occurs r^* times where

$$r^* = (r+1)\frac{n_r + 1}{n_r} \tag{8.8}$$

where n_r is the number of n-grams that occur exactly t times in the training data. To convert this count to a probability, we just normalize: for an n-gram α with r counts, we take

$$p_{GT}(\alpha) = \frac{r^*}{N} \tag{8.9}$$

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

$$N = \sum_{r=0}^{\infty} n_r r^* = \sum_{r=0}^{\infty} (r+1) n_{r+1} = \sum_{r=0}^{\infty} r n_r$$

The Good-Turning estimate cannot be used when $n_r = 0$, it is generally necessary to "smooth" the n_r before the smoothing estimation. We can use linear interpolation to adjust the n_r so that they are all above zero.

In practise, the Good-Turing estimate is not used by itself for n-gram smoothing, because it does not include the combination of higher-order models with lower-order models necessary for good performance.

8.3.4 Jelinek-Mercer Smoothing

Consider the case of constructing a bigram model on training data where c(STRING THE)= 0 and c(STRING FONZ) = 0, both additive smothing and the Good-Turning estimate will give us

$$p(THE|STRING) = p(FONZ|STRING)$$

However, intuitively we should have

$$p(THE|STRING) = p(FONZ|STRING)$$

because the word THE is much more common than the word FONZ. To capture this behavior, we can *interpolate* the bigram model with a *unigram*

model. We can linearly interpolate a bigram model and unigram model as follows:

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

where $0 \leq \lambda \leq 1$. Because $p(THE|STRING) = p(FONZ|STRING) = 0$, while presumable $p_{ML}(THE) > p_{ML}(FONZ)$, we will have that

$$p_{interp}(THE|STRING) > p_(FONZ|STRING)$$

Jelinet and Mercer gives an elegant way to describe this interpolation is given by, the nth-order smoothed model is defined recursively as a linear interpolation between the nth-order maximum likeliood model and the $(n-1)$th-order smoothed model.

$$p_{interp}(w_i|w_{i-n+1}^{i-1}) = \lambda_{w_{i-n+1}^{i-1}} p_{ML}(w_{i-n+1}^{i-1})) + (1-\lambda_{w_{i-n+1}^{i-1}})p_{interp}(w_i|w_{i-n+2}^{i-1}) \tag{8.10}$$

Different methods for finding the values of λ correspond to variety of different smoothing methods. To yield meaningful results, the data used to estimate λ need to be different from the data used to calculate the p_{MP}. In the *held-out interpolation*, one reserverers a section of data of the training data for this purpose, where this *held-out*data is not used in calculating the p_{ML}. Alternatively a different technique known as *deleted interpolation* or *deleted estimation* where different parts of the training data rotate in training either the p_{ML} or the λ; the results are then averaged.

8.3.5 Katz Backoff Smoothing

Katz smoothing extends the intuitions of the Good-Turing estimate by including the Backoff Smoothing. We describe Katz smoothing for bigram models with count $r = c(w^i i-1)$. We calculate its corrected count using the equation

$$c_{katz}(w^i i-1) = \begin{cases} d_r r & \text{if } r > 0 \text{ and } r < k \text{ for } d_r = \frac{r*}{r} \\ r & \text{if } r \geq k \\ \alpha(w_{i-1})p_{ML}(w_i) & \text{if } r = 0 \end{cases} \tag{8.11}$$

All bigrams with a nonzero count r are discounted according to a *discount ratio* d_r predicted by the Good-Turning estimate. The counts subtracted from the nonzero counts are then distributed among the zero-count bigrams

according to the next lower-order distribution. The value of $\alpha(w_{i-1})$ is chosen so that the total number of counts in the distribution of $\sum_{w_i} c_{katz}(w^i i-1)$ is unchanged. The appropriate value of $\alpha(w_{i-1})$ is

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

To calculate $p_{katz}(w_i|w_{i-1})$ from the corrected count, we just normalize:

$$p_{katz}(w_i|w_{i-1}) = \frac{c_{katz}(w_{i-1}^i)}{\sum_{w_i} c_{katz}(w_{i-1}^i)} \tag{8.12}$$

This calculation is reliable for large counts, therefore they are not discounted. In particular, Katz omits the discount for all $c(w_{i-1}w_i) \geq k$,where Katz suggests $k = 5$. The discount ratios for the lower counts $c(w_{i-1}w_i) < k$ are chosen to be proportional to the discounts predicted by the Good-Turning estimate, so that the total number of counts discounted in the global bigram distribution is equal to the total number of counts that should be assigned to bigrams with zero counts.

