Modified Kneser-Ney Language Modeling

Brian Romanowski
btr@msu.edu

February 26, 2011

1 Introduction

Statistical language models are a convenient way to capture linguistic information for use in automated systems. They assign a probability to a sequence of words such that common/expected word sequences have higher associated probability than uncommon/unexpected sequences. Estimating[1] these probabilities from training data allows for good performance in new domains with minimum human intervention. Language models are often combined with other sources of information to improve performance in applications like automated speech recognition, machine translation, and spell checking.

This report describes one particular language modeling technique in detail: the simpler version of Chen and Goodman's modified Kneser-Ney n-gram language model [1]. Supporting sections describe the mathematical notation, language modeling background, and the evaluation of language models.

2 Notation

Let w_t indicate the word at index t in some sequence of n words w_1^n chosen from a vocabulary V. The subsequence starting with the word w_i and ending with the word w_j, inclusive, is denoted by w_i^j. Alternatively, a subsequence of n words that ends at word w_j is denoted by $w_{j-(n-1)}^j$. This notation is illustrated in figure 1.

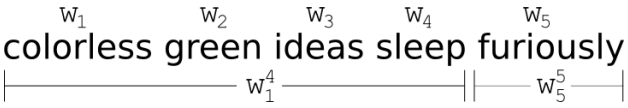

Figure 1: Indexing notation

The probability of a sequence of n words $v', v'', \ldots, v'^{\cdots\prime}$ is denoted by:

$$P\left(\boldsymbol{w}_{\boldsymbol{j-(n-1)}}^{\boldsymbol{j}}{=}v', v'', \ldots, v'^{\cdots\prime}\right) = P\left(\boldsymbol{w_{j-(n-1)}}{=}v', \boldsymbol{w_{j-(n-2)}}{=}v'', \ldots, \boldsymbol{w_j}{=}v'^{\cdots\prime}\right) \tag{1}$$

In this expression, each $\boldsymbol{w_t}$ is a categorically distributed random variable with possible outcomes being the finite set of vocabulary words $v \in V$.

When the actual word sequence is clear from context or when we make statements about the probability of a generalized word sequence, this simpler notation is used:

$$P\left(w_{j-(n-1)}^{j}\right) = P\left(w_{j-(n-1)}, w_{j-(n-2)}, \ldots, w_j\right) \tag{2}$$

[1] I would say statistical language models $\subset$ probabalistic language models; they both report probabilities, but statistical language model parameters are necessarily based on statistical estimates from some training data

The conditional probability of the word at position w_j given the previous $n-1$ word history is used extensively in language modeling:

$$P\left(w_j \mid w_{j-(n-1)}^{j-1}\right) = \frac{P\left(w_{j-(n-1)}^{j}\right)}{P\left(w_{j-(n-1)}^{j-1}\right)} \tag{3}$$

When the conditional probability depends upon words at index $j < 1$, such as at the start of a document, generic "words" that serve to mark the start of the document are assumed/faked/inserted.

Counts of word sequences are often used to produce probability estimates like those above. The counting function $\mathfrak{C}(\cdot)$ counts the number of times word sequences occur, such that:

$$\mathfrak{C}\left(w_i^j\right) = d \tag{4}$$

is defined to mean that the word sequence w_i^j occurred d times in some sample of text.

The Kneser-Ney family of language model smoothing techniques make use of an additional quantity:

$$N_i(w_{j-(n-1)}^{j-1}\bullet) = \text{\# unique words that follow } w_{j-(n-1)}^{j-1} \text{ exactly } i \text{ times} \tag{5}$$

The notation N_{i+} indicates the number of unique words that follow the word sequence at least i times.

3 N-gram Language Models

A straightforward way to estimate the probability of a word sequence is to use the maximum likelihood estimator[2]:

$$\widehat{P}\left(w_i^j\right) = \frac{\mathfrak{C}\left(w_i^j\right)}{N} \tag{6}$$

If the word sequence w_i^j is relatively long (say six words) compared to the amount of training data that is available (say one billion words of web data), we will not find many occurrences of w_i^j in the training data. For sequences that occur infrequently, the maximum likelihood estimate $\widehat{P}\left(w_i^j\right)$ is not reliable, and will often be a poor estimate of the true probability $P\left(w_i^j\right)$.

