MKSSS's Cummins COLLEGE OF ENGINEERING FOR WOMEN

MINI PROJECT REPORT

ON

" Text Summarizer "

Submitted By

C No: C22020441611 Sakshi Bharambe (3611)

C No: C22020441612 Sakshi Bhosale (3612)

DEPARTMENT OF INFORMATION TECHNOLOGY

MKSSS's CUMMINS COLLEGE OF ENGINEERING FOR WOMEN,

PUNE

2022 – 2023

INDEX

NATURAL LANGUAGE PROCESSING

TEXT SUMMARIZER

Summarization

Summarization is a technique to shorten long texts such that the summary has all the important points of the actual document. It is a natural language processing task or application.

Approaches to Automatic summarization
There are mainly two types of summarization:

1. Extraction-based Summarization:
   The extractive approach involves picking up the most important phrases and lines from the documents. It then combines all the important lines to create the summary.
   So, in this case, every line and word of the summary actually belongs to the original document which is summarized.

2. Abstraction-based Summarization:
   The abstractive approach involves summarization based on deep learning. So, it uses new phrases and terms, different from the actual document, keeping the points the same, just like how we actually summarize. So, it is much harder than the extractive approach.

We have taken an extraction based approach to implement a text summarizer web application.

1. It selects the most important sentences from a text to create a summary.
2. It involves using algorithms to identify important sentences based on factors like word frequency.
3. Summary may lack coherence or does not follow the flow of content well.
4. Commonly used in news aggregation, academic contexts and news articles summary.

Features of extractive summarizer web application:

i. Text to summary
ii. Image (.png, .jpg, .jpeg) to summary
iii. Text file (.txt) to summary
iv. Percentage summary

1. Text to summary

   Here we have used spaCy, a free and open-source library with a lot of built-in capabilities for processing and analyzing data in the field of NLP and default model for the English language is en_core_web_sm is used.

2. Image (.png, .jpg, .jpeg) to summary

   An image can be uploaded from local storage through UI.
   Pytesseract Python library is used for processing the image contents in which Pytesseract.image_to_string function is available which extracts the text from image.
   Image should have a fair amount of text in order to get detected while converting to text.

3. Text file (.txt) to summary

   Here the file is uploaded from local storage and stored in the main.py file directory.
   Then text is extracted from the file and a summarization function is applied on it.

4. Technology For Frontend:

   HTML, CSS are used to develop the user interface of the web application of text summarizer.

5. Technology For Backend:

   Flask framework is used to connect logic implemented in python language and UI of website .

6. Library/ Framework used:

   Flask, spacy, pathlib, tesseract.

Algorithm for summarization function:

1. Text input is taken, if it is file or image then text is extracted from it.
2. The words which are not punctuation and stop words are converted to their root form and stored in a new list called word tokens. This is called lemmatization.
3. Word frequency of the words in the list is counted.
4. From word frequency maximum frequency is taken and all word frequencies are normalized.
5. Then the sentence score is counted by adding all words frequency for the words in that sentence.
6. And according to the required length of the summary, top sentences by score are taken into account to form a summary.

