A1 CSE 490UA

Jason You (ID: 1427878)

January 18, 2017

1 Problem

Solution:

The model provided in the handout sheet is **not** a proper probability distribution. Because

$$\sum_{i=1}^{3} p_i > 1$$

1.1 *Proof:*

- Assume that we train on a corpus with the model provided in the handout.
- Then according to the slide (class notes), we know that $\sum p_{ML}(w_i|w_{i_2}, w_{i-1}) = 1$ will always be true for a training data set.
- Assume there are more than one trigrams in the test corpus not exist in the train corpus, and assume that all the unigram maximum likelihood exist.
- So we know that the p_2 and p_3 defined in this model will be greater than zero, but this will cause the cumulative sum of the whole model to be greater than one, which is a contradict to the definition of a proper probability model.
- Thus we proved our claim.

1.2 *Improvements*

1. Multiply p_2 by a hyper-parameter β_1 and multiply p_3 by β_2, where

$$\beta_1 = 1 - \sum_{w \in A(w_{i-2}, w_{i-1})} p_{ML}(w_i|w_{i-2}, w_{i-1}) \tag{1}$$

$$\beta_2 = 1 - \sum_{w \in A(w_{i-1})} p_{ML}(w_i|w_{i-1}) \tag{2}$$

These two hyper-parameter is determined by developement corpus (a holdout corpus other than train and test).

2. This approach makes sure that the model will discount a portion of probability from the p_1 if the tri-grams not found in the development corpus, and the same machanism for the bi-grams absence case.

2 Experiment

Solution:

2.1 Issues

Language model plays an central role for the probabilistic side of NLP, and one of the most common question people will ask is how likely is a sequence of words, or sentence, appear in a real corpus. In this experiment I'll examing two entry level smoothed language models that people normally used to explain how language models work.

2.2 Problems

Before start to build the model, a few questions need to be considered. I need to define how the word sequences will be treated, and the way to combine their frequencies into a proper probability distribution. I'll also need to deal with the situation that words in the train corpus do no exist in the test corpus. Finally, a acceptable way to test and evaluate the models will be prefered.

2.3 Cleaning Corpus

For this experiment I choose the Python programming language, and the 'nltk' package provides a pre-cleaned corpus dataset, which include the 'brown', 'gutenberg', and 'reuters' for this assignment. I split the corpus by sentences, and added the ⟨**STOP**⟩ sign as a stop symbol at the end of each sentence.

2.4 Corpus Selection

For each of the 'brown', 'gutenberg', and 'reuters', I use 60% as train corpus, 20% as the test, and 20% as the development corpus.

Before finding the frequencies for each of the tri-grams, bi-grams, and uni-gram of each corpus, I convert all the words with frequency of **1** into ⟨**UNK**⟩ symbol, which means unknow word. This approach is for the smoothing method I'll discuss later.

2.5 Back-off Method

After gaining the frequency distribution for each of the tri-grams and bigrams in the train corpus, I can gain a decent predicting result for my train corpus with the Maximum Likelihood (ML) method. But the issue here is that we can guarantee this performance in the test corpus, that is, some words or grams might just don't exist in there.

The idea of Back-off is to discount a small portion of probability from the trigram model, which we can determine this portion by a development corpus that don't show in train nor test corpus, and use this portion as a parameter to multiply the Maximum Likelihood of a lower level gram (e.g., from trigrams to bigrams ML).

The function I used to calculate these two hyper-parameters was given in the Problem one above, except in my Python code I set the upper bound to be less than 0.5, and assign the parameters both to be 0.3 if the value excess 0.5. The value of 0.3 is based on the approximate missing rate of trigrams in the test corpus:

$$Approximated\ Parameters \approx 1 - \frac{c(trigramsexistintestcorpus)}{c(totaltrigramsfromtrain)}$$

where the $c()$ function in here is just counting elements.

