TEXT MINING

This project of Text Mining was developed by Andrea Cardinali (911556) and Adonis Kingsley Granita ( ) for the course of Text Mining & Search.

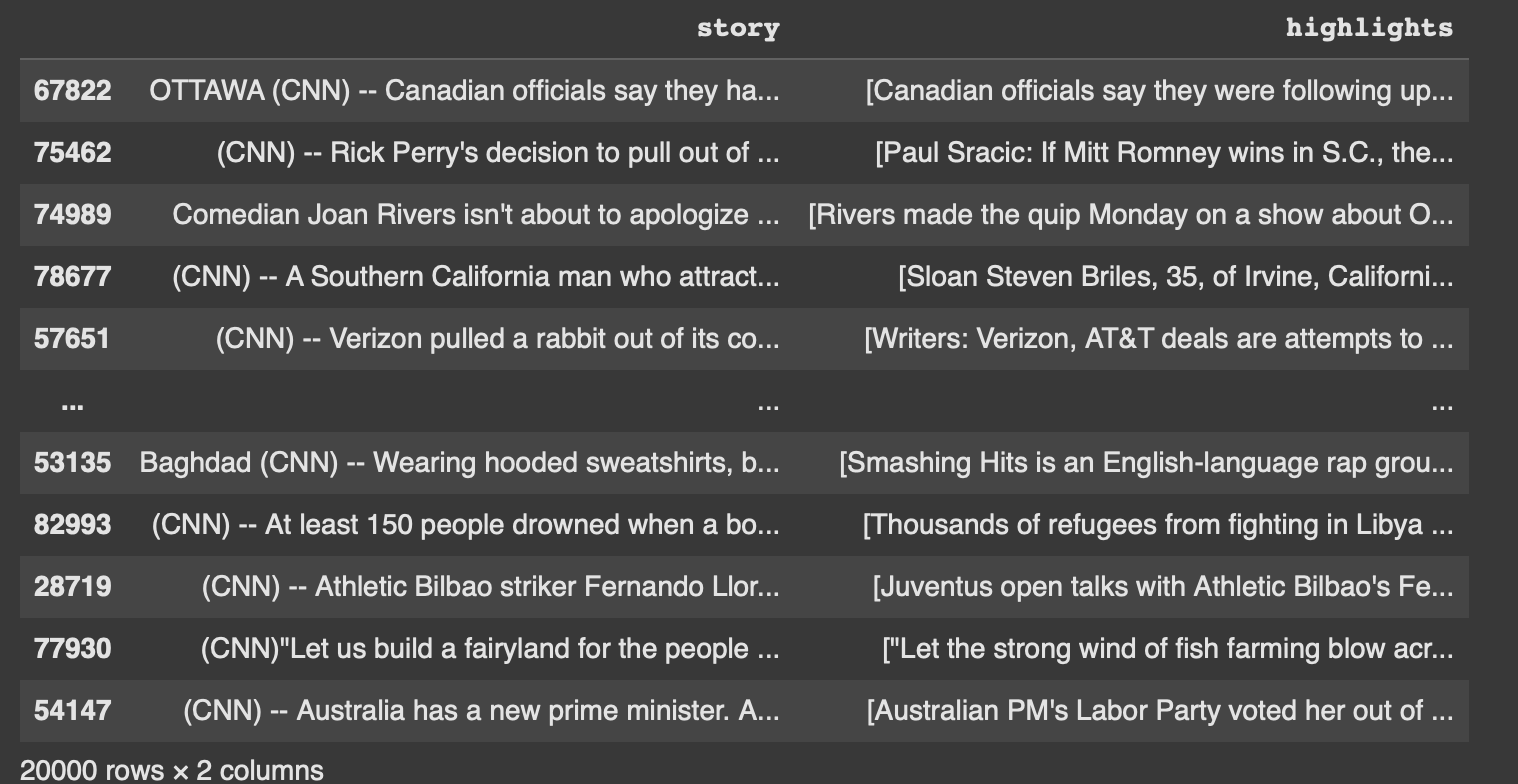
The aims of this project are to perform two text mining tasks (text summarization and text classification) on two different repositories of texts.

# TEXT SUMMARIZATION

### INTRO

This project involved performing extractive text summarization on a dataset of CNN articles. A sample of 20,000 articles was extracted from the dataset to serve as the basis for the summarization process.

The dataset contained articles and their corresponding sentences, which were used as ground truth to evaluate the summarization.



The summarization models applied were Latent Semantic Analysis (LSA) and TextRank, both of which are widely used techniques for extractive summarization.

