# Coursework 2: Vector Space Semantics for Similarity between Friends Characters

# Q1. Improve pre-processing (20 marks)

**Solution 1**- To improve pre-processing, I have added extra steps post tokenization to filter out the unwanted tokens. The techniques I have tried are as follows: -

1. **Lowercasing** - This will convert each doc of corpus into lower case.
2. **Tokenization** - Using word\_tokenize to generate tokens for the character text.
3. **Punctuation** **Removal** - This will remove any punctuation in the character text.
4. **Stopwords** **Removal** - This will remove stopwords include common words such as "the," "and," "is," "in," and so on.
5. **Stemming** - This will reduce words to their base or root form. It involves removing suffixes from words to obtain a common base form.
6. **Special** **Character** **Removal** - This involves eliminating non-alphabetic characters.
7. **Handling** **Contractions** - Handling contractions involve expanding contractions in text. For example, converting "don't" to "do not" or "I'll" to "I will."
8. **Removing** **HTML Tags and URLs** - Removing HTML tags and URLs involve eliminating HTML tags and web links from text data.
9. **Handling Accented Characters** - It involves converting accented characters (e.g., é, ü) to their non-accented counterparts (e.g., e, u).
10. **Handling Rare Words** - It removes words that appear only once.
11. **Lemmatization** - It reduce words to their base or dictionary form (lemma). Unlike stemming, lemmatization considers the context and aims to transform words to their canonical or meaningful base form.

NOTE:- The pre and post metrics are given at last.

# Q2. Improve linguistic feature extraction (30 marks)

**Solution 2-** Optimization in **to\_feature\_vector\_dictionary** function: - It includes adding basic features like length, number of words, and average word length for the character doc in corpus. Post that I added **N-gram** where I have used Bi-gram, Tri-gram, Tetra-gram. Another technique I used is **part-of-speech (POS)** tagging.

Optimization in **create\_document\_matrix\_from\_corpus** function: - It includes **TF-IDF** and **k-best selection** technique to generate improved document matrix from corpus.

This function also handles the fitting or transforming of the documents based on the value of the fitting parameter:

* If **fitting is True**, it means this is the training phase, and the vectorizer and feature selector need to be fitted on the training data. It fits the vectorizer and transforms the documents using both the vectorizer and feature selector.
* If **fitting is False**, it means this is the testing or validation phase. It only transforms the documents using the pre-fitted vectorizer and feature selector.

The function returns the transformed feature matrix, the fitted vectorizer, and the fitted feature selector. This allows you to use the same fitted vectorizer and feature selector when transforming new data during the testing or validation phase.

NOTE:- The pre and post metrics are given at last.

**Q3. Add dialogue context and scene features (15 marks)**

**Solution 3-** Adjusted the method ***create\_character\_document\_from\_dataframe*** which aims to generate character documents with context by joining each character's lines into a single string and including lines spoken by other characters in the same scene as context.

**The method involves Grouping by Episodes and Scenes, Iterating Over Episodes and Scenes, Iterating Over Characters in the Scene, Processing Each Character's Lines, Processing Context Lines, Combining Character's Lines with Context.**

Hence, **it incorporates the context of the line spoken by the characters in terms of the lines spoken by other characters in the same scene.**

NOTE:- The pre and post metrics are given at last.

**Q4. Parameter Search (15 marks)**

**Solution 4-** It is a good practice to conduct a systematic parameter search instead of a random search as this will give you more reliable results. The code to performs a **grid search** over a specified set of hyperparameters can be found in jupyter notebook, attempting to find the combination that yields the best mean rank. The range of values in param grid were 'k\_best': [chi2], 'k\_feature': [100, 200, 'all'], 'mindf': [5,10]

It uses these hyperparameters to create a **TF-IDF vectorizer** and a **feature selector**, then evaluates the performance on a validation set. The best hyperparameters and their corresponding mean rank are for bestaccuracyof 1.3 is **{'k\_best': chi2, 'k\_feature': 200, 'mindf': 5}.**

Training & Validation Result: - **mean rank 1.3**| **mean cosine similarity 0.8190194121795077** | **8 correct out of 10 / accuracy: 0.8.**

# Q5. Analyse the similarity results (10 marks)

**Solution 5-** We will refer heat map to identify the heldout character vectors ranked closest to each other character's training vector and those furthest away.

The **Chandler** & **Joey** and **Ross** & **Joey** are the **closest characters,** the reason being highest similarity of 0.96 and 0.962 respectively. This implies that Chandler and Joey could exhibit resemblances in their language patterns, engage in similar topics, or have interactions that result in similar representations of their dialogue.

The **Chandler** & **Rachel** and **Joey** & **Rachel** are the **furthest characters,** the reason being highest similarity of 0.932 and 0.935 respectively.

Various linguistic and contextual factors, including dialogue context, vocabulary, topics, interactions with other characters, amount of dialogue, and speech style, should be considered to determine the highest match between the target character in the training set and their vector in the held-out set. It's important to acknowledge model limitations, such as the choice of features (e.g., n-grams) and the vectorization technique (e.g., TF-IDF), which may not effectively capture the nuances required for character distinction.

# Q6. Run on final test data (10 marks)

# Solution 6- The final values after improved pre-processing, feature extraction, grouping feature for context and using the optimized hyperparameter are as follows: -

The model performs well with a **mean rank of 1.1**, indicating the target character's document is consistently ranked high. The **mean cosine similarity of 0.9687** signifies strong similarity between the target character's document and test documents of the same class. The model correctly predicted 9 out of 10 documents, resulting in an **accuracy of 90%.**

|  |  |  |
| --- | --- | --- |
|  | **Training and Validation** | **Training and Testing** |
| **First run** | mean rank 4.2  mean cosine similarity 0.89157254047686  1 correct out of 10 / accuracy: 0.1 | mean rank 4.0  mean cosine similarity 0.8925164628242  3 correct out of 10 / accuracy: 0.3 |
| **Optimised Pre- processing steps:** | mean rank 3.8  mean cosine similarity 0.990134754654987  3 correct out of 10 / accuracy: 0.3 | mean rank 3.3  mean cosine similarity 0.989790382449  5 correct out of 10 / accuracy: 0.5 |
| **Optimised feature extraction steps:** | mean rank 4.8  mean cosine similarity 0.062866284486955  2 correct out of 10 / accuracy: 0.2 | mean rank 4.9  mean cosine similarity 0.0777677971579  1 correct out of 10 / accuracy: 0.1 |
| Add dialogue context | mean rank 1.0  mean cosine similarity 0.99577204166123  10 correct out of 10 / accuracy: 1.0 | mean rank 1.0  mean cosine similarity 0.9876407648285  10 correct out of 10 / accuracy: 1.0 |
| Final Run after grid search | mean rank 1.2  mean cosine similarity 0.96598026854915  8 correct out of 10 / accuracy: 0.8 | mean rank 1.1  mean cosine similarity 0.8587978028259  9 correct out of 10 / accuracy: 0.9 |