8.3.6 Witten-Bell Smoothing

The nth-order smoothed model is defined recursively as a linear interpolation between the nth-order maximum likelihood model and the $(n-1)$th-order smoothed model as in

$$p_{WB}(w_i|w_{i-n+1}^{i-1}) = \lambda_{w_{i-n+1}^{i-1}} p_{ML}(w_i|w_{i-n+1}^{i-1}) + (1 - \lambda_{w_{i-n+1}^{i-1}}) p_{WB}(w_i|w_{i-n+2}^{i-1}) \tag{8.13}$$

To compute $\lambda_{w_{i-n+1}^{i-1}}$ for *Witten-Bell* smoothing, we will need to use the number of unique words that follow the histroy of $w_i|w_{i-n+1}^{i-1}$. We will write this value as $N_{1+}(w_{i-n+1}^{i-1}\bullet)$, formally defined as

$$N_{1+}(w_{i-n+1}^{i-1}\bullet) = |w_i : c(w_{i-n+1}^{i-1}w_i) > 0| \tag{8.14}$$

The notation N_{1+} is meant to evoke the number of words that have one or more counts, and the $\bullet$ is meant to evoke a free variable that summed over. This definition is often called as *diversity*. We can then assign the parameters $\lambda_{w_{i-n+1}^{i-1}}$ for Witten-Bell smoothing such that

$$1 - \lambda_{w_{i-n+1}^{i-1}} = \frac{N_{1+}(w_{i-n+1}^{i-1}\bullet)}{N_{1+}(w_{i-n+1}^{i-1}\bullet) + \sum_{w_i} c(w_{i-n+1}^i)} \tag{8.15}$$

It is reasonable that we should use the higher-order model is the corresponding n-gram occurs in the training data (the value of $\lambda_{w_{i-n+1}^{i-1}}$ is higher than $(1-\lambda_{w_{i-n+1}^{i-1}})$), and back off to the lower order model otherwise. Consider the case of constructing a bigram model on training data of $p_{WB}(BROWSER|THE)$ and $p_{WB}(BROWSER|WEB)$.

$$(1-\lambda_{w_{i-n+1}^{i-1}}) \text{ of } p_{WB}(BROWSER|THE) = \frac{N_{1+}(THE\bullet)}{N_{1+}(THE\bullet) + \sum_{w_i} c(THE, w)}$$

$$(1-\lambda_{w_{i-n+1}^{i-1}}) \text{ of } p_{WB}(BROWSER|WEB) = \frac{N_{1+}(WEB\bullet)}{N_{1+}(WED\bullet) + \sum_{w_i} c(WEB, w)}$$

The number of unique words following THE is higher than the number of unique words following WEB. Therefore I am more certain to know the word that follows WEB than the word follows THE.

8.3.7 Absolute Discounting

Absolute discounting involves the interpolation of higher and lower order models. However, instead of multiplying the higher-order maximum-likelihood distribution by a factor$\lambda_{w_{i-n+1}^{i-1}}$), the higher-order distribution is created by subtracting a fixed discount $D \leq 1$ from each nonzero count. Instead of equation (8.10)

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

we have

$$p_{abs}(w_i|w_{i-n+1}^{i-1}) = \frac{\max\left\{c(w_{i-n+1}^{i}) - D, 0\right\}}{\sum_{w_i} c(w_{i-n+1}^{i})} + (1 - \lambda_{w_{i-n+1}^{i-1}}) p_{abs}(w_i|w_{i-n+2}^{i-1}) \tag{8.16}$$

