**Text Summarization**

# 

# **Motivation**

There is always redundant and overlapping information present in the document which leads to wastage of time. Summarizing the documents and extracting important information reduces a large amount of data into a smaller size and saves a lot of time. Manual text summarization is a time-consuming and laborious task, so there is a need for automatic text summarization. Here our objective is to use Natural Language Processing (NLP) to compare two types of implementations, mainly using a library and implementing from a factor-based approach.

# **Background**

Text summarization is one of the most important applications of NLP. In order to understand the history of text summarization, we need to understand the history of NLP. During the second world war in 1940, the Machine Translation (MT) origin of NLP was introduced which translates the Russian Language into English and vice versa using the computer. Later in 1950, there were developments in syntactic theory language and parsing algorithms, but it was not good enough for building efficient MT. Many researchers started giving attention to NLP with the development of Artificial Intelligence. Winograd’s SHRDLU thesis at MIT in 1971 published their work to capture attention outside mainstream NLP. Finally after a decade, during the 1980s and 1990s, NLP started gaining interest as a topic of research and there were many developments in this field.

Earlier summarization was based on rule-based algorithms called “Importance Evaluator”. In this method, according to importance, ranking is given to different parts of the text. There were a lot of developments made in order to understand the importance of sentences in a given corpus. One such method calculates the relatedness of text to other text in the corpus. The neural network for text summarization also plays an important role in the development of NLP. The neural network used a corpus of texts for training and generated summaries by ranking sentences. While training, it also learns about what sentences need to be included in the final summary. Parallel developments have happened in NLP. One such development was the diversity-based approach to extractive summarization. This method calculates the diversity of sentences and also removes redundant sentences from the final summary. For finding diversity, the k means algorithm was used. Other extractive approaches like graph-based and encoder-decoder were also introduced.