One step towards a solution is to make the word sequences short, so that they are more likely to occur often in training data. To accomplish this process, we first decompose the probability of a sequence of words according to the chain rule of probability:

$$\begin{aligned} P\left(w_i^j\right) &= P\left(w_i^{j-1}\right) \cdot P\left(w_j \mid w_i^{j-1}\right) \\ &= P\left(w_i\right) \cdot P\left(w_{i+1} \mid w_i\right) \cdots P\left(w_{j-1} \mid w_i^{j-2}\right) \cdot P\left(w_j \mid w_i^{j-1}\right) \end{aligned} \tag{7}$$

If we assume that the each word w_j is only dependent on the previous $n-1$ words, equation 7 simplifies to the following:

$$P\left(w_i^j\right) \approx P\left(w_j \mid w_{j-(n-1)}^{j-1}\right) \cdot P\left(w_{j-1} \mid w_{j-(n-1)}^{j-2}\right) \cdots P\left(w_{i+1} \mid w_i\right) \cdot P\left(w_i\right) \tag{8}$$

[2]This is the maximum likelihood estimator for a multinomial distribution over conditionally independent categorical events. It is analogous to the maximum likelihood estimator for a binomial distribution over a sequence of Bernoulli random variables.

This is the n-*gram assumption*.

In an n-gram model, the probability of the n^{th} word depends only upon the $n-1$ previous words. Each of these conditional probability distributions (i.e., model parameters) may be estimated by the relative frequency calculation:

$$\widehat{P}\left(w_j \;\middle|\; w_{j-(n-1)}^{j-1}\right) = \frac{\widehat{P}\left(w_{j-(n-1)}^{j}\right)}{\widehat{P}\left(w_{j-(n-1)}^{j-1}\right)} \tag{9}$$

$$= \frac{\mathfrak{C}\left(w_{j-(n-1)}^{j}\right)/N}{\mathfrak{C}\left(w_{j-(n-1)}^{j-1}\right)/N} \tag{10}$$

$$= \frac{\mathfrak{C}\left(w_{j-(n-1)}^{j}\right)}{\mathfrak{C}\left(w_{j-(n-1)}^{j-1}\right)} \tag{11}$$

N-gram models are often referred to with Latin prefixes for low order n-grams and with numeral prefixes for high order n-grams as: unigram, bigram, trigram, 4-gram, 5-gram, etc.

4 The Problem of Sparsity

The n-gram assumption helps mitigate sparsity problems, but there are still many word sequences that are seen rarely or not at all in training data. The worst case occurs when there is a sequence of words encountered during testing that was not encountered during training.

As an example, consider a bigram model is trained on the Brown corpus. Suppose the model is used to calculate the probability of the sentence: "the quick brown fox".

$$P\left(w_1^4\right) = P(\text{“the”})P(\text{“quick”}|\text{“the”})P(\text{“brown”}|\text{“quick”})P(\text{“fox”}|\text{“brown”}) \tag{12}$$

It is reasonable that all of these bigrams will appear in a large training corpus and will have non-zero probabilities. However, it may be the case that some bigrams in the similar sentence "the quick *wily* fox" will *not* appear in training data.[3]

If the bigram "quick wily" was not seen in the training data, its relative frequency estimate is zero.

$$P\left(w_1^4\right) = P(\text{“the”})P(\text{“quick”}|\text{“the”})P(\text{“wily”}|\text{“quick”})P(\text{“fox”}|\text{“wily”}) \tag{13}$$

$$= P(\text{“the”})P(\text{“quick”}|\text{“the”}) \cdot \mathbf{0} \cdot P(\text{“fox”}|\text{“wily”}) \tag{14}$$

$$= \mathbf{0} \tag{15}$$

The presence of just one unseen bigram can prevent consideration of a syntactically and semantically valid sentence.