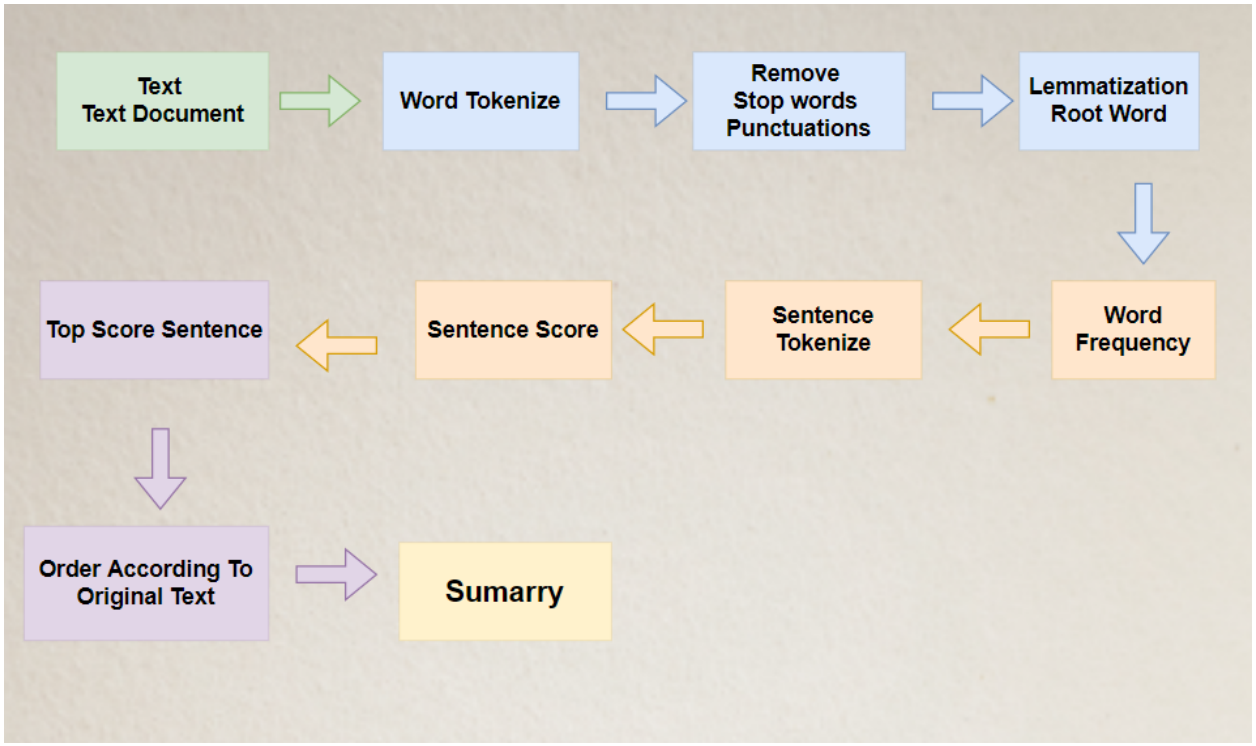
Applications:
For newspaper summary
For generating summary of long report
For generating summary of story book

Code of summarization function:

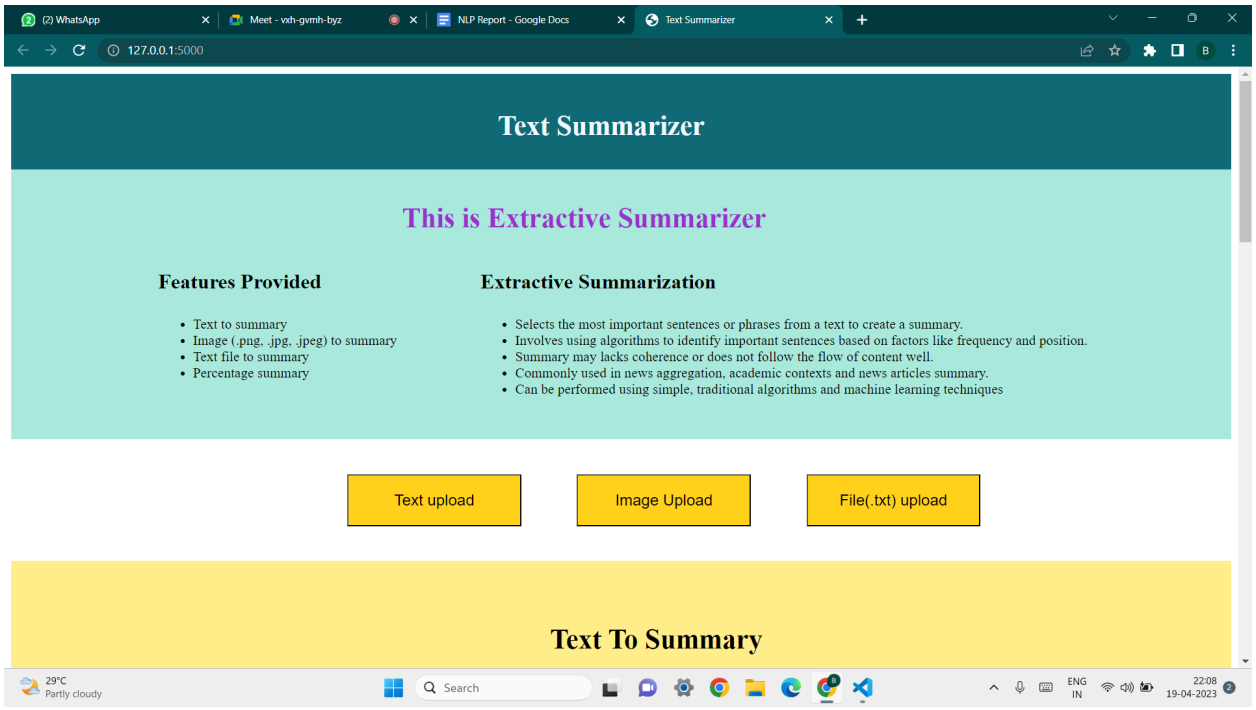
```
import spacy
import pathlib

def summary(choice,text,percentage1):
 nlp=spacy.load('en_core_web_sm')
word_Tokens=[]
  for word in doc:
    if word.text.lower() not in stopwords:
      if word.text.lower() not in punctuation:
        w = (word.lemma_)
        word_Tokens.append(w.lower())

word_frequencies = {}
  for word in word_Tokens:
    if word.lower() not in word_frequencies.keys():
      word_frequencies[word.lower()]=1
    else:
      word_frequencies[word.lower()]+=1
```

```
for word in word_frequencies.keys():
    word_frequencies[word]=word_frequencies[word]/max_freq

  # Sentence Tokenization from given text
  sentence_tokens = [sent for sent in doc.sents]
  print(sentence_tokens)

 sentence_scores ={}
  for sent in sentence_tokens:
    for word in sent:
      if word.text.lower() in word_frequencies.keys():
        if sent not in sentence_scores.keys():
          sentence_scores[sent] = word_frequencies[word.text.lower()]
        else:
          sentence_scores[sent] += word_frequencies[word.text.lower()]
  print("sentence ",sent,"  \nscore : ",sentence_scores[sent])

 select_length = int(len(sentence_tokens)*percentage1*0.01)
  if select_length < 1 :
    select_length=1;

from heapq import nlargest
  top_sents = nlargest(select_length, sentence_scores,key = sentence_scores.get)
  cnt=0
  for sent in top_sents:
    cnt+=1
    print(cnt,":",sent,)

 summary=[]
  for sent in sentence_tokens:
    if sent in top_sents:
      summary.append(sent)

  # Creating Summary paragraph of list of top sentences in order
  final_words = [word.text for word in summary]
  final_summary = ' '.join(final_words)
  print("\nExtractive Summary of News Article\n")
  print("\n\n",final_summary,"\n\n")
  return final_summary
```