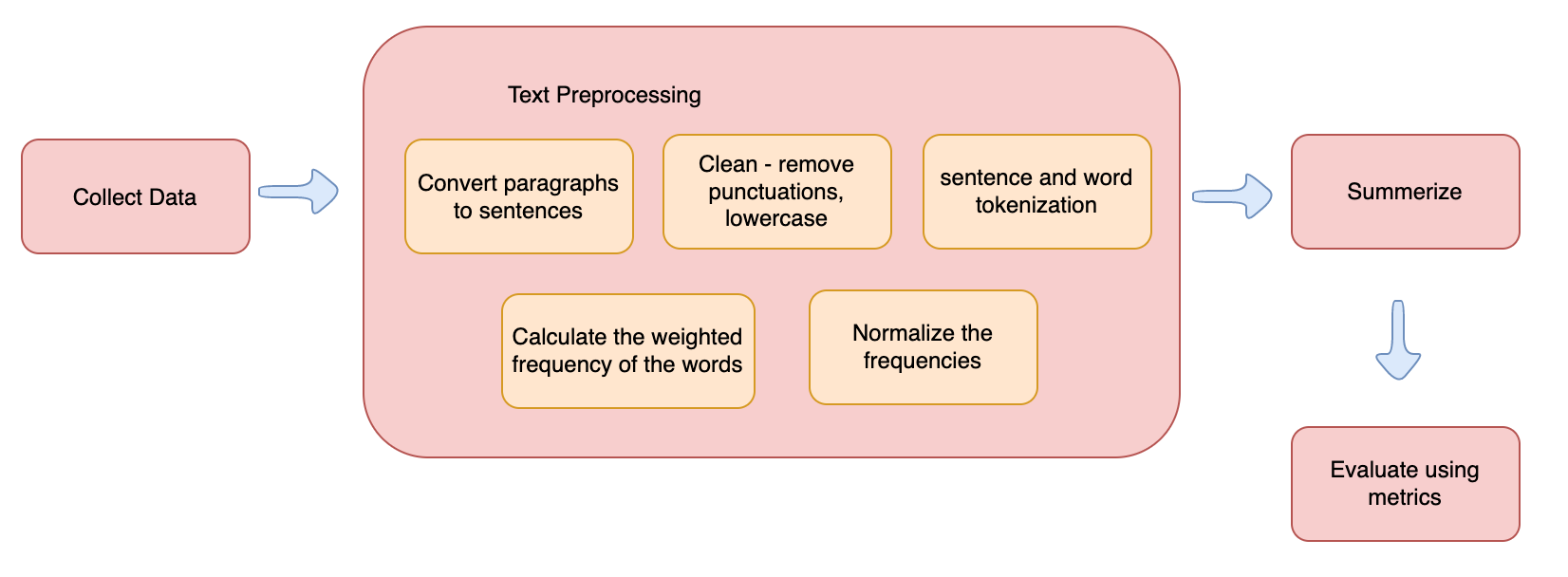
# **Literature Review**

| Author | Year | Method | Description |
| --- | --- | --- | --- |
| Chin-Yew Lin et al. | 2004 | ROUGE-N, ROUGE-L, ROUGE-W, and ROUGE-S | Automatic evaluation package for text summarization |
| Dharmendra Hinhu et al. | 2015 | Extractive Text summarization approach | Used Wikipedia Articles as input to the system and identifies the text scoring |
| Pankaj Gupta et al. | 2016 | SVM, Naive Bayes classifier | Sentiment analysis, analyze the sentiments, emotions present in the text |
| Akshil Kumar et al. | 2017 | TextRank, LexRank, Latent Semantic Analysis (LSA) | ROUGE is used for evaluation, TextRank provides better results |
| Tacho Jo | 2017 | K- Nearest Neighbor | Text Summarization viewed as a classification task,  KNN and clustering provides better performance |