2.6 Interpolation Method

The idea of this method is to combine all three grams ML models, and use the hyper-parameters to balance the weight assigned to each one of them. So as long as the hyper-parameters astisfy the following equations, this model will in a proper probability distribution:

$$p_{ML}(w_i|w_{i-2}, w_{i-1}) = \lambda_1 p(w_i|w_{i-2}, w_{i-1}) + \lambda_2 p(w_i|w_{i-1} + \lambda_3 p(w_i)$$

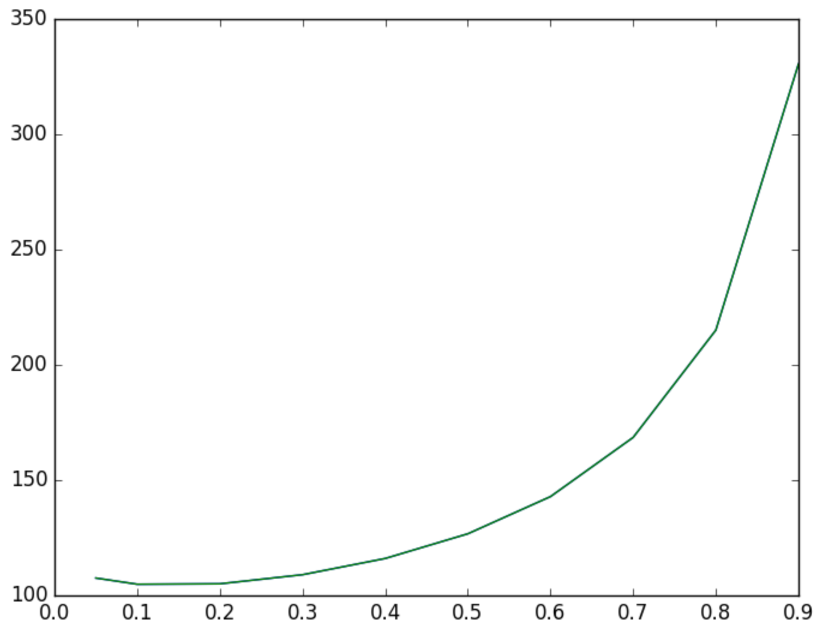
and

$$\sum_{i=1}^{3} \lambda = 1 \tag{3}$$

$$\lambda_1,\ \lambda_2,\ \lambda_3 > 0 \tag{4}$$

In my implementation, if any of the n-gram(s) doesn't exist in the test corpus, then that term will be diminished into zero, which is coherent with the definition of the original model.

For finding these hyper-parameters, I use the held-one out method, which is just using a independent development corpus to evaluating the performance of the model with different gussed λ_i.



Here is an example of evaluating these these hyper-parameters with $\lambda_1 = 0.05 \to 0.9$, and $\lambda_2, \lambda_3 = (1 - \lambda_1)/2$.

1. **brown:**

$$\lambda_1 = 0.15 \tag{5}$$
$$\lambda_2 = 0.425 \tag{6}$$
$$\lambda_3 = 0.425 \tag{7}$$

2. **gutenberg:**

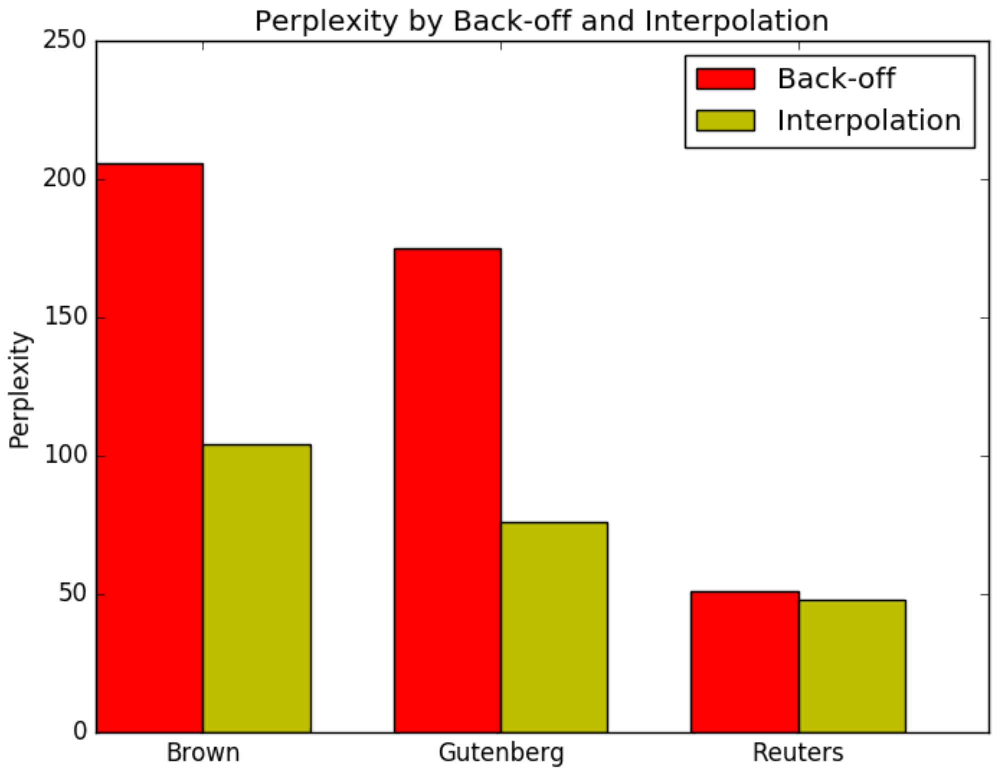
$$\lambda_1 = 0.2 \tag{8}$$
$$\lambda_2 = 0.4 \tag{9}$$
$$\lambda_3 = 0.4 \tag{10}$$