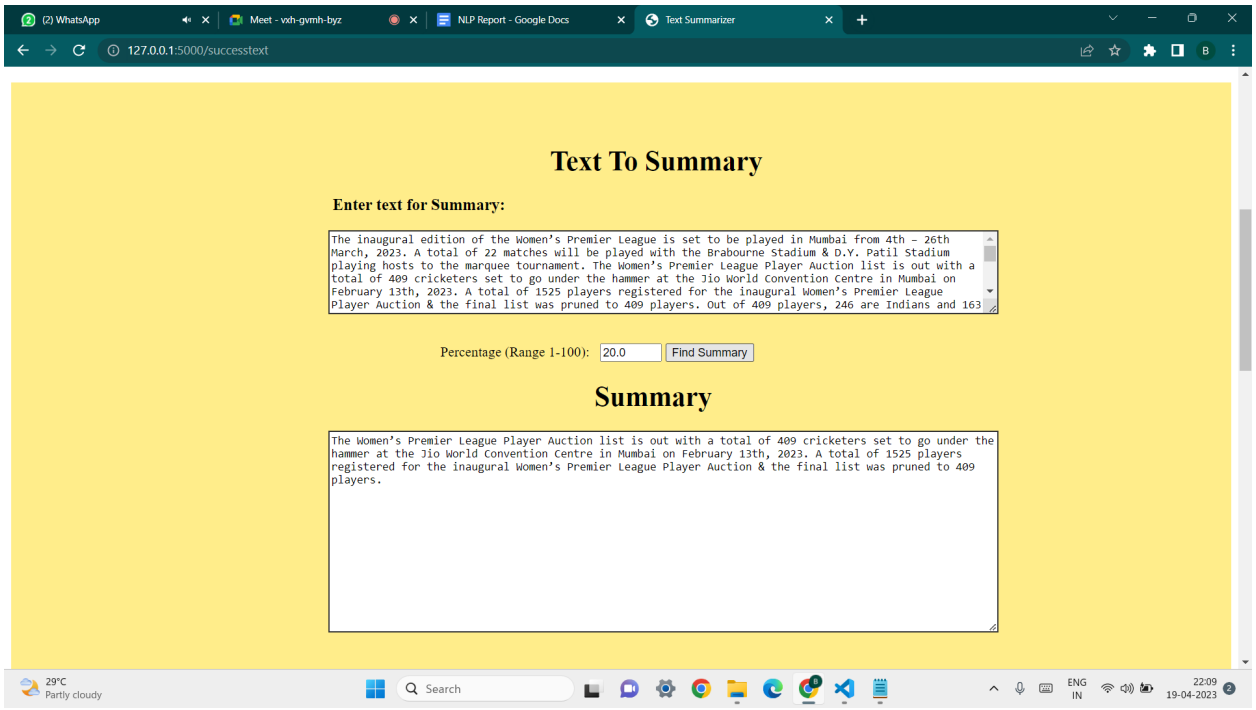
Flowchart of summarization:



