# Natural Language Processing: Computer Assignment #2

Due on April 24, 2023

**Tohid Abdi**

# Problem 1

In the first step, we upload the kaggle API in Google colab and download the data using the appropriate code. With the help of engine argument and setting it to "python", we separate the columns and get the appropriate data frame. Using the appropriate code, we use 20% of data, while maintaining the appropriateness of the classes. Then we split data to train and test.

**Part One**
To do this part, we use the "hazm" library, which is suitable for Persian text. We perform normalization and tokenization.

**Part Two**
With the help of appropriate functions, we use the training data and obtain the "tf-idf" and "PPMI" co-ocurrence matrices for these data. For tf-idf, we can calculate the values of "term frequency" and "inverse document frequency", and PPMI is obtained from the calculation of probabilities (done in code).

**Part Three**
For classification, I use Gaussian NaiveBayes classifier once and Multinomial Naive Bayes classifier another time. With the help of each of these classifiers, we obtain the recall, precision, and F1-score metrics for both feature extraction methods. These values are shown in Table 1.

Table 1: Evaluation criteria for Gaussian Naive Bayes and Multinomial Naive Bayes classifiers by extracting features with tf-idf and PPMI methods

| - | **GNB for tf-idf** | **MNB for tf-idf** | **GNB for PPMI** | **MNB for PPMI** |
|---|---|---|---|---|
| **Recall** | 0.38 | 0.89 | 0.28 | 0.73 |
| **Precision** | 0.67 | 0.78 | 0.6 | 0.72 |
| **F1-Score** | 0.48 | 0.83 | 0.38 | 0.72 |

As can be seen, the use of Multinomial Naive Bayes Classifier performs a better classification than Gaussian Naive Bayes Classifier. The obtained results show that feature extraction with tf-idf method produces better results than PPMI. Also, Multinomial Naive Bayes model obtains the best results with the help of feature extraction with tf-idf.

# Problem 2

**Part One**

We receive the data using their link. We perform the necessary pre-processing, which includes removing special characters and punctuation, tokenizing text into sentences and words, converting to lowercase, removing stopwords, and lemmatizing. We create the skipgram model according to the problem specifications and train it for 15 epochs. The value of the learning rate is 0.03. We get the representation matrix with the summation of W and C vectors.

**Part Two**

With the help of Principal Components Analysis, we reduce the dimensions of the vectors to two. The expected words and their differences are shown in Figure 1.

