HW3: Language Modeling

Jeffrey Ling
jling@college.harvard.edu

Rohil Prasad
prasad01@college.harvard.edu

March 9, 2016

1 Introduction

In this assignment, we examine the problem of language modeling. Given a set of words, we want to determine which word comes after it. Note, however, that there may not be a "best" word to follow a prefix. For example, "fish" and "chicken" could both be almost equally likely to follow the prefix "I ate ___". Therefore, it makes more sense to try and find a good **probability distribution** of the next word given the prefix. We will define what it means to be "good" later in the paper.

We implement and discuss several different methods of language modeling on the Penn Treebank in this paper. Our first class of algorithms are probabilistic count-based algorithms, consisting of maximum likelihood estimation, Laplace smoothing, Witten-Bell smoothing, and Kneser-Ney smoothing. Our second class of algorithms are deep-learning algorithms, consisting of the neural network described in [Bengio]. Furthermore, we also experiment with augmenting this neural network with word embeddings learned via Noise Contrastive Estimation (NCE).

In Section 2, we give a formal description of our problem and establish our notation. In Section 3, we give detailed descriptions of the algorithms listed above. In Section 4, we present our experimental results. In Section 5, we discuss our findings.

2 Problem Description

Assume our training corpus consists of a sequence of words $w_1, w_2, \dots, w_N$ where w_i is drawn from a vocabulary $\mathcal{V}$ for every i.

Our training data represents this corpus as a set of tuples $(\mathbf{x}_i, y_i)$, where $\mathbf{x}_i$ is the prefix $w_1 w_2 \dots w_{i-1}$ and y_i is equal to the suffix w_i.

Our goal is to train a model that will take in an input sequence $\mathbf{x}$ of words and output a vector $\mathbf{y} \in \mathbb{R}^{1\times|\mathcal{V}|}$ where $\mathbf{y}_i = p(v_i|x)$ for v_i the ith element of $|\mathcal{V}|$.

2.1 Count-Based Model Notation

We use some specific notation for our count-based models.

- w_{i-n+1}^{i-1} is our shorthand for the size $n-1$ prefix $w_{i-n+1} w_{i-n+2} \dots w_{i-1}$ of w_i. When $n = 2$, we will continue to use w_{i-1}.

- $c(w_i)$ denotes the total number of occurrences of the word w_i in the training corpus.
- $c(w_{i-n+1}^{i-1}, w_i)$ denotes the total number of occurrences of the word w_i after the prefix w_{i-n+1}^{i-1} in the training corpus.
- $N(\cdot, w_i)$ denotes the total number of unique prefixes of size n that w_i has in the training corpus.
- $N(w_{i-n+1}^{i-1}, \cdot)$ denotes the total number of unique suffixes that the sequence w_{i-n+1}^{i-1} has in the training corpus.

2.2 Evaluation

Unlike earlier assignments, we are outputting a *distribution* of possibilities given an input $\mathbf{x}$ rather than a *prediction* of the word following $\mathbf{x}$.

Therefore, we will evaluate by a metric called *perplexity*.

3 Model and Algorithms

3.1 Count-Based Models

All of our count-based models are n-gram models, meaning we fix an n and assume $p(w_i|w_1^{i-1}) = p(w_i|w_{i-n+1}^{i-1})$.

3.1.1 Maximum Likelihood Estimation

The maximum likelihood estimation $p_{MLE}(w_i|w_{i-n+1}^{i-1})$ is given by

$$p_{MLE}(w_i|w_{i-n+1}^{i-1}) = \frac{c(w_{i-n+1}^{i-1}, w_i)}{\sum_{w' \in \mathcal{V}} c(w_{i-n+1}^{i-1}, w')}$$

This is by definition the probability over the training corpus of the word w_i appearing conditioned on the prefix being equal to w_{i-n+1}^{i-1}.

3.1.2 Laplace Smoothing

One thing to note is that our MLE model will by definition assign a (clearly erroneous) probability of 0 to $p(w|w_{i-n+1}^{i-1})$ for any w_{i-n+1}^{i-1} that does not appear in the training corpus. This poses a problem for larger n, since the number of distinct n-grams in the training corpus will only form a small fraction of the number $|\mathcal{V}|^n$ of possible n-grams.

The traditional maximum likelihood n-gram model by definition will often erroneously assign a probability of 0 to $p(w|w_{i-n+1}^{i-1})$.

We fix this by adding α to the count of every word. Then, our probability becomes

$$p_L(w_i|w_{i-n+1}^{i-1}, \alpha) = \frac{\alpha + c(w_{i-n+1}^{i-1}, w_i)}{\alpha \cdot |\mathcal{V}| + \sum_{w' \in \mathcal{V}} c(w_{i-n+1}^{i-1}, w')}$$

The optimal value for α is determined by grid search on a validation set.

3.1.3 Witten-Bell Smoothing

The Witten-Bell probability for bigrams with an additive smoothing parameter α is written as

$$p_{WB}(w_i|w_{i-1}) = \lambda_{w_{i-1}} p_L(w_i|w_{i-1}, \alpha) + (1 - \lambda_{w_{i-1}}) p_L(w_i|\alpha)$$

where $\lambda_{w_{i-1}}$ is determined by the following equation:

$$1 - \lambda_{w_{i-1}} = \frac{N(w_{i-1}, \cdot)}{N(w_{i-1}, \cdot) + c(w_{i-1}, w_i)}$$

We can then define Witten-Bell recursively for n-grams as

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

where $\lambda_{w_{i-n+1}^{i-1}}$ is given by:

$$1 - \lambda_{w_{i-n+1}^{i-1}} = \frac{N(w_{i-n+1}^{i-1}, \cdot)}{N(w_{i-n+1}^{i-1}, \cdot) + c(w_i, w_{i-n+1}^{i-1})}$$

The optimal value for α is determined by grid search on a validation set.

3.1.4 Kneser-Ney Smoothing

For Kneser-Ney, we define the lower-order probabilities as follows. Fix a "discount parameter" δ. For unigrams, the equation is:

$$p_{KN}(w_i) = N(\cdot, w_i) / \sum_{w' \in \mathcal{V}} N(\cdot, w')$$

Then for bigrams, it is defined as

$$p_{KN}(w_i|w_{i-1}) = \frac{\max(c(w_{i-1}, w_i) - \delta, 0)}{\sum_{w' \in \mathcal{V}} c(w_{i-1}, w')}$$

Finally, we can define it recursively for general n-grams:

$$p_{KN}(w_i|w_{i-n+1}^{i-1}) = \frac{\max(c(w_{i-n+1}^{i-1}, w_i) - \delta, 0)}{\sum_{w' \in \mathcal{V}} c(w_{i-n+1}^{i-1}, w')}$$

The optimal value for δ is determined by grid search on a validation set.

3.2 Neural Network Model

3.3 Noise Contrastive Estimation

4 Experiments

5 Conclusion

References