E1 246 - Natural Language Understanding
Assignment 1 - Language Models

Kunal Chelani
CSA, Indian Institute of Science
kunalchelani@iisc.ac.in
https://github.com/kunalchelani/NLU-Assignment

1 Introduction

As part of this assignment, I have tried ngram modelling with various smoothing techniques to support them. In this report, I first mention the pre-processing steps that I performed on the data, then in section 3, I describe my approach towards handling OOV (Out of Vocabulary) words, and the subsections of section 4, describe the smoothing techniques that I applied. Section 5 states the results across the four settings mentioned as part of the assignment task-1, section 6 gives some qualitative insight about the models' performances. I conclude in section 6 by summarizing the results achieved and stating a few other techniques I could have tried.

2 Pre-Processing

Corpus-specific pre-processing techniques have been applied. Here is a quick description along with reasons why they were applied.

2.1 Gutenberg Corpus

The Gutenberg Corpus is a fairly simpler one to deal with for n-gram modelling. The data was tokenized into sentences using nltk's sentence tokenizer. This was done so that SOS and EOS tags could be added before the start and after the end of sentences respectively. The sentences were then tokenized into words, and all words converted to lowercase, to remove capitalization-based differences amongst same tokens. There were certain tokens having an underscore at the start and end. These underscores were removed as they added no feature and increased the vocab size.

2.2 Brown Corpus

The brown data as part of speech associated with every word. For simplicity I have used the Brown corpus included as part of the NLTK's corpus module. This gives the sentences tokenized. The unicode strings have been converted to strings before performing counts and smoothing.

3 Vocabulary

To handle unseen unigrams in the unseen data, I have used the standard technique of making the vocabulary closed. I choose a min-freq threshold, the words in the training data, that lie below it are tagged as ¡UNK¿ (unknown). Hence the probability is distributed amongst a a fixed set of words. The unseen unigrams in the test data are also mapped to ¡UNK¿ and therefore get non-zero probabilities of being observed.

4 Smoothing

Just like unobserved unigrams were assigned non-zero probabilities by introducing ¡UNK¿, smoothing techniques allow higher order unobserved n-grams to obtain non-zero probabilities. This can be done by discounting certain amount of probability mass from the observed n-grams.

4.1 Laplacian Smoothing

Laplacian Smoothing is the simplest of all smoothing techniques. To make the probability of each n-gram non-zero, we add 1 to their count and to make it a valid probability distribution, we add $|V|^(n-1)$ to the denominator, where n is the level we are smoothing at (for eg. n = 2 for bigrams). Since this method adds a huge value to the denominator, it has a big impact on the probabilities of most observed n-grams and therefore, does not result in good performance. This method has been tried for bigrams since for trigrams, it further dilates all probabilities.

4.2 Add-k Smoothing

Add-k smoothing is just a modified version of the Laplacian smoothing, where instead of adding a 1 to the numerator always, we add k (less than 1) to the numerator and therefore add $(k|V|)^{n-1}$ to the denominator. The formula for a bigram probability (observing w_i given the previous word was w_{i-1}), becomes:

$$P_{Add-k} = \frac{C(w_{i-1}w_i)+k}{C(w_{i-1})+k|V|}$$

4.3 Katz Backoff

The idea of backing-off to lower order n-grams when higher order n-grams haven't been observed, is a novel one. Katz Backoff method proposed an elegant way of discounting the probability mass from the observed n-grams and redistributing that mass in proportion to lower order frequencies. The following equations that govern Katz discounting and redistribution:

$$P_{BO}(w_i|w_{i-2}w_{i-1}) = \alpha \frac{P_{BO}(w_i|w_{i-1})}{\sum_{w\in\beta 1} P_{BO}(w|w_{i-1})} \quad (1)$$

where α depends on $w_{i-2}w_{i-1}$ and $\beta 1$ is the set of words that complete the unseen trigrams. Also, the bigram backoff probability is recursively defined as:

$$P_{BO}(w_i|w_{i-1}) = \alpha \frac{C(w_i)}{\sum_{w\in\beta 2} C(w))} \quad (2)$$

This method performs quite well in my setting.

4.4 Kneser ney Smoothing

Kneser and Ney suggested an even better way to distribute the total discount, which is dependent on the count of the novel contexts in which an word appears. Following is the interpolation form of the Kneser-ney:

I have used Kneser-ney in a bigram setting as the trigram level implementation becomes cumbersome. The equation governing the bigram Kneser-ney smooting is:

$$P_{KN}(w_i|w_{i-1}) = \frac{max(C(w_{i-1}w_i) - d, 0)}{C(w_{i-1})} + \lambda P_{cont}(w_i)$$

Here,

$$P_{cont}(w_i)$$

is proprtional to the number of novel contexts in which w_i appears.