3. **brown:**

$$\lambda_1 = 0.3 \tag{11}$$
$$\lambda_2 = 0.35 \tag{12}$$
$$\lambda_3 = 0.35 \tag{13}$$

3 Perplexity of These Two Methods

Using the perplexity we can easily visualize the performance of the Back-off and Interpolation methods:

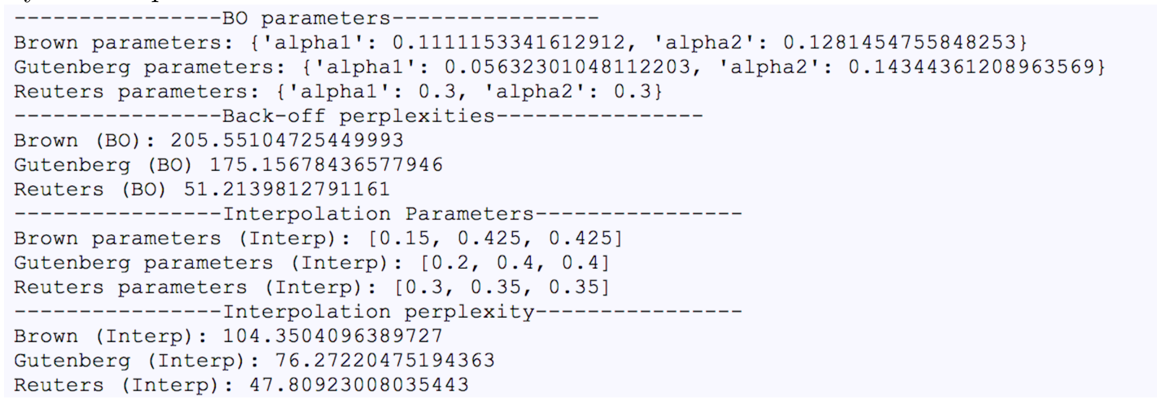


4 Conclusion

Since I didn't use a hold-one out method to evaluate the hyper-parameters of the Back-off model, the performance of it is worse than the Interpolation model by a factor of 2.

However, for the reuters approach these two method both did a good prediction, and I this might caused by the fact that I the reuters corpus is so diverse that most of the grams are covered in the train corpus.

Below are some outputs from my Python build models. For more details, please refer to my Python script.



```
----------------BO parameters----------------
Brown parameters: {'alpha1': 0.1111153341612912, 'alpha2': 0.1281454755848253}
Gutenberg parameters: {'alpha1': 0.05632301048112203, 'alpha2': 0.14344361208963569}
Reuters parameters: {'alpha1': 0.3, 'alpha2': 0.3}
----------------Back-off perplexities----------------
Brown (BO): 205.55104725449993
Gutenberg (BO) 175.15678436577946
Reuters (BO) 51.2139812791161
----------------Interpolation Parameters----------------
Brown parameters (Interp): [0.15, 0.425, 0.425]
Gutenberg parameters (Interp): [0.2, 0.4, 0.4]
Reuters parameters (Interp): [0.3, 0.35, 0.35]
----------------Interpolation perplexity----------------
Brown (Interp): 104.3504096389727
Gutenberg (Interp): 76.27220475194363
Reuters (Interp): 47.80923008035443
```