5 Modified Kneser-Ney Smoothing

Kneser-Ney smoothing [3] was modified slightly by Chen and Goodman [1]. Below is a description of the simpler version of their algorithm; the more complicated version that optimizes parameter values on held-out data is not discussed.

[3] All bigrams in 12 appear at least 100,000 times in reported Google search results, whereas the bigram "quick wily" appears in only 500 reported Google search results.

The modified Kneser-Ney smoothing relation is defined recursively as follows:

$$\widehat{P}\left(w_j \;\middle|\; w_{j-(n-1)}^{j-1}\right) = \frac{\mathfrak{C}\left(w_{j-(n-1)}^{j}\right) - D_n\left(\mathfrak{C}\left(w_{j-(n-1)}^{j}\right)\right)}{\sum_{w_j} \mathfrak{C}\left(w_{j-(n-1)}^{j-1}, w_j\right)} + \gamma\left(w_{j-(n-1)}^{j-1}\right) \widehat{P}\left(w_j \;\middle|\; w_{j-(n-2)}^{j-1}\right) \tag{16}$$

where

$$D_n(c) = \begin{cases} 0 & \text{if } c = 0 \\ 1 - 2\left(\frac{n_1}{n_1+2n_2}\right)\left(\frac{n_2}{n_1}\right) & \text{if } c = 1 \\ 2 - 3\left(\frac{n_1}{n_1+2n_2}\right)\left(\frac{n_3}{n_2}\right) & \text{if } c = 2 \\ 3 - 4\left(\frac{n_1}{n_1+2n_2}\right)\left(\frac{n_4}{n_3}\right) & \text{if } c \geq 3 \text{ indicated by "} c = 3+\text{"} \end{cases} \tag{17}$$

where n_i are the number of n-grams of order n that appear i times in the training data, and:

$$\gamma\left(w_{j-(n-1)}^{j-1}\right) = \frac{D_n(1)N_1\left(w_{j-(n-1)}^{j-1}\bullet\right) + D_n(2)N_2\left(w_{j-(n-1)}^{j-1}\bullet\right) + D_n(3+)N_{3+}\left(w_{j-(n-1)}^{j-1}\bullet\right)}{\sum_{w_j} \mathfrak{C}\left(w_{j-(n-1)}^{j-1}, w_j\right)} \tag{18}$$

We assume a closed vocabulary and terminate the recursion such that the unigram probability is just the maximum likelihood estimate:

$$\widehat{P}\left(w_j\right) = \frac{\mathfrak{C}\left(w_j\right)}{N} \tag{19}$$

These equations are all taken directly from the Chen and Goodman paper; however, we make explicit the fact that the parameters $D_n(c)$ are calculated for each order n.

Things to prove or examine (most of these things are proven in Chen and Goodman):

- $\sum_{w_j} \widehat{P}\left(w_j \;\middle|\; w_{j-(n-1)}^{j-1}\right) = 1$
- intution of $\gamma(\cdot)$
- intuition behind KN
- Is this generally how SRILM does it?

6 Evaluation

For a particular task, language model performance is best modeled by using a language model as a component *in situ*, in a full system. In this setup, the language model may effect system accuracy, but may also be analyzed with respect to training time, model size, and performance as the task domain changes in time. However, the necessity of repeated system runs and the difficulty of collecting adequate test data can make *in situ* evaluations costly. Additionally, it is not obvious how performance results conditioned on a particular task informs the *in situ* performance in a different task or domain.

6.1 Perplexity

The *perplexity* metric measures language model performance with respect to textual testing data. Language model perplexity is often strongly correlated with task performance. Perplexity may be defined as shown [2]:

$$\text{PP}(W) = P\left(w_1^N\right)^{-\frac{1}{N}} \tag{20}$$

where $\text{PP}(W)$ is the perplexity of a test data set w_1^N.

