# Distributional Semantics

Khalid Salman

Using a bag of words model without any improvements resulted in a mean rank of 4.5. In this report I will explain how I was able to get a mean rank of 1 (classify all characters correctly).

## Q1: Improve pre-processing

I started by removing stop words because they occur frequently in most documents hence, they are not good classifiers. This led to the mean rank dropping to **2.875**. After that I moved on to remove punctuation because I wanted words such as "Lesly!" and "Lesly" to be the same thing. This had a minor improvement and the mean rank decreased to **2.625**. Then I tried both stemming and lemmatization, I found that stemming was increasing the mean rank to **2.875** while lemmatization decreased the mean rank to **2.4375** so I used only lemmatization. Finally, I tried converting all words to lower case, but this had a negative impact on the mean rank as it increased to **2.6875**.

My final configuration for Q1 is stop words removal + punctuation removal + lemmatization. This configuration had a mean rank of **2.4375** and **9 correct classifications out of 16**.

**Note:** Even though I am planning on removing the_EOL_ tag, I decided to leave it to the feature extraction stage because I want to be able to know when a sentence end (to use sentiment analysis and pos tagging on a sentence level.)

## Q2: Improve linguistic feature extraction

In this part I build on the best configuration in Q1 and try adding different features.

First, I removed the "_EOL_" tag from the count and tried adding 2 grams counts. This helped significantly as the mean rank dropped to **1.5** with 9 correct classifications. Then I removed the 2 grams counts and used 3 grams counts instead, the mean rank was **1.5625** with 8 correct classifications (worse than 2 grams but better than not using either 2 or 3 grams). The previous experiments led me to try adding both 2 grams and 3 grams, doing so resulted in a mean rank of **1.5** but this time with 10 correct classifications.

After that I moved on to try adding pos tags. First, I tried adding them as a count (the number of occurrences of each pos tag). However, this didn't produce better results, so I tried adding the pos tags as word@postag counts. Doing so had a noticeable improvement as it decreased the mean rank to **1.375** with 12 correct classifications.

Then, I tried using sentiment analysis. I count the number of sentences with positive sentiment and the number of sentences with negative sentiment. This feature had a negative result. The mean rank increased to reach **1.6875.**

Finally, I selected the best k% features by trying all numbers from 1% to 99% on the validation dataset and I found that using k = 20% leads to the best results (mean rank of **1.3125**).

My final configuration for Q2 is remove _EOL_ + use 2 grams + use 3 grams + word@postag + select best 20% features. This configuration had a mean rank of **1.3125** and **12 correct classifications out of 16.**

## Q3: add dialogue context data and features

This part builds on the previous configuration and incorporates previous line, next line, and scene information.

First, I tried using previous line and scene information (making sure that scene information occurs only once per scene even if the character has more than one line in one scene) I also made sure that the features are added with a unique identifier “prev_” and “scene_”. This is because I want a way to distinguish between a word that occurred in the previous line and the same word when it is said in the current line. Doing so resulted in notable enhancement as the mean rank reached a low of **1.0625** with 15 correct classifications.

Then. I added next line information (also making sure to put the unique identifier next_). Even though this feature didn’t reduce the mean rank, looking at the confusion matrix, this feature enhances the separation between the classes (this will result in a better generalization).

My final configuration for Q3 is use previous line + scene information + next line. This configuration had a mean rank of **<u>1.0625</u>** and **15 correct classifications out of 16**.

**<u>Q4: improve vectorization method</u>**

This part builds on the previous configuration and uses TFIDF transformation.

In this part I used the TFIDF transformation to transform the count feature matrix to TFIDF matrix. Although doing so didn’t change the mean rank, it made a very noticeable difference in the confusion matrix! Classes are now separated more comfortably. OfCourse this is a good thing because it means that we have a higher chance of generalizing better in the test dataset.

My final configuration for Q4 is same as Q3 + TFIDF transformation. This configuration had a mean rank of **<u>1.0625</u>** and **15 correct classifications out of 16**.

**<u>Q1,2,3,4: Further improvements</u>**

To this point, my configuration was miss classifying only one character. Looking at the confusion matrix, the confusion was between “Clare” and “Sean”. Knowing this, I decided to look at lines in the dataset said by Clare and Sean aiming to come up with some features that may help in reducing the confusion between them.

I decided not to remove punctuation at the pre-processing stage, instead, I keep the punctuation to the feature extraction stage and extract how many times each character used each punctuation and only then remove the punctuation. This approach proved beneficial as I was able to get a mean rank of **1.00** (16 correct classifications).

**<u>Q5: Select and test best vector representation method</u>**

This part tests using the best vector representation obtained by using the best configuration on the test dataset.

<u>My best configuration is:</u>

stop words removal + punctuation removal + lemmatization + remove _EOL_ + use 2 grams + use 3 grams + word@postag + select best 7% features + previous line + scene information + next line + TFIDF transformation

Doing so resulted in a **<u>mean rank = 1.00</u>** in the test dataset.