# NLP Assignment 2 : Language Modelling

180128022

March 2019

## 1 Introduction

In this assignment, three language models have been implemented which help to complete sentences by suggesting a word (from a list of candidate words) with the highest probability. The three language models are Unigram, Bigram and Bigram with Add-1 Smoothing.

## 2 Implementation

The functions in use are:

1. **ngram_func**: This is the function that operates on the training data. It takes the training file, 'news-corpus-500k.txt' and the choice of language model, n (1 for unigram, 2 for bigram) as arguments. It reads the file, line by line, converts the text into lowercase and upon getting the frequency of the words, returns a dictionary of each word with its respective probability.

2. **ngrams_generation**: This function is called by the above function to do the pre-processing task of tokenising, thereby, returning a concatenation of tokens.

3. **unigram_test_read_func**: This Unigram function is used for operating on the test file, 'questions.txt'. Here, two list of words are made. The first list contains the all the first suggestion words in the test file and the second list contains all the second suggestion words for completing each sentence. Their individual probabilities are found in the returned dictionary from **ngram_func** and compared. The comparison leads to a list of ten words (from the two groups of suggestion words) with the highest probabilities. A matching criterion is set, where the correct words (defined in *correct_list*) and the resulting ten words are compared. The number of common words are counted and accuracy is returned.

4. **bigram_test_read_func**: This Bigram function is similar to the above Unigram function, howeverthe imoplementation is a somewhat different. Here

also, there are two lists (formed by taking into account the two groups of suggestion words) but, they are stored with the words appearing before and after '____' in each sentence, as the word likely to be grouped with those words. Their probabilities are measured in this way: If a group of words from the test file is present in the Bigram dictionary of training data, then the overall probability is measured as the probability of that group of words in the test file divided by the probability of the word appearing before and after '____' in the training file for each iteration, else the probability is 0. This is done for both the groups of suggestion words. Then, as above, after comparing the probabilities, the group of words are pitted against a list of correct group of two words (*correct_bigram_list*), and the accuracy is returned.

5. **smooth_bigram_test_read_func**: This Bigram with Add-1 Smoothing function does exactly as the Bigram function, albeit the overall probabilities are set by adding 1 so that possibility of a group of two words with probability 0, is avoided.

## 3 Accuracy of Each Model

The following table describes the accuracy obtained in each case:

| Model | Accuracy |
|---|---|
| Unigram | 0.6 |
| Bigram | 0.8 |
| Bigram with Add-1 Smoothing | 0.9 |

## 4 Discussion on the Results

Accuracy of the Unigram model is 0.6 and that of the Bigram model is 0.8. This is because the possibility of probability being 0, can be true. However, with smoothing, the accuracy increases to 0.9 as expected, as smoothing nullifies all the chances of getting a 0 probability. Also in the code, the multiplication of both the possible biagrams has been done. Consider the below sentence:
My name is '____' Schrodinger. (Erwin/cat)
Only the probabilities of 'is Erwin' and 'Erwin Schodinger' and 'is cat' and 'cat Schrodinger' are to be taken into account in order to get the correct answer as none of the other probabilities matters, which can be only done if the two probabilities are multiplied.