To make this distribution sum to 1, we take

$$1 - \lambda_{w_{i-n+1}^{i-1}} = \frac{D}{\sum_{w_i} c(w_{i-n+1}^{i})} N_{1+}(w_{i-n+1}^{i-1}\bullet) \tag{8.17}$$

where $N_{1+}(w_{i-n+1}^{i-1}\bullet)$ is defined as in equation (8.14) and where we assume $0 \leq D \leq 1$. Ney, Essen and Kneser(1994) suggest setting D through deleted estimation on the training data. They arrive at the estimate

$$D = \frac{n_1}{n_1 + 2n_2} \tag{8.18}$$

8.3.8 Kneser-Ney Smoothing

Consider building a bigram model on data where there exists a word that is very common, say FRANCISCO, that occurs only after a single word, say SAN. Since $c(FRANCISCO)$is high, the unigram probability $p(FRANCISCO)$ will be high and algorithm such as absolute discounting will assign a relatively high probability to the word FRANCISO occuring after novel bigram histroy.

$$p(FRANCISCO|EGGPLANT) > p(STEW|EGGPLANT)$$

However, intuitively this probability should not be high since the training data the word FRANCISCO follows only a single history. However, STEW is common word and also occurs in many contexts. Extending this line of reasoning, the unigram probability used should not be proportional to the number of occurrences of a word, but instead to the number of different words that it follows.

$$\sum_{w_{i-1}} p_{KN}(w_{i-1}w_i) = \frac{c(w_i)}{\sum_{w_i} c(w_i)} \tag{8.19}$$

The left-hand side of this equation is the unigram marginal for w_i of the smoothed bigram distribution p_{KN}, and the right-hand side if the unigram frequency of w_i found in the trainning data. The resulting distribution presented by Kneser and Ney(1995) is:

$$p_{KN}(w_i|w_{i-n+1}^{i-1}) = \frac{\max\left\{c(w_{i-n+1}^{i}) - D, 0\right\}}{\sum_{w_i} c(w_{i-n+1}^{i})} + \frac{D}{\sum_{w_i} c(w_{i-n+1}^{i})} N_{1+}(w_{i-n+1}^{i-1}\bullet) p_{KN}(w_i|w_{i-n+2}^{i-1}) \tag{8.20}$$

We aim to find a unigram distribution $p_{KN}(w_i)$, such that the constraints given by equation (8.19) are satisfied. Expanding equation (8.19), we get

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

For $p(w_{i-1})$, we simply take the distribution found in training data

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

Substituing and simplying, we have

$$c(w_i) = \sum_{w_{i-1}} c(w_{i-1}) p_{KN}(w_i|w_{i-1})$$

Substituing into equation (8.20), we have

$$c(w_i) = c(w_i) - N_{1+}(\bullet w_i)D + Dp_{KN}(w_i)N_{1+}(\bullet\bullet)$$

where $N_{1+}(\bullet w_i)$ is the number of different words w_{i-1} that precede w_i in the training data and where $N_{1+}(\bullet\bullet)$ is the number of all distinct bigram. Solving for $p_{KN}(w_i)$, we get

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

Applying this to our example,

$$p_{KN}(FRANSICO|EGGPLANT) = \frac{1}{\text{number of distinct bigrams}}$$

$$p_{KN}(STEW|EGGPLANT) = \frac{\text{some number larger than 1}}{\text{number of distinct bigrams}}$$

This boosts up the $P(STEW|EGGPLANT)$, so that it is greater than $P(FRANSICO|EGGPLANT)$.

8.4 Algorithm Summary

Most existing smoothing algorithms can be described with the following equation

$$p_{smooth}(w_i|w_{i-n+1}^{i-1}) = \begin{cases} \alpha(w_i|w_{i-n+1}^{i-1}) & \text{if } c(w_{i-n+1}^{i}) > 0) \\ \gamma(w_{i-n+1}^{i-1})p_{smooth}(w_i|w_{i-n+2}^{i-1} & \text{if } c(w_{i-n+1}^{i}) = 0) \end{cases} \tag{8.21}$$