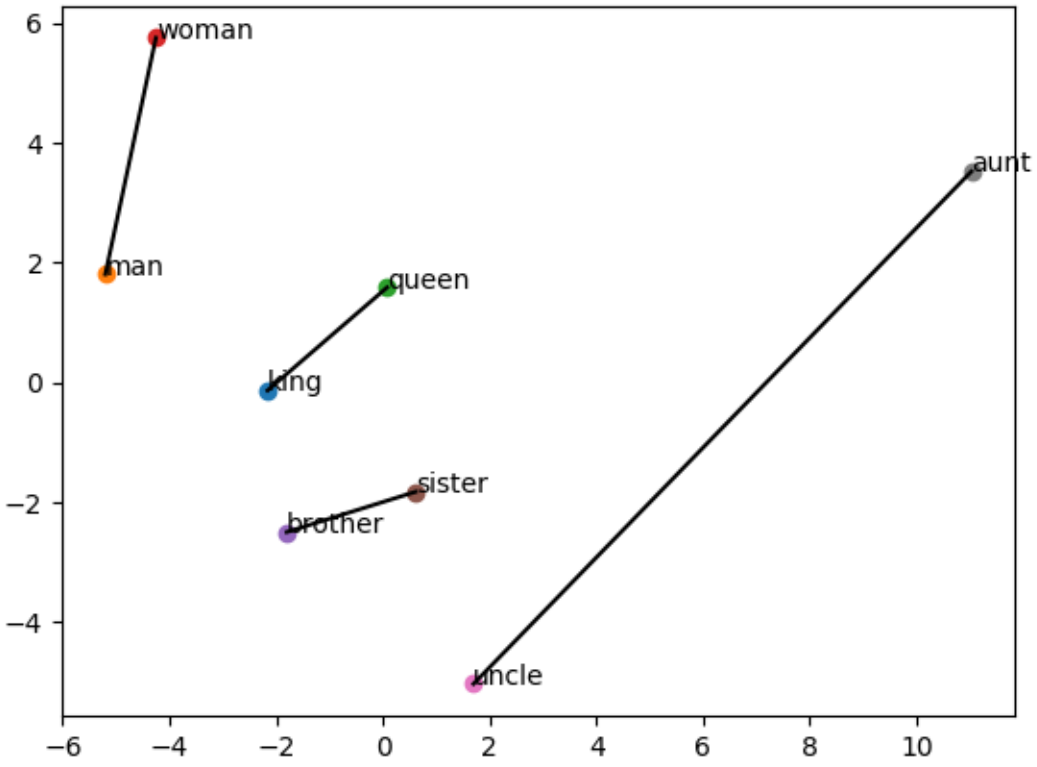


Figure 1: Eight words and their differences in two dimensions

The figure illustrates ability of the model to automatically organize concepts and learn implicitly the relationships between them, as during the training we did not provide any supervised information about differences of genders.

# Problem 3

With the help of Kaggle API, we download the csv file and create the dataset by separating the columns. We do basic pre-processing, including conversion to lowercase letters, remove punctuation marks, tokenization, stop word removal, and lemmatization. In the next step, according to the problem, we train the skipgram model and obtain the word representation vectors. Also, we divide the data into two parts, training and testing.

**Part One**
We train the Logistic Regression model with the help of the assumptions of the problem. We evaluated this model with the help of test data and the following values were obtained:
Precision: 0.71, Recall: 0.71, F1-score: 0.71
The confusion matrix is shown in the notebook.

**Part Two**
The imbalanced distribution of classes in the dataset can have several effects on the logistic regression model.

First, the model may tend to predict the majority class more often than the minority classes, leading to biased predictions. In this case, the majority class is the neutral class, and the model may have a tendency to predict the neutral class more often than the positive or negative classes.

Second, the model may have difficulty learning the features that are important for predicting the minority classes. This is because the number of samples for the minority classes is relatively small compared to the majority class. As a result, the model may not be able to capture the nuances of the minority classes and may have lower accuracy in predicting them.

Third, the imbalanced distribution of classes can affect the evaluation metrics of the model. For example, if the model has high accuracy in predicting the neutral class but low accuracy in predicting the positive and negative classes, the overall accuracy may appear to be high, but the model may not be useful in practical applications where predicting the minority classes is important.

To mitigate the effects of the imbalanced distribution of classes, techniques such as oversampling the minority classes, undersampling the majority class, or using weighted loss functions can be used. These techniques aim to balance the distribution of classes in the dataset and give more importance to the minority classes during training.

**Part Three**
In the case of imbalanced datasets, Naive Bayes can often outperform Logistic Regression due to its assumption of conditional independence between features, which allows it to handle high-dimensional data and noisy data well. This assumption can often lead to better performance on imbalanced datasets, as the classifier can still make accurate predictions even when the number of instances for a particular class is small.

On the other hand, Logistic Regression does not make any assumptions about the distribution of the features and can handle non-linear relationships between the features and the target variable. However, when dealing with imbalanced datasets, Logistic Regression can be biased towards the majority class due to the imbalance in the data, which can lead to poor performance for the minority class.

Therefore, in the case of imbalanced datasets, Naive Bayes may provide better performance compared to Logistic Regression due to its ability to handle noisy data and its assumption of conditional independence

Problem 3 continued on next page... 

between features, which can lead to better performance on imbalanced datasets. However, it is always recommended to experiment with both algorithms and compare their performance on the specific dataset to determine which one works best for the problem at hand. In this issue, the Naive Bays model was also evaluated and the following results were obtained:
Precision: 0.71, Recall: 0.71, F1-score: 0.71