### PREPROCESS

The text representation used for both summarization models, Latent Semantic Analysis (LSA) and TextRank, was based on Term Frequency-Inverse Document Frequency (TF-IDF). To optimize the quality of the TF-IDF representation, a thorough preprocessing pipeline was implemented to clean and standardize the text data.

A custom stopwords list was created to include additional context-specific words such as “cnn,” “say,” “said,” “new,” “wa,” and “ha.” This was achieved using a function that allowed the customization of stopwords by adding or retaining specific words based on the needs of the dataset.

The preprocessing workflow involved several key steps. Regular expressions were used to remove unwanted patterns, punctuation, and non-alphanumeric characters. The text was converted to lowercase for consistency, and contractions were expanded to their full forms. Words were lemmatized to reduce them to their dictionary root forms, ensuring that different inflections of the same word were treated uniformly. Optional stemming was available but was not applied, as lemmatization preserves the semantic meaning of the words. The custom stopwords list was then applied to filter out irrelevant words, further enhancing the quality of the resulting text representation.

Although the advanced preprocessing pipeline seemed theoretically superior, simpler preprocessing methods were also tested and unexpectedly produced better results based on evaluation metrics. However, the models utilizing the advanced preprocessing were still preferred. We assumed that the ground truth might not have been robust enough to accurately evaluate the nuanced improvements provided by the detailed preprocessing. The theoretically sound approach was deemed more reliable for practical applications.

### LSA

Latent Semantic Analysis (LSA) was applied as one of the methods for extractive text summarization. The process begins with the creation of a Term Frequency-Inverse Document Frequency (TF-IDF) matrix to represent the text corpus numerically. This matrix is computed using the TfidfVectorizer, which was configured to include a maximum of 5,000 features and to exclude standard English stopwords for optimal performance.

Once the TF-IDF matrix is constructed, LSA is performed using the TruncatedSVD algorithm. This method reduces the dimensionality of the TF-IDF matrix by focusing on the top n\_components, chosen to be 5, latent topics, where each component represents a significant theme within the text. This step highlights the most relevant structures in the text, reducing noise and redundancy in the data.

Each sentence's importance is then quantified by computing scores based on the squared sum of its contributions across all latent topics. These scores provide a measure of how well a sentence represents the key themes extracted by the LSA.

The final step involves selecting the top-scoring sentences to generate the summary. The indices of the highest-scoring sentences are identified, and the corresponding sentences from the original text are extracted to form the summary. The approach ensures that the summary retains the most critical and thematically relevant information from the document.

This pipeline was implemented using a function that processes each document iteratively, enabling scalable summarization of large datasets. The modular structure of the pipeline allows flexibility in configuring the number of components for LSA and the number of sentences included in the summary, making it adaptable to different use cases.

### TEXT-RANK

TextRank was employed as another method for extractive text summarization. This algorithm, inspired by Google’s PageRank, ranks sentences based on their importance within the context of the entire document. The process begins by splitting each article into sentences using natural language tokenization. These sentences form the foundation for the subsequent analysis.

To capture the relationships between sentences, a TF-IDF matrix is computed for the entire set of sentences in the document. This matrix is then used to calculate a cosine similarity matrix, where each element represents the similarity between two sentences based on their TF-IDF vectors. The cosine similarity matrix effectively serves as the adjacency matrix for a graph representation of the document, where nodes correspond to sentences, and edges reflect their similarity.

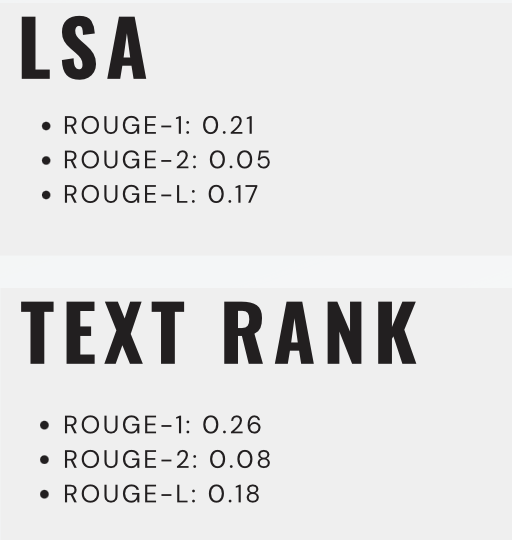
A graph is constructed from this similarity matrix, and the PageRank algorithm is applied to calculate the importance of each sentence. This step assigns scores to sentences, reflecting their centrality and relevance within the graph. Sentences with higher scores are deemed more important and are prioritized for inclusion in the summary.

The final summary is generated by selecting the top-ranked sentences according to their PageRank scores. The number of sentences in the summary is configurable, allowing flexibility based on the desired level of detail.

