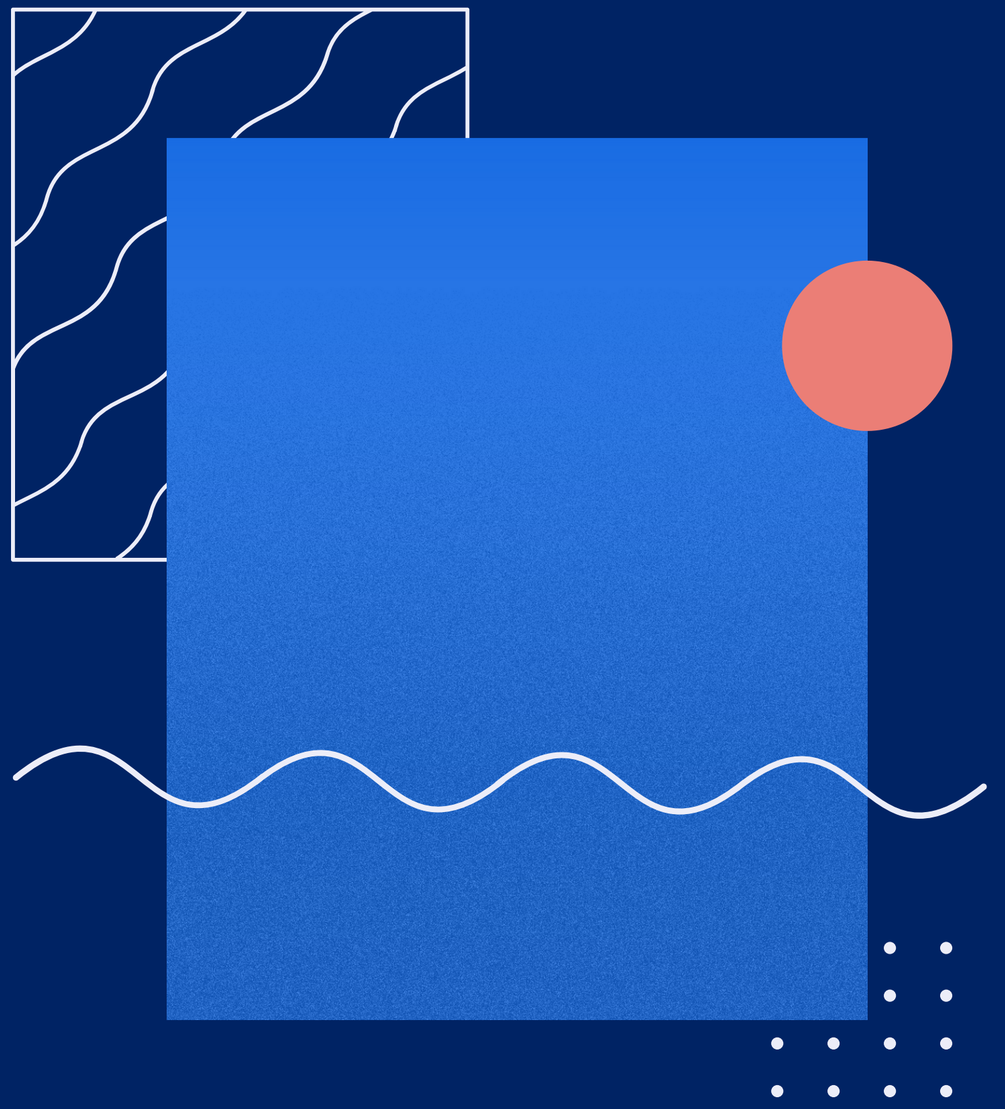
TEXT SUMMARIZATION

TO SUMMARIZE THE SUMMARY OF THE SUMMARY!