That is, if an n-gram has a nonzero count then we use the distribution $\alpha(w_i|w_{i-n+1}^{i-1})$. Otherwise, we *backoff* to the lower-order distribution $p_{smooth}(w_i|w_{i-n+2}^{i-1}$, where the scaling factor $\gamma(w_{i-n+1}^{i-1})$ is chosen to make the conditional distribution sum to one. Algorithms that fall directly in this framework as *backoff* models. Serveral smoothing algorithms are experessed as the linear interpolation of higher- and lower- order n-gram models as in equation (8.10)

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

We refer to models of this form as *interpolated* models. The key difference between backoff and interpolated models is that in determining the probability of n-grams with *nonzero* counts, interpolated models use information

from lower-order distributions while backoff models do not. In both backoff and interpolated models, lower-order distributions are used in determining the probability of n-grams with *zero* counts.

algorithm	$\alpha(w_i \mid w_{i-n+1}^{i-1})$	$\gamma(w_{i-n+1}^{i-1})$	$p_{\text{smooth}}(w_i \mid w_{i-n+2}^{i-1})$
additive	$\frac{c(w_{i-n+1}^{i})+\delta}{\sum_{w_i} c(w_{i-n+1}^{i})+\delta\lvert V\rvert}$	0	n.a.
Jelinek-Mercer	$\lambda_{w_{i-n+1}^{i-1}}\, p_{\text{ML}}(w_i \mid w_{i-n+1}^{i-1}) + \ldots$	$(1-\lambda_{w_{i-n+1}^{i-1}})$	$p_{\text{interp}}(w_i \mid w_{i-n+2}^{i-1})$
Katz	$\frac{d_r r}{\sum_{w_i} c(w_{i-n+1}^{i})}$	$\frac{1-\sum_{w_i : c(w_{i-n+1}^{i})>0} p_{\text{katz}}(w_i \mid w_{i-n+1}^{i-1})}{\sum_{w_i : c(w_{i-n+1}^{i})=0} p_{\text{katz}}(w_i \mid w_{i-n+2}^{i-1})}$	$p_{\text{katz}}(w_i \mid w_{i-n+2}^{i-1})$
Witten-Bell	$(1-\gamma(w_{i-n+1}^{i-1}))p_{\text{ML}}(w_i \mid w_{i-n+1}^{i-1}) + \ldots$	$\frac{N_{1+}(w_{i-n+1}^{i-1}\bullet)}{N_{1+}(w_{i-n+1}^{i-1}\bullet)+\sum_{w_i} c(w_{i-n+1}^{i})}$	$p_{\text{WB}}(w_i \mid w_{i-n+2}^{i-1})$
absolute disc.	$\frac{\max\{c(w_{i-n+1}^{i})-D,0\}}{\sum_{w_i} c(w_{i-n+1}^{i})} + \ldots$	$\frac{D}{\sum_{w_i} c(w_{i-n+1}^{i})} N_{1+}(w_{i-n+1}^{i-1}\bullet)$	$p_{\text{abs}}(w_i \mid w_{i-n+2}^{i-1})$
Kneser-Ney (interpolated)	$\frac{\max\{c(w_{i-n+1}^{i})-D,0\}}{\sum_{w_i} c(w_{i-n+1}^{i})} + \ldots$	$\frac{D}{\sum_{w_i} c(w_{i-n+1}^{i})} N_{1+}(w_{i-n+1}^{i-1}\bullet)$	$\frac{N_{1+}(\bullet w_{i-n+2}^{i})}{N_{1+}(\bullet w_{i-n+2}^{i-1}\bullet)}$

This table shows the summary of smoothing algorithms using notation from equation (8.21); the token "..." represents the term $\gamma(w_{i-n+1}^{i-1})p_{smooth}(w_i|w_{i-n+2}^{i-1}$ corresponding to interpolation with a lower order distribution.

Bibliography

[1] Stanley Chen and Joshua Goodman, *An Empirical Study of Smoothing Techniques for Language Modeling*. Technical Report TR-10-98, Harvard University, Aug 1998.

[2] Kevin Knight, Sections 1-14 from *Statistical machine translation workbook*. Manuscript.