ALGORITHM FOR WEB INFORMATION RETRIEVAL PROJECT - UE19CS332

PROJECT TITLE: **WIKIPEDIA ARTICLE SUMMARIZER**

PROJECT BATCH **– TEAM NO 15**

TEAM MEMBERS:

BHOOMIKA P BHAVIMATH – PES2UG19CS091

DEEPALI SURAJ ATTAVAR - PES2UG19CS106

K KEERTHANA - PES2UG19CS170

INTRODUCTION:

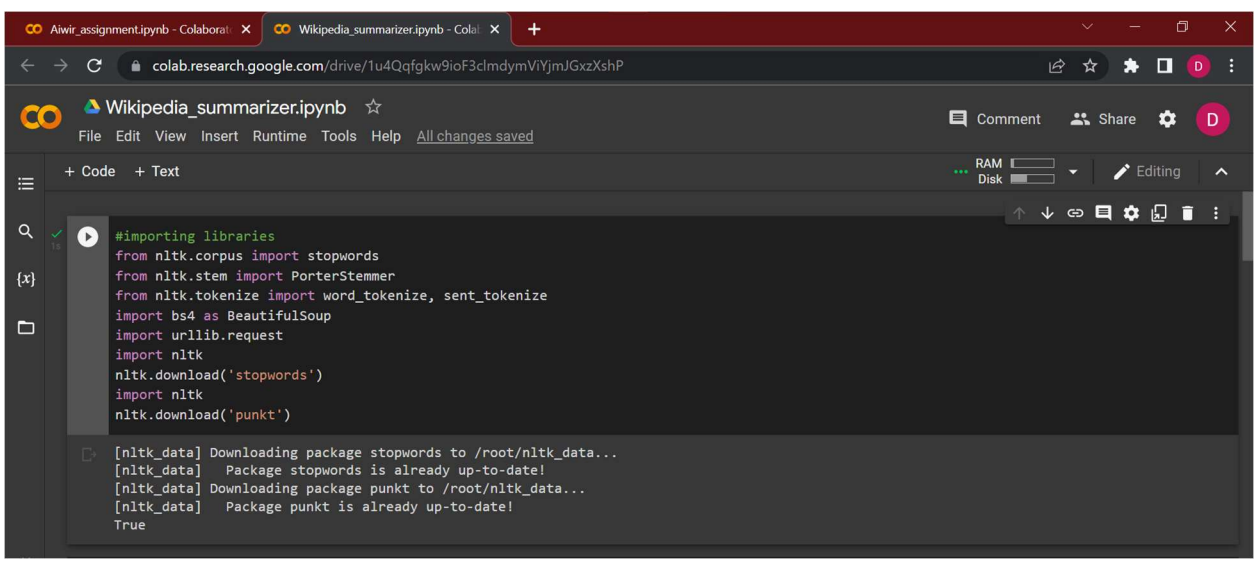
Wikipedia articles cover topics at several levels of detail:
the lead contains a quick summary of the topic's most important points, and each major subtopic is detailed in its own section of the article.

Text Summarization is one of the techniques used in NLP to create short meaningful collection of text called summaries from text resources like articles, books, research papers or even a webpage.

The main objective of a text summarization system is to identify the most important information from the given text and present it to the end users. In this project, Wikipedia articles are given as input to system and extractive text summarization is presented by identifying text features and scoring the sentences accordingly. The text is first pre-processed to tokenize the sentences and perform stemming operations. We then score the sentences using the different text features. The scores are used to classify the sentence to be in the summary text.