TUSHAR DESHMUKH | SHAURYA SOOD | RITTIK BASU



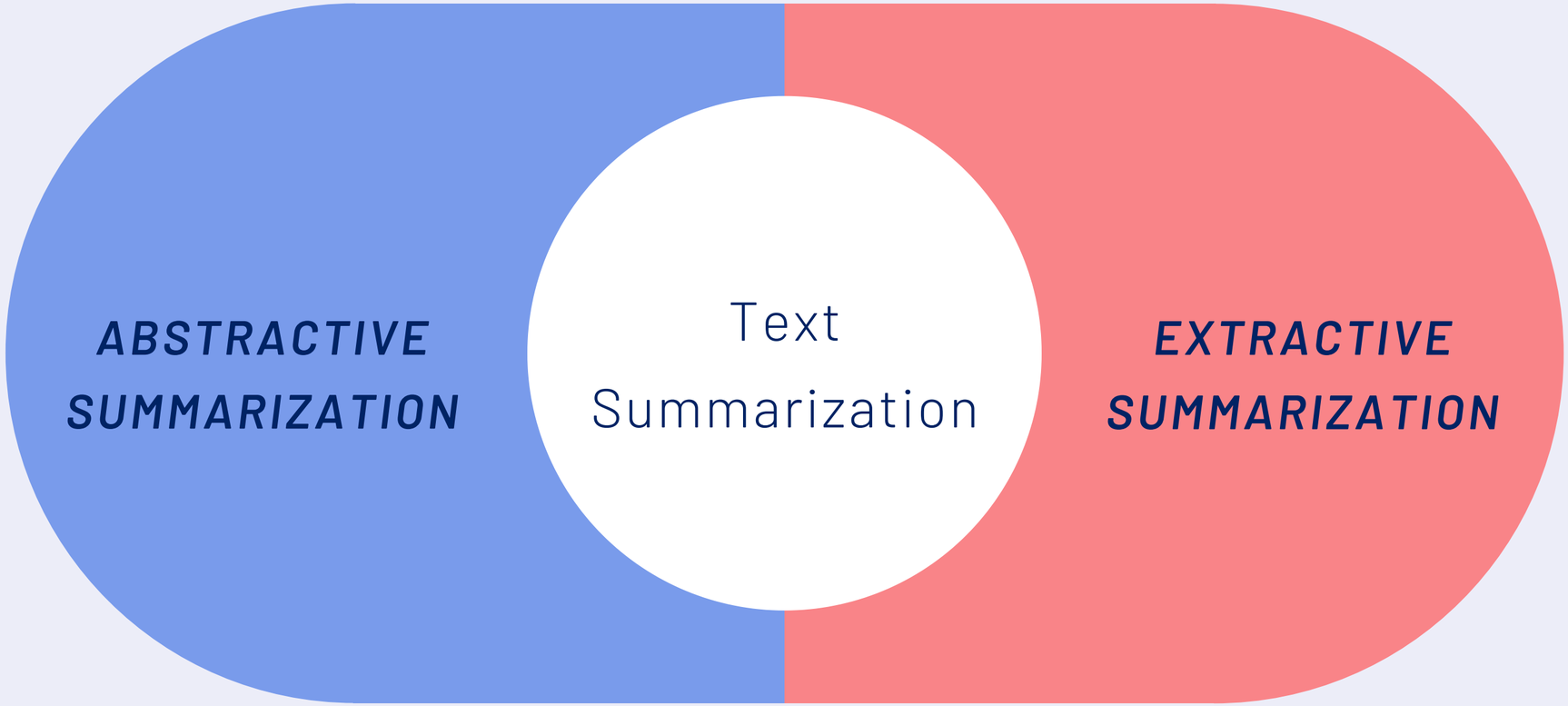
WHAT IS TEXT SUMMARIZATION?

THE CHALLENGE

Summarisation is very challenging because when we as humans summarize a piece of text, we usually read it entirely to develop our understanding, and then write a summary highlighting its main points. Since computers lack human knowledge and language capability, it makes automatic text summarization a very difficult and non-trivial task.