This definition of perplexity may be interpreted as the geometric mean of the inverse probability of the test set. The probability of the test data is a product of N individual conditional n-gram probabilities. The geometric mean is the number that, when multiplied N times, produces an equivalent probability. This value characterizes the average uncertainty of a word in the testing data.

The perplexity, as the inverse of the geometric mean, can be viewed as average size of the set of equiprobable words at each index in the test dataset. Thus, *lower perplexity indicates a better fitting model*, because there is a smaller number of word alternatives for each word in the testing data, on average.

Perplexity is also related to the information theoretic quantity *cross entropy*. See appendix 7 for the proof that perplexity is related to cross entropy in the following manner:

$$H(p, q) = \lg \text{PP}(w_1^N) \tag{21}$$

where $H(\cdot, \cdot)$ is cross entropy and $\lg$ is the base 2 logarithm.

Improvements in perplexity do not always correspond to improvements in task performance. Perplexity is evaluated only on valid sentence test data, so no measure of false positive performance is indicated. Additionally, a language model that assigns more probability mass to common words may increase perplexity, but it may also "overwhelm" the probability mass contributed by other sources of information, like the acoustic model in speech recognition decoders.

6.2 Statistical Significance

7 Perplexity and Cross Entropy

An argument will be made for use of the cross entropy metric when evaluating language model data. The cross entropy metric is also shown to be equivalent to the previously defined perplexity metric. The reasoning presented below is taken from Jurafsky and Martin [2]; additional annotation and discussion is also provided.

First, consider the entropy of a language consisting of all possible word sequences, as the length of the word sequences approaches infinity.

$$H(p) = -\lim_{n\to\infty} \frac{1}{n} H_n \tag{22}$$

$$= -\lim_{n\to\infty} \frac{1}{n} \sum_{w_1^n} p(w_1^n) \lg p(w_1^n) \tag{23}$$

$$= -\lim_{n\to\infty} \frac{1}{n} \lg p(w_1^n) \tag{24}$$

where $\lg x$ is the base-2 logarithm of x, $H(p)$ is the total entropy of a language W, H_n is the entropy of a sequence of words of length n, and w_1^n is the set of all sequences of length n. In equations 24, we make use of the Shannon-McMillan-Breiman theorem. The necessary assumption is that the language W is ergodic, meaning that all possible word sequences appear in an infinitely long sample from the language, and that

each word sequence appears with a frequency corresponding to its true probability of appearance. Then, looking at an infinitely long string captures the same information available from any weighted, shorter string.

Next, we examine the cross entropy $H(p, q)$:

$$H(p,q) = -\lim_{n\to\infty} \frac{1}{n} \sum_{W_1^n} p(W_1^n) \lg q(W_1^n) \tag{25}$$

$$= -\lim_{n\to\infty} \frac{1}{n} \lg q(W_1^n) \tag{26}$$

$$\approx -\frac{1}{N} \lg q(W_1^N) \tag{27}$$

where the Shannon-McMillan-Breiman theorem is used again in equation 26. The approximation in equation 27 is derived by assuming that the testing data on which we wish to evaluate a language model is long enough to capture important language properties.

This derivation allows us to state that the entropy of a language is bounded by the cross entropy of our language model:

$$H(p) \leq H(p,q) \tag{28}$$

It should be clear that a better language model will approach "closer", that is have lower cross entropy, to the entropy of the language.

Perplexity is an equally valid metric, as it is related to cross entropy as follows:

$$H(p,q) = \lg PP(W_1^N) \tag{29}$$

8 License

References

[1] S. Chen and J. Goodman. An empirical study of smoothing techniques for language modeling. Technical Report TR-10-98, Harvard, 1998.

[2] D. Jurafsky and J. Martin. *Speech and Language Processing*. 2 edition, 2008.

[3] R. Kneser and H. Ney. Improved backing-off for m-gram language modeling. In *Acoustics, Speech, and Signal Processing, 1995. ICASSP-95., 1995 International Conference on*, volume 1, pages 181–184, 1995.