This method ensures that the generated summary captures the most contextually significant sentences, providing a concise and coherent representation of the original text. The pipeline was applied to each document in the dataset using an automated approach, making it scalable for large-scale summarization tasks.

### EVALUATION

The evaluation of the summarization models was conducted using ROUGE (Recall-Oriented Understudy for Gisting Evaluation) metrics, which compare the overlap of unigrams, bigrams, and the longest common subsequence between the generated summaries and the ground truth references. The generated summaries were evaluated against the cleaned highlights from the dataset, with scores computed for ROUGE-1, ROUGE-2, and ROUGE-L.

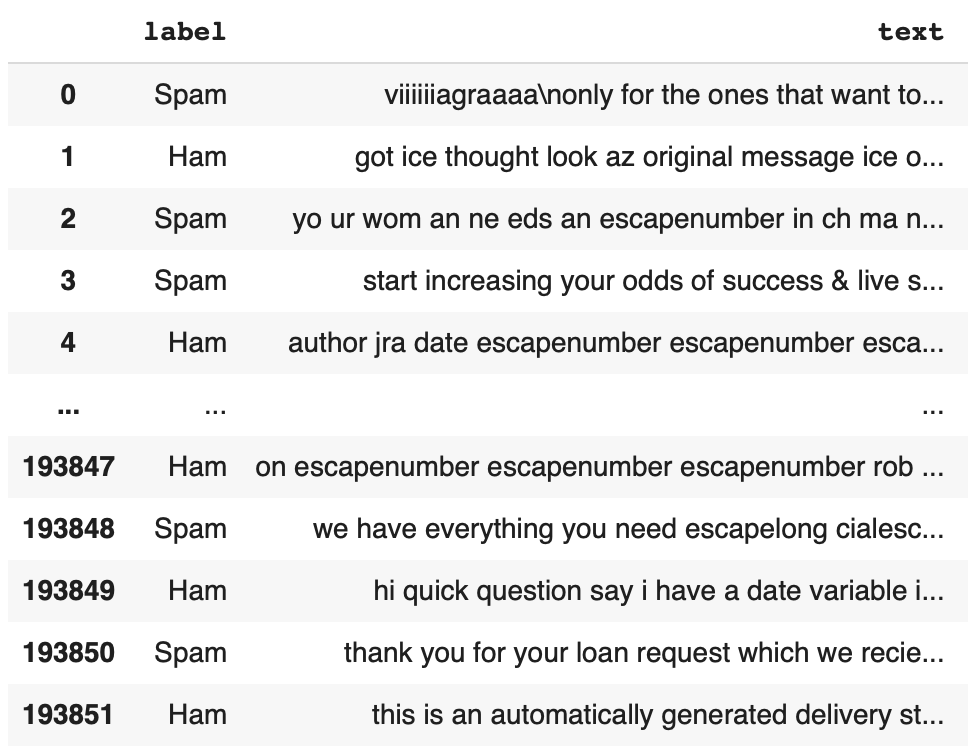


Despite the detailed preprocessing and theoretically robust methodologies, the overall performance of the models, as indicated by the ROUGE scores, was not entirely satisfactory. Among the models tested, TextRank emerged as the better-performing approach, achieving slightly higher scores than Latent Semantic Analysis (LSA). However, the scores suggested that there was still room for improvement in terms of capturing the full essence of the reference summaries.

The underwhelming performance may be attributed to limitations in the ground truth summaries, which might not have been comprehensive enough to accurately evaluate the nuanced quality of the generated summaries. Additionally, the inherent complexity of summarization tasks and the reliance on word overlap metrics like ROUGE may not fully reflect the models’ ability to generate semantically meaningful summaries.