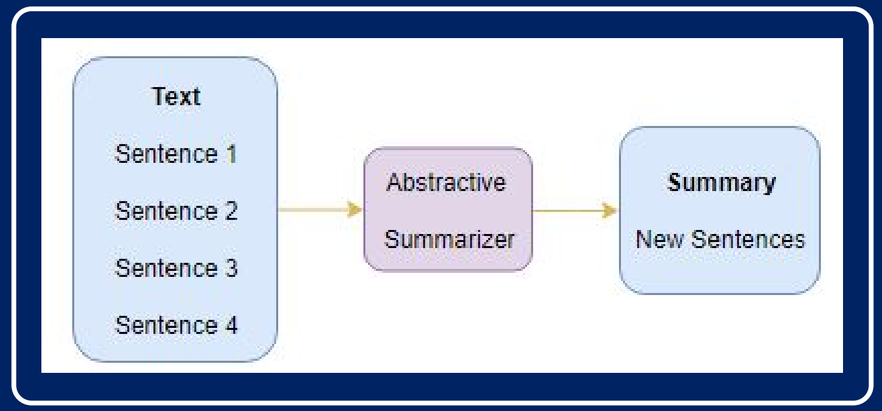
Various models based on machine learning have been proposed for this task. Most of these approaches model this problem as a classification problem which outputs whether to include a sentence in the summary or not. Other approaches have used topic information, Latent Semantic Analysis (LSA), Sequence to Sequence models, Reinforcement Learning and Adversarial processes.

TYPES OF SUMMARIZATION

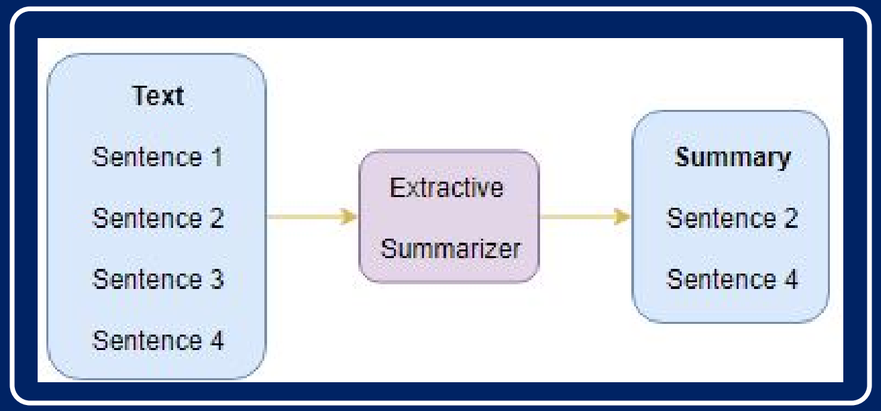


ABSTRACTIVE VS EXTRACTIVE

This is a very interesting approach. Here, we generate new sentences from the original text. This contrasts with the extractive approach we saw earlier where we used only the sentences that were present. The sentences generated through abstractive summarization might not be present in the original text:



The name gives away what this approach does. We identify the important sentences or phrases from the original text and extract only those from the text. Those extracted sentences would be our summary. The below diagram illustrates extractive summarization:



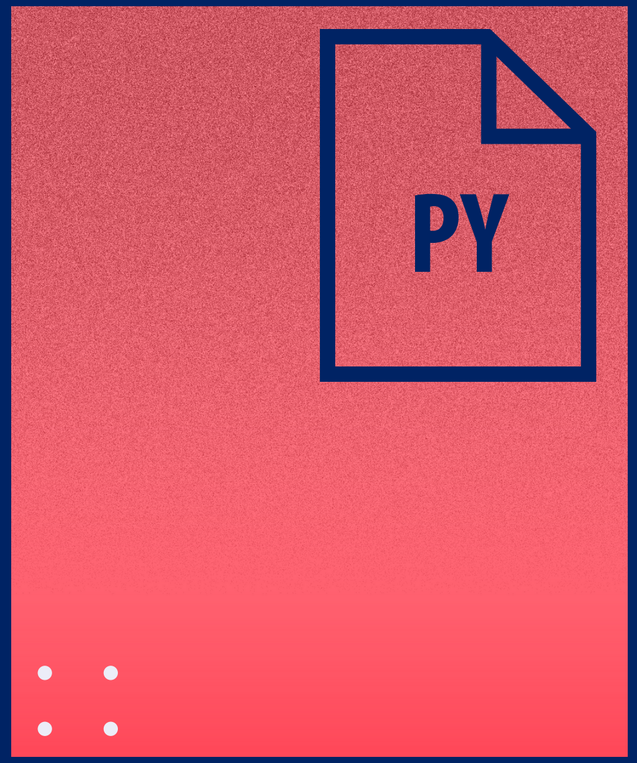
Abstractive Text Summarizer using Deep Learning requires Sequence-to-Sequence which consists of encoders and decoders and its implementations require using kera’s which is a NNL written in python

We will be implementing the basics of text summarization by implementing the extraction summarization.