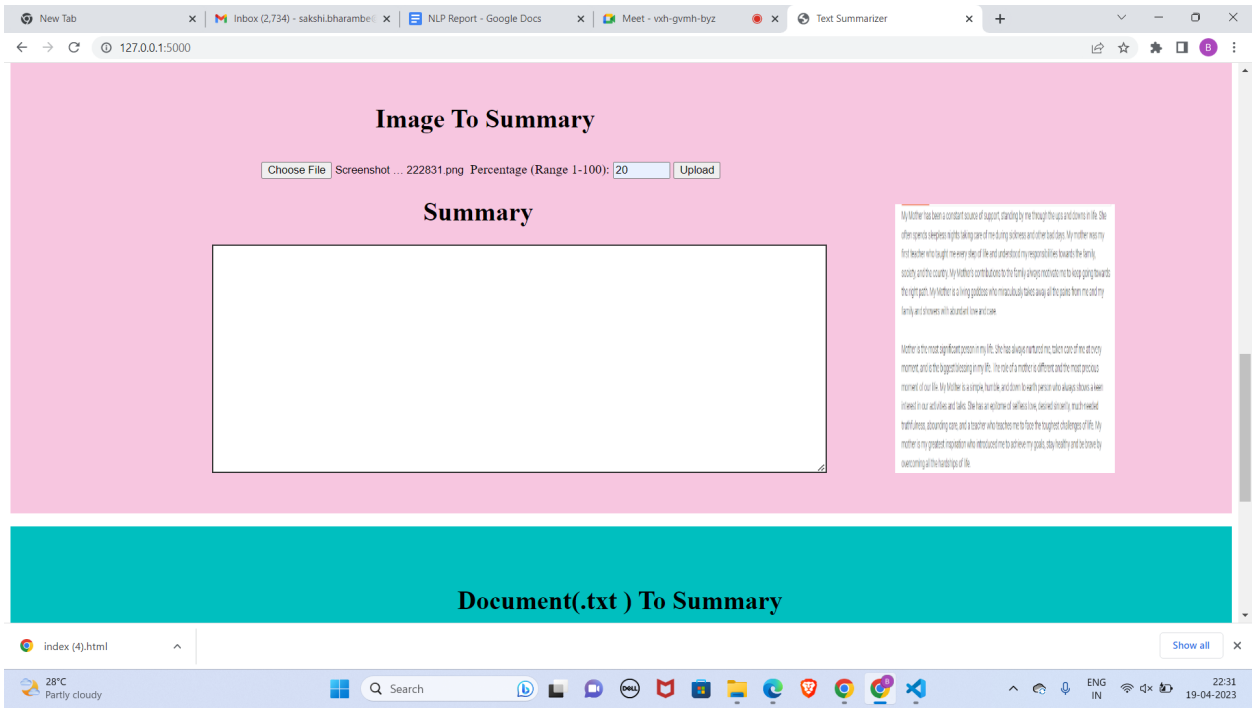
Text summarizer web application images



1)Text to summary



2)Image to summary



3).txt file to summary

.txt file content:

Extractive summarization is a technique used in natural language processing to automatically generate a summary of a given text by selecting the most relevant sentences or phrases from the original content. This approach is different from abstractive summarization, which involves generating new sentences or phrases to convey the essence of the original text.

Extractive summarization typically involves three main steps: sentence selection, sentence ranking, and summary generation. In the first step, the algorithm identifies the most important sentences in the text based on various criteria such as keyword frequency, sentence length, and semantic relevance. In the second step, the selected sentences are ranked based on their importance and relevance to the overall content. Finally, the summary is generated by selecting the top-ranked sentences and concatenating them together.

There are several benefits to using extractive summarization. Firstly, it can save time and effort by quickly generating a summary of lengthy documents, making it easier for readers to grasp the key points. Secondly, it can be useful for applications such as news aggregation, where multiple articles on the same topic can be summarized into a single, comprehensive summary. However, extractive summarization also has limitations, such as difficulty in handling texts with complex or ambiguous meanings, and the potential for important information to be overlooked if it is not included in the selected sentences.

Overall, extractive summarization is a valuable technique for automating the summarization of large volumes of text, but it should be used with caution and in combination with other natural language processing techniques to ensure the accuracy and completeness of the generated summaries.