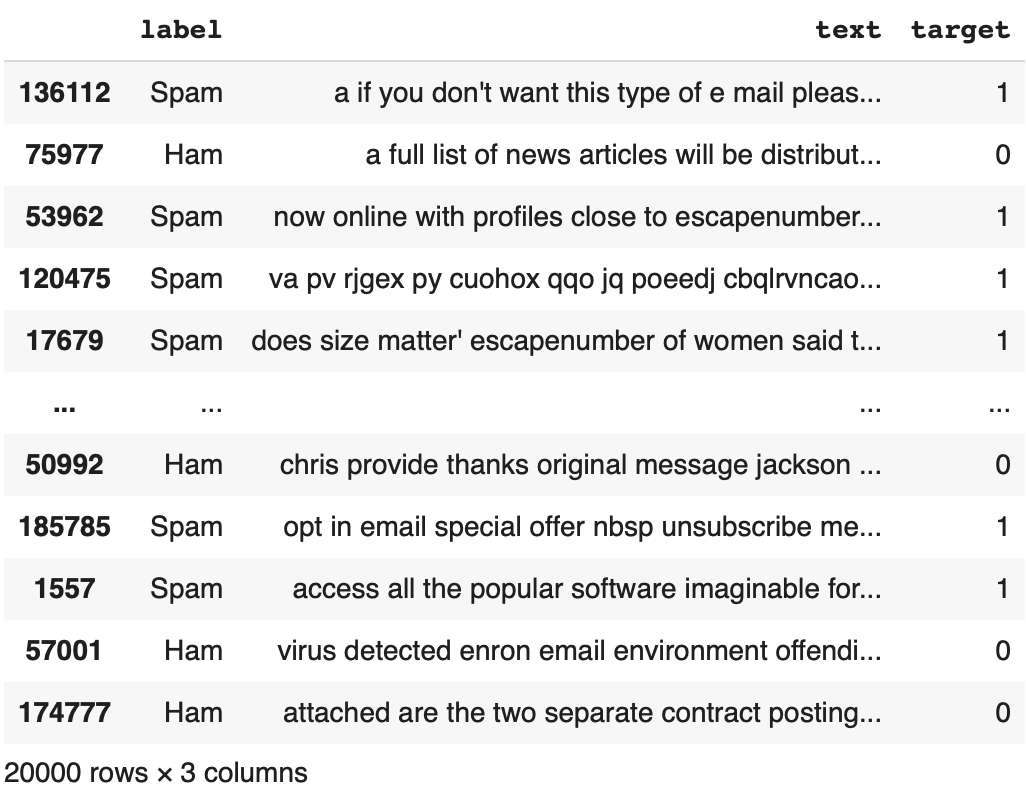
# Text Classification

The text classification task aims at classifying documents into predefined categories. In this case it was chosen to apply text classification to a repository of more than 190K emails that were labelled either as spam or ham (non spam). For each email it’s available the actual text and the corresponding label.



The first step was to add a new variable that is equal to 1 for an email classified as spam and 0 for non-spam. This was done because models don’t work with words directly, but rather with numbers. Another step was to check and remove any null values.

After multiple tries, it was decided to take a much smaller sample size instead of all the emails available. The reason behind this was that the computing resources were not enough to perform a vectorization and apply different classification models. A sample of 20,000 emails was a good compromise between performance and results.



The emails’ text has been vectorized first using a simple count vectorizer, which takes the frequencies of words in the text, then it was decided to use a tf-idf vectorizer, where words are weighted according to their discriminative power, because the ladder yields slightly better results than the former.

Three models were used for this classification task: Decision Tree, SVC and Random Forest. After applying each model, a 5-fold cross validation was carried out to try improving the accuracy of the models.

### Decision Tree

The first algorithm to be applied is the decision tree. It is based on a tree-like structure, starting from a root node and splitting data into subsets according to the features value. This is generally used when interpretability is crucial.

After dividing the emails into training and test sets, the decision tree algorithm was applied and produced rather good results with an accuracy of 91%, recall and F1 both reached 90%.

### SVC

The SVC, or Support Vector Classifier, belongs to the SVM family for classification tasks. Like the SVM, the SVC tries to find the best hyperplane that separates the two classes in the feature space.

Each data is mapped to a high-dimensional space and then it finds the hyperplane that maximises the distance between the two classes.

The results obtained for the SVC were the most accurate, demonstrating the highest predictive accuracy among all models tested. The accuracy and F1 were 97% and recall was 98%.

### Random Forest

This algorithm is based on a collection of Decision Trees. By creating multiple Decision Trees for the training phase, it then combines their output in one for the prediction.

The Random Forest showed an improvement with respect to the Decision Tree, as all three measures are greater than the ones achieved by DT (accuracy: 96%, F1: 95%, recall: 94%).

### Stratified 5-Fold Cross Validation

The results obtained from the three algorithms were promising; however, it was decided to explore whether using a 5-fold cross-validation could yield better results, i.e. 98/99% of accuracy.. The k-fold CV is widely used for assessing the model performance and ensuring that the evaluation is not biased by the training data.

By dividing the data into 5 equally sized subsets, the model is then trained k times where in each iteration a subset is used as a validation set. In this case, a stratified CV ensures that the target class is equally distributed in all the 5 subsets.

For each iteration, the same three algorithms (Decision Tree, SVC and Random Forest) were applied to the subsets, so as to compare the final results of the F1 score.

With no surprise, the F1 score obtained from averaging all the 5 results were the same as the one obtained from the single application of the algorithms. In fact, for the Decision Tree it was 90%, for SVC 97% and finally for the Random Forest 95%. This is no strange because the target class was well represented in the data, so there were not imbalanced subsets.