5 Hyperparameter Tuning

The following subsections provide the hyperparameter tuning plots for the respective smoothing methods

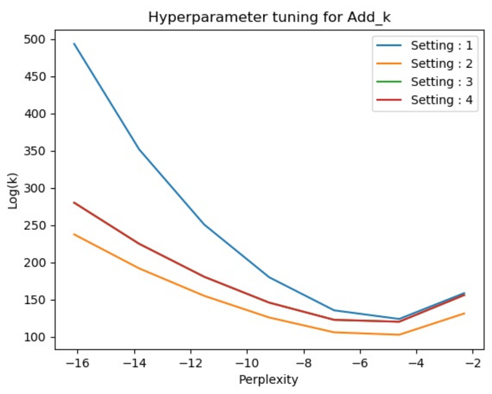

Figure 1: The plot shows that somewhere close to e^-5, the minimum perplexity is achieved. This is used for the final perplexity calculation. third and fourth settings have the same dev-set hence the overlap

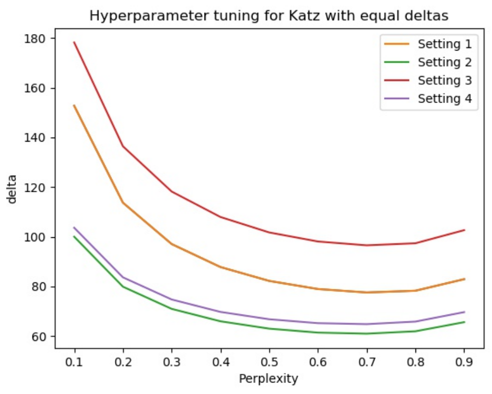

Figure 2: This plot shows the tuning of hyperparameter delta of Katz backoff. I have used the same discount for bigrams and trigrams. Value close to 0.75 gives the best results

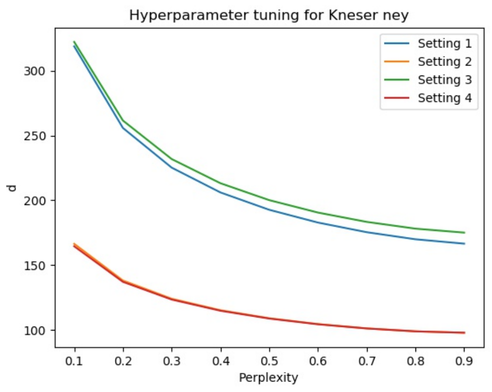

Figure 3: This shows that on increasing d uptil 0.9 the Kneser ney perplexity decreases. This has been used as the optimal value

Setting	Smoothing Method	Perplexity on test data
Setting 1	Laplacian	328.7
Setting 1	Add-k	137.4
Setting 1	Kneser-ney bigram	174.09
Setting 1	Katz Backoff	**77.58**
Setting 2	Laplacian	247.5
Setting 2	Add-k	103.5
Setting 2	Kneser-ney bigram	101.34
Setting 2	Katz Backoff	**61.58**
Setting 3	Laplacian	328.7
Setting 3	Add-k	137.4
Setting 3	Kneser-ney bigram	175.09
Setting 3	Katz Backoff	**102.58**
Setting 4	Laplacian	275.7
Setting 4	Add-k	107.4
Setting 4	Kneser-ney bigram	97.4
Setting 4	Katz Backoff	**64.58**

6 Results

- The table above summarizes the best perplexity values after choosing across the best hyper-parameters.
- The threshold for handling out of vocabulary words has been kept at 7.
- The Katz method shows the best result as we are taking trigrams into account and the discounted mass of probabilities is distributed in much more sophesticsted manner than the laplacian or add-k methods.
- An important observation here is that it can be seen that training on a larger corpus not always, is helpful. The test data should be similar to the training data, which is true in the first two settings. The third and fourth setting show that the probabilities have been shifted to give poorer results.

7 Qualitative Assesment

Qualitative examples are probably the best way to assess language models, since other metrics such as perplexity can be altered by changing vocablary size and by other ways as well. here are a few sentences generated by the best performing model.

- i'm a big black moustache, he is.
- we shall bend before those god in vineyards.
- he shall be your joint; then 21:11 thy grace"
- about being over meadows patience they called

8 Conclusion

I would like to conclude by saying that trigram Kneser would definitely had been a plus to implement and I could have gained, much better results using it. Also, looking at the sentences, we can easily figure out that the ngram language modelling technique isn't quite enough for generating. Powerfull methods that encode word meanings and structures are required for this task. It was fun though, to experiment with this assignment.