ROUGE: Recall Oriented Understudy for Gisting Evaluation

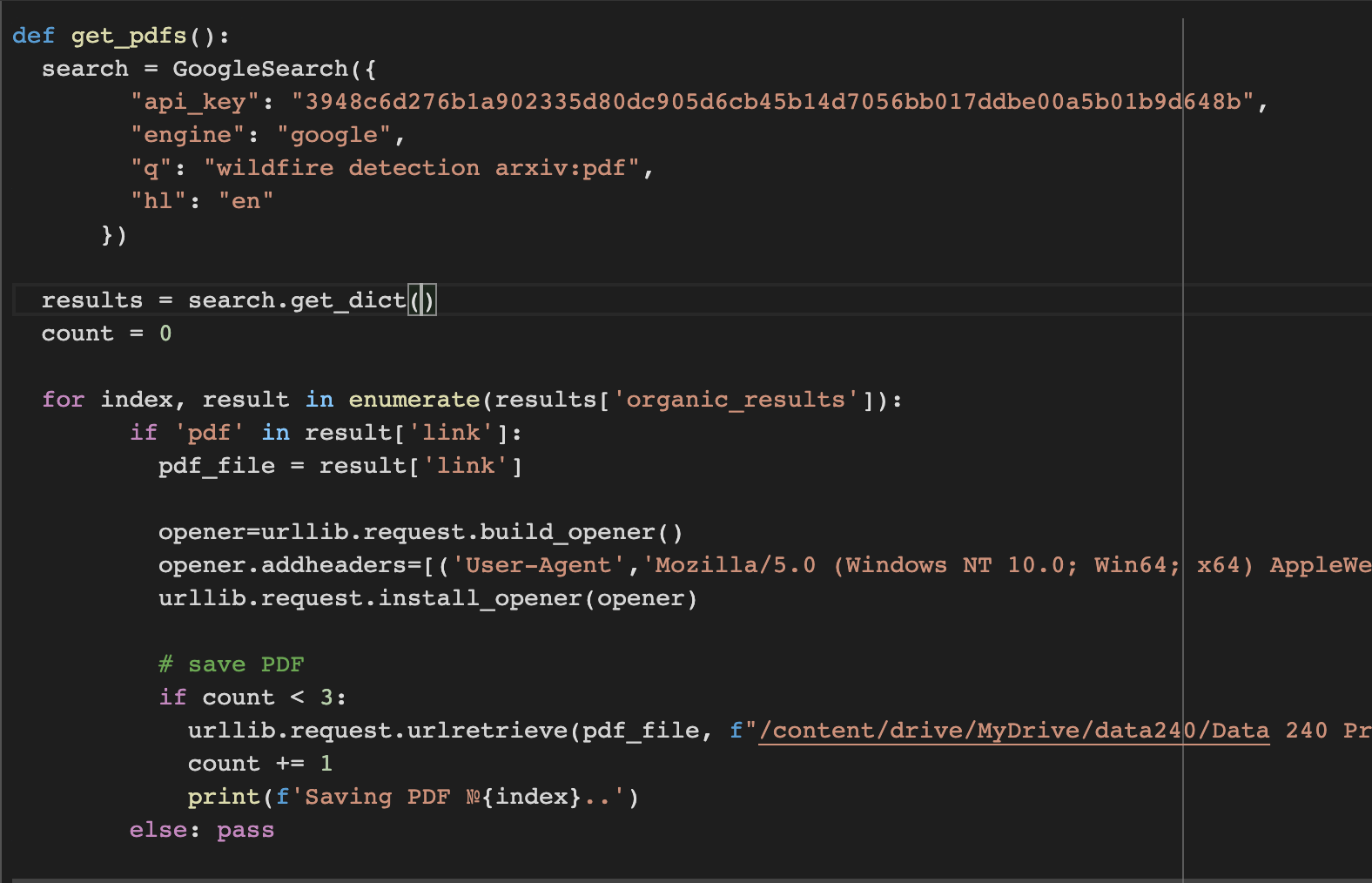
**Methodology**

We are following the Cross-Industry Standard Process for Data Mining (CRISP-DM). We are first collecting data. Then we are doing text cleaning, and pre-processing, and using a different model to generate a summary. Once the summary is created, we are evaluating it. The complete architecture of the project is shown below.



## **Data Collection**

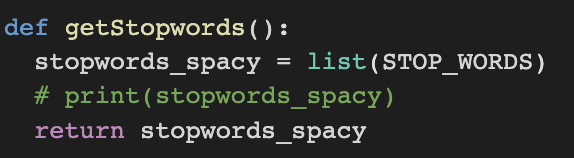
In data collection, we are downloading data by calling the python script. In python script, we are providing query and secret api key created from the serpapi website. We are using the serpapi python library to download pdf articles from the google search engine. Downloaded pdf files are stored in google drive. Once all files are downloaded, we are merging all together for text preprocessing. Below is the snippet of the python function for downloading articles from the google search engine.



## **Pre-processing**

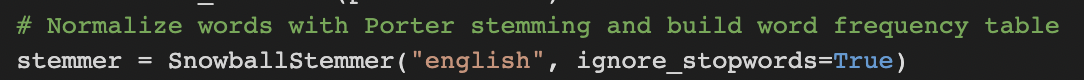
Below are the preprocessing steps that we have used as part of the project.

***Remove the stop words:*** These are the most commonly occurring words in the text such as it, these, those, etc. which don’t provide any valuable information. We have used the stopwords list from the ‘spacy’ library. This contains 326 English stop words.



***Removing the punctuations:*** We need to take care of punctuations in the summarization process. There are a total of 32 punctuations in general that need to be removed. We used a punctuation list from the ‘string’ library.

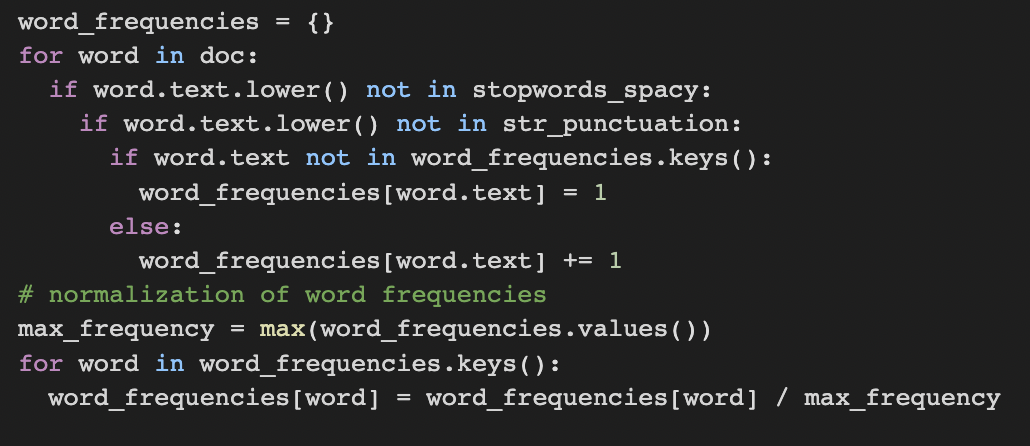
***Stemming:*** It is the process of reducing a word to its root word. For example, the words run, running, and runned are derived from the same word ‘run’. Wherever we find these words, stemming converts them to ‘run’. It is a word normalization technique. We did stemming using the snowball stemmer algorithm from Natural Language Tool Kit (NLTK) package.



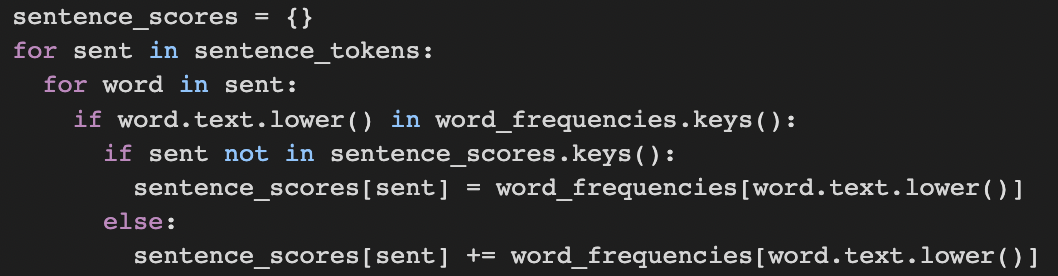
***Word Tokenization:*** It is the process of splitting large text into words. Each of these smaller units is called a token. We have used the word tokenization method of the spacy library for heap queue implementation and NLTK for the factor-based approach.

***Sentence Tokenization:*** We also have to tokenize the sentences which means we convert the paragraphs in the text to the smaller units - sentences. For this also, we have taken the advantages of spacy and NLTK by using their sentence tokenization methods.

***Calculate Weighted Word Frequency:*** After the tokenization is done, the next is to get the frequencies of each word in the text. We have taken the weighted average of the frequencies by dividing the count of occurrence of each word by the maximum frequency count and storing them in the form of key-value pair. This is a kind of normalization process.

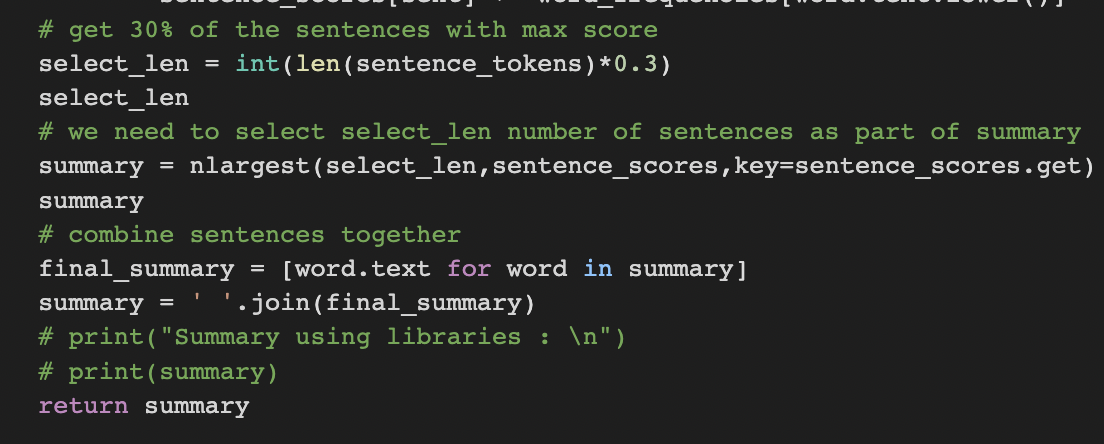


***Calculate Sentence Scores:*** It is a process to assign a numerical value to each sentence based on the priority of the algorithm used. In our project, we obtained the sentence score by adding up the frequency count of all the words of the sentence and we stored them as a key-value pair.

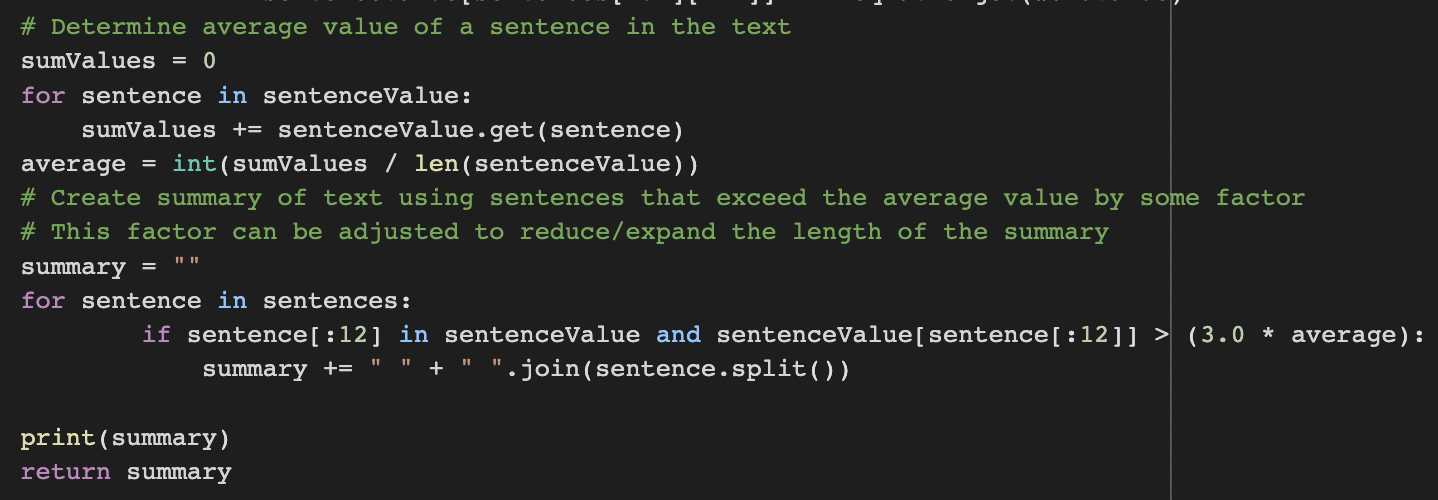


## **Modeling**

We have provided two approaches to modeling. One is using the heap queue algorithm from the nlargest library and the other is considered a factor-based approach. In the heap queue algorithm, we are taking 30% (n) largest word tokens from the sentence tokens and joining the sentences obtained into a list to form the final summary. The code snippet is shown below.



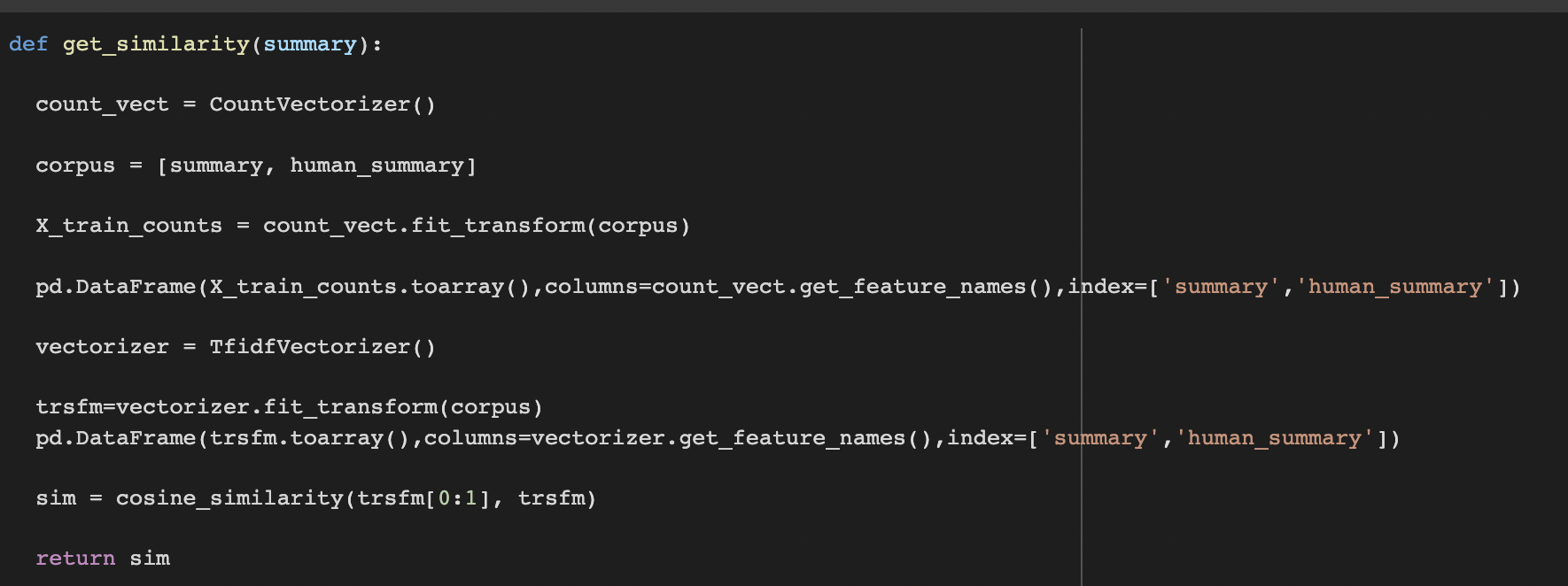
In the other approach, we take the average of the sentence scores calculated in the preprocessing step. We take the first 12 words of each sentence and if it is three times the average, then we consider that sentence to be part of the summary. The code snippet is shown below.

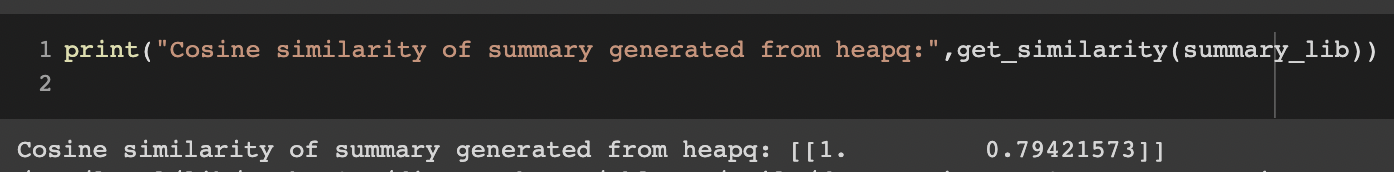


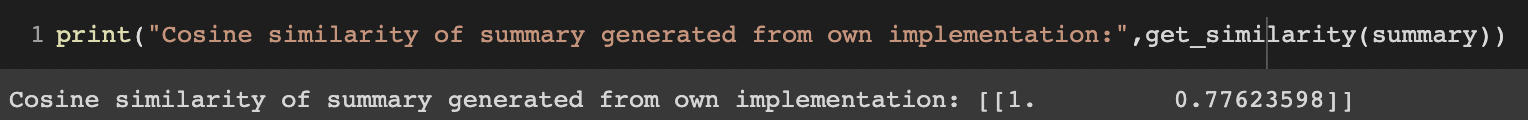
# **Experiments and Results**

After we build the two models, we have evaluated the performance of those models by using intrinsic measures that compare the output summary of each model with the human-generated summary. Two evaluation metrics: cosine similarity and Rouge library which gives us the f1 score, precision, and recall are used.

