NLP Paper Draft

Anonymous ACL submission

1 Introduction

Our team chose to investigate word prediction techniques. Specifically, we wanted to measure the effectiveness of different language models and smoothing techniques on word prediction accuracy. We built bigram and trigram language models to measure how much accuracy would be gained by the trigram language model and what the performance cost would be. We used add-lambda smoothing on both models, with varying lambda values. With these techniques, we aim to determine which lambda value results in the most empirically accurate word prediction results.

2 Experimental Setup

We trained our bigram and trigram models on the first 90% of the sentences in a corpus of simple sentences we found online. We've found the corpus at the site linked here [1]. We trained using 356,549 sentences. We cut out the last 10% of the sentences to use them for evaluation. After training a given model with a given lambda on the corpus, we would ask it to predict the next word in a sequence on the test data. Specifically, for each sentence in the test data, we would cut the sentence off at a random point between the second and final words. Then we would ask the trained model to predict the next word in the sentence. We would evaluate the model based on its accuracy in predicting the correct word in the test data.

We trained 11 different models total. In other words, we ran 11 experiments, each with a different lambda λ

- λ : 0.0, 0.2, 0.4, 0.6, 0.8, 1.0

When evaluating of the success of each of these trials we will calculate the accuracy of our model's predictions against the true test data. We simply use the sheer number of correct word predictions divided by the number of test sentences.

[1]https://zenodo.org/record/205950.ZFHhr-zMIq0

To detail some obstacles we faced in our process, we originally planned to also study discounting to compare to lambda smoothing. However, we encountered many errors trying to generate the probabilities as the process is complex to generate the discounted probabilities for the trigram model. After several hours of debugging we decided to scrap discounting so we focused solely on lambda smoothing.

3 Results

3.1 Tables and figures

Lambda	Bigram Accuracy	Trigram Accuracy
0.0	9.07 %	11.9 %
0.2	8.94 %	11.5 %
0.4	8.94 %	11.5 %
0.6	8.62 %	12.3 %
0.8	9.02 %	12.7 %
1.0	8.42 %	12.2 %

See the Table 3.1 for the prediction accuracy results of our experiment for the bigram model and the trigram model.

We used 4004 test sentences for all the lambda values and tested the accuracy of our model for each.

We observe that a lambda of 0 has the highest bigram accuracy being 9.07 % and a lambda of .8 has the highest trigram accuracy with a 12.7 % accuracy.

We ran a χ^2 good fit test on the trigram data with a 5 % significance level and found that there was no statistical significance in the data related to which lambda value resulted in the highest accuracy. Our χ^2 value was 2.72 but, with 5 degrees of freedom, we would have needed a value of 11.07 or higher to reject the null hypothesis that the amount of correct predicted words is relatively the same across all lambdas.

We ran a χ^2 good fit test on the bigram data with a 5 % significance level. We found that also was no statistical significance in the data related to which lambda value resulted in the highest accuracy. Our χ^2 value was 1.5 but, with 5 degrees of freedom, we would have needed a value of 11.07 or higher to reject the null hypothesis that the amount of correct predicted words is relatively the same across all lambdas.

3.2 Appendices

For the hours spent: Noah Nevens (14 hours+), Adam Cohan, (14.5 hours+), Waverly Wang (14 hours+)

Adam created the overall class structure of our project. He created the idea of making an NGram-Model class so we could then make the Bigram and Trigram Model classes. He also created an NGram Class with its own methods to obtain the NGram counts.

Noah helped create the Trigram Class and Waverly created the Bigram Class and helped create the discounting method. We all worked on the Smoothing class which would execute lambda smoothing and discounting.

All members helped look over each other's code and assist in the long debugging processes.

4 Ethics

The ethical discussion surrounding our project is similar to those surrounding ChatGPT as they are both predictive text-generation models (albeit one is slightly more effective than the other). While some of the considerations to take in are our model suggesting words that we as its creators don't support the usage of (say, slurs), this is only part of the discussion. Some of the dangers of an effective text generation model are people using it to impersonate people, whether it's for school assignments or to automatically spread misinformation online en masse. In terms of the workings of the model, perhaps it's best to have something related to the perplexity or some other metric always fall in a given range so that it's easy to verify whether it was generated by the model or not, regardless of how real it sounds. However, the issue with this and other hidden verification methods is that they may hinder the performance of the model. While in the long run, this may be for the best, in the short run it is counterintuitive, especially when the goal of the project is to push the boundaries and get the predictive text of the model to be as real sounding as possible.