Our project code link:
https://colab.research.google.com/drive/1u4Qqfgkw9ioF3clmdymViYjmJGxzXshP#scrollTo=MtTNZvqPV5A8

Code and output screenshots:



Aiwir_assignment.ipynb - Colaborat | Wikipedia_summarizer.ipynb - Colal

colab.research.google.com/drive/1u4Qqfgkw9ioF3clmdymViYjmJGxzXshP

Wikipedia_summarizer.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

RAM Disk Editing

```python
def _create_dictionary_table(text_string) -> dict:

    #removing stop words
    stop_words = set(stopwords.words("english"))

    words = word_tokenize(text_string)

    #reducing words to their root form
    stem = PorterStemmer()

    #creating dictionary for the word frequency table
    frequency_table = dict()
    for wd in words:
        wd = stem.stem(wd)
        if wd in stop_words:
            continue
        if wd in frequency_table:
            frequency_table[wd] += 1
        else:
            frequency_table[wd] = 1

    return frequency_table
```

1s completed at 1:38 PM

Aiwir_assignment.ipynb - Colaborat | Wikipedia_summarizer.ipynb - Colal

colab.research.google.com/drive/1u4Qqfgkw9ioF3clmdymViYjmJGxzXshP

Wikipedia_summarizer.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

RAM Disk Editing

process to associate a numerical value with a sentence based on the used algorithm's priority

```python
def _calculate_sentence_scores(sentences, frequency_table) -> dict:

    #algorithm for scoring a sentence by its words
    sentence_weight = dict()

    for sentence in sentences:
        sentence_wordcount = (len(word_tokenize(sentence)))
        sentence_wordcount_without_stop_words = 0
        for word_weight in frequency_table:
            if word_weight in sentence.lower():
                sentence_wordcount_without_stop_words += 1
                if sentence[:7] in sentence_weight:
                    sentence_weight[sentence[:7]] += frequency_table[word_weight]
                else:
                    sentence_weight[sentence[:7]] = frequency_table[word_weight]

        sentence_weight[sentence[:7]] = sentence_weight[sentence[:7]] / sentence_wordcount_without_stop_words

    return sentence_weight
```

1s completed at 1:38 PM

Aiwir_assignment.ipynb - Colaborat | Wikipedia_summarizer.ipynb - Colal

colab.research.google.com/drive/1u4QqfgkW9ioF3clmdymViYjmJGxzXshP

Wikipedia_summarizer.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

RAM Disk Editing

[4]
```python
def _calculate_average_score(sentence_weight) -> int:

        #calculating the average score for the sentences
        sum_values = 0
        for entry in sentence_weight:
            sum_values += sentence_weight[entry]

        #getting sentence average value from source text
        average_score = (sum_values / len(sentence_weight))

        return average_score
```

```python
def _get_article_summary(sentences, sentence_weight, threshold):
        sentence_counter = 0
        article_summary = ''

        for sentence in sentences:
            if sentence[:7] in sentence_weight and sentence_weight[sentence[:7]] >= (threshold):
                article_summary += " " + sentence
                sentence_counter += 1

        return article_summary
```

1s completed at 1:38 PM

Aiwir_assignment.ipynb - Colaborat | Wikipedia_summarizer.ipynb - Colal

colab.research.google.com/drive/1u4QqfgkW9ioF3clmdymViYjmJGxzXshP

Wikipedia_summarizer.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

RAM Disk Editing

[6]
```python
def _run_article_summary(article):

        #creating a dictionary for the word frequency table
        frequency_table = _create_dictionary_table(article)

        #tokenizing the sentences
        sentences = sent_tokenize(article)

        #algorithm for scoring a sentence by its words
        sentence_scores = _calculate_sentence_scores(sentences, frequency_table)

        #getting the threshold
        threshold = _calculate_average_score(sentence_scores)

        #producing the summary
        article_summary = _get_article_summary(sentences, sentence_scores, 1.5 * threshold)

        return article_summary
```

```python
#fetching the content from the URL
fetched_data = urllib.request.urlopen('https://en.wikipedia.org/wiki/Charles_Leclerc')
article_read = fetched_data.read()
```

1s completed at 1:38 PM

Aiwir_assignment.ipynb - Colaborat | Wikipedia_summarizer.ipynb - Colab

colab.research.google.com/drive/1u4QqfgkW9ioF3clmdymViYjmJGxzXshP

