COM6513: Language Modelling

Registration Number: 180128251

March 2, 2019

1 Introduction

This lab involves the use of statistical methods to perform text completion. Different language models is introduced as well as a technique to overcome challenges when working with certain aspects of a particular language model.

2 Implementation

The application is developed with Python 3 and was tested on a computer with a 8GB RAM and an Intel i5 - 8th Gen at 1.60Hz. It accepts two command-line arguments; the first being the corpus for our training data and the second being a list of questions for testing our language models. The program consists of 3 functions namely: **unigram**, which accepts an argument being *value* that is a word with which its probability is to be computed using the unigram language model. The **bigram**, accepts 2 arguments: *value* being the word to be computed with the bigram language model and *smoothing* which is an optional parameter with a default value of 0 that is applied to the computation of the probability with smoothing (we however use an add-1 smoothing so our value of *smoothing* is either 1 or 0). Finally, the **predict** function that contains 4 arguments: *p1*, *p2*, *a1* and *a2*, these are the probabilities and text answers to be predicted respectively. This function attempts to find the right answer given their corresponding prediction values and returns the predicted answer, it's probability and if the probability values of both answers are the same. The program starts by first training our language models(unigram and bigram) using our corpus. To test our models, a list of all the right answers is created for computing the accuracy of our models and each model is used to predict a missing word from our test data.

3 Evaluation

Using the **unigram** language model, an accuracy of 50% was obtained after testing. Although, we were able to obtain the probabilistic values of the questions, half of the predicted answers were wrong. This is expected as the words in the question are independent of each other and as such the answer with the highest probability will be the predicted choice regardless of the words that are surrounding it. It should also be noted that, computing the entire probability of the sentence is unnecessary since all other values but the predicted answers will be the same; a comparison of the individual probability values of the answer would suffice.

Table 1: Accuracy of language models in text completion

	Unigram	**Bigram**	**Bigram with add-1**
Accuracy	50%	70%	90%

The **bigram** language model obtained an accuracy of 70% after testing. It was however unable to predict answers for some of the questions as it returned 0 probability values for both answers. This is mostly due to lack of a *context_word* combination in the training set. However, with regards to the predicted answers that were found in the training set, it achieved a 100% accuracy with the testing. The model will however be unable to find a probability in the bigram language model and return *ZeroDivisionError* when a context is unavailable in the training context. This issue is resolved by applying a smoothing technique to the model.

As expected, the **bigram with add-1 smoothing** language model performed better than the other models; achieving a 90% accuracy value. The nature of this model allows it to cater for missing words in the training set, thus preventing it from returning 0 probabilites. This high accuracy value can be attributed to the models ability to use the contextual nature of the bigram in addition to the smoothing technique that pretends to have seen all instances of the test data in the tranining data.

4 Conclusion

Based on our results, we can see that the **bigram with add-1 smoothing** outperforms the **unigram** and **bigram** models by far. It should also be noted that, if a word in the test question is not found in the training data (probably because of a typographical error or improper pre-processing), it will cause the **unigram** and **bigram** models to return 0 probabilities regardless of whether or not the answers to be predicted are in the training set. As such, using just the answers to be predicted or the context of the words surrounding the answers as is the case with the **unigram** and **bigram** models respectively can help reduce the occurences of 0 probabilities.