# Natural Language Processing

## Assignment 2

Pritish Wadhwa 2019440
Rohan Jain 2019095

## Assumptions

- Each line of the input file contains text which is already tokenized.
- Each question has at least 1 masked token and the prediction corresponds to the first masked token in the question.
- **`validation.jsonl`** and **`train.txt`** are in the same directory as the ipynb notebook
- Output is stored in **`output.txt`**

## Preprocessing

- We added **`<start>`** and **`<end>`** tokens in every sentence
- We removed punctuation tokens like: **`, . ? !`**
- We did not standardise the data by lowering all the characters because it was giving lesser accuracy than that without standardisation.

## Methodology

1. Co-occurrence matrix is stored in the form of a dictionary having keys as bigrams. We used **pd.DataFrame** at first but the training time was in the range of 2 hours. However, using a dictionary turned out to be much faster with training time less than 5 minutes.
2. We iterated over multiple values of k for each model, and found that smaller values, ~$10^{-5}$ worked the best for most models.
3. For the bonus question, we considered the **$P(w_{i+1} \mid option)$** for each option and chose the **$\max(P(w_{i+1} \mid option) * P(option \mid w_{i-1}))$** over all options.

4. **P(option | $w_{i-1}$ )** is based on the models. The formulas used are the ones given in lecture:
    a. Normal

$$P(x \mid y) = \frac{C_{y,x}}{C_y}$$

    b. Add-1

$$P(w_i \mid w_{i-1}) = \frac{count(w_{i-1}w_i) + 1}{count(w_{i-1}) + V}$$

    c. Add-k

$$P(w_i \mid w_{i-1}) = \frac{count(w_{i-1}w_i) + k}{count(w_{i-1}) + kV}$$

$$P(w_i \mid w_{i-1}) = \frac{count(w_{i-1}w_i) + \frac{m}{V}}{count(w_{i-1}) + m} \qquad \text{let, } m = kV$$

$$P(w_i \mid w_{i-1}) = \frac{count(w_{i-1}w_i) + mP(w_i)}{count(w_{i-1}) + m}$$

## Accuracy of the models

| **Model** | **Accuracy (%)** |
|---|---|
| Without any preprocessing (all the 3 models) | 54.8 |
| After adding start and end tokens(all the 3 models) | 55.9 |
| After standardizing the text i.e. making every word lowercase (all the 3 models) | 52.75 |
| After removing **, . ? !** (all the 3 models) | **56.2** |
| After removing all punctuation marks (all the 3 models) | 54 |
| After removing stopwords (all the 3 models) | 52.9 |
| Stemming | 42.1 |

### Final Accuracies

Bigram model: 56.2 %
Bigram model with add-1 smoothing: 56.2%
We have implemented add-k smoothing with all the given formulas in the lecture slides. However, we got the same accuracy from all the models = 56.2%. (k = 0.00001)

**Bonus**
Bigram model: 74.45 %
Bigram model with add-1 smoothing: 68.8%
Add-k smoothing without using m = k*V is 74.35% (k = 0.001)
Add-k smoothing with m = k*V is 74.45% (k = 0.00000001)
Add-k smoothing with m = k*V and using P(wi) is 74.7% (k = 0.000001)

## Contribution

Both the members contributed equally in coding, making readme and preprocessing of the data.