Steps:

1. Convert the paragraph into sentences.
2. Text Processing
3. Tokenization and Preprocessing
4. Evaluate the weighted occurrence frequency of the words:
5. Substitute words with their weighted frequencies

LANGUAGE & ENVIRONMENT



We have tried to use python and have python3 packages installed and therefore we are using Jupyter notebook as an IDE

For this project, you need to have the following packages installed in your python.

- **bs4 — used for parsing Html page.**
- **lxml — it is the package that is used to process Html and XML with python.**
- **nltk — for performing natural language processing tasks.**
- **urllib — for requesting a webpage.**

import bs4
import urllib.request as url
import re
import nltk
nltk.download('punkt') {//Used for tokenization//}
nltk.download('stopwords') {//Used to remove stop words//}

1.url_name = input("Enter url:") {//This is used to input text directly from webpage//}

2.We have requested the page source with urllib and then parse that page with BeautifulSoup

bs4.BeautifulSoup(web,'html.parser')

3.Now we remove all the special characters from that string variable articlethat contains the whole article that is to be summarized.

processed = article.replace(r'^\s+|\s+?$','')

For this, we have simply used inbuilt replacefunction

processed = processed.replace("\","")

We have simply used the sent_tokenize function of nltk to make the list that contains sentences of the article

```
sentences = sent_tokenize(processed)
```

After that, we convert the characters of article to lowercase. Then we loop through every word of the article and check if it is not stop word or any punctuation

```
processed1 = processed.lower()
```

And if the word is none of them, we just added that word into the dictionary and then further count the frequency of that word.
You can see the dictionary containing every word with its count in the article(higher the frequency of the word, more important it is). Now you know why we have removed stopwords like “of, the, for” otherwise, they will come on top.

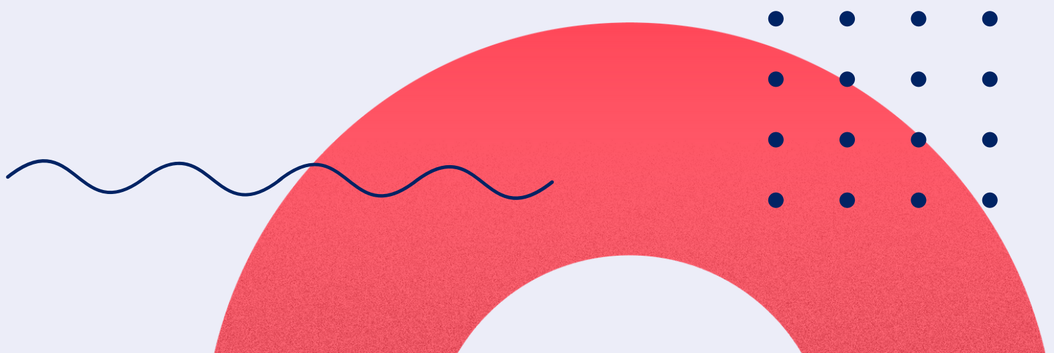
```
frequency = {}

for  word in word_tokenize (processed1):
        if  word not in stop_word and word not in string.punctuation:
                if word not in frequency.keys():
                        frequency[word]=1
                else:
                        frequency[word]+=1
```

frequency

<We have calculated the importance of every word in the dictionary by simply dividing the frequency of every word with the maximum frequency among them.>

```
max_fre = max(frequency.values())
        for word in frequency.keys():
                frequency[word]=(frequency[word]/max_fre)
```

frequency



DEMO

THANK YOU!

SUMMARIZING THE SUMMARY OF THE SUMMARY!

Tushar Deshmukh | Shaurya Sood | Rittik Basu