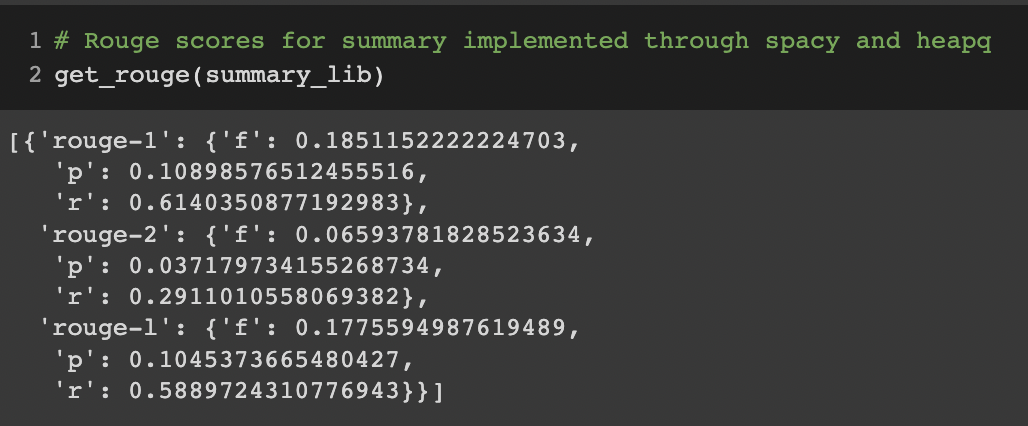
Cosine similarity is a good measure for document comparison as it tells the similarity between the documents by considering the angle between the input vectors (which are word frequencies in our case) without considering the size of the documents. We got the cosine similarity score for the heap queue implementation as 79.4% whereas, for the factor-based approach, it is 77.6% for the documents that we have taken.

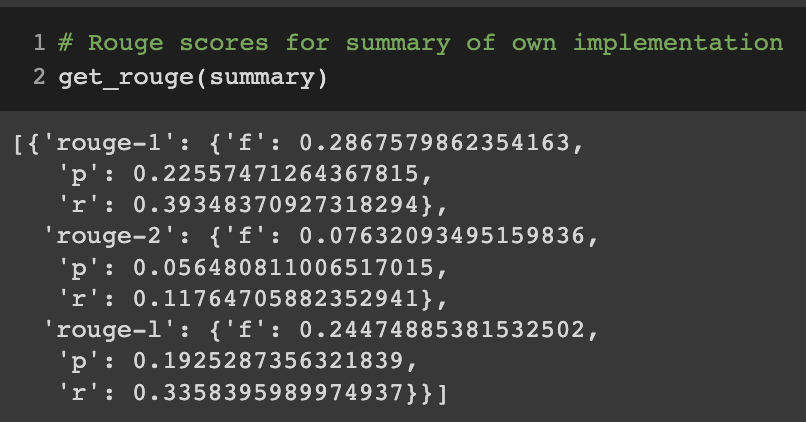






ROUGE stands for Recall-Oriented Understudy for Gisting Evaluation. It is a well-known measure for the text summarization task. It takes into account the overlapping words from each document and the human-generated summary and calculates the f1, precision, and recall. ROUGE-N measures unigram, bigram, trigram,and higher-order n-gram overlap. In our project, we got the ROUGE-1 which means unigram, and ROUGE-2 which means bigram scores. We also have the ROUGE-L that measures the longest matching sequence of words.





However, we need to also consider the readability of the summary generated. From our results, we found that the summary generated using the heap queue algorithm is not so readable when compared to that of the latter implementation.

**Discussion and Future Improvements**

Limitations:

* Implemented only using the pdf and text documents.
* Used only intrinsic measures for evaluation. We need to consider extrinsic evaluations as well which aim at evaluating the system’s output based on its impact on other NLP tasks.
* Applicable to only the English language. More research needs to be done on the other language areas.

Future Improvements:

* Need to improve the readability of the text and also the similarity scores. This can be done by performing more cleaning. For example, removing extra spaces, removing HTML tags, words containing digits, spell and grammar checking, etc.
* Can also implement using deep learning models like Sequence-to-Sequence (Seq2Seq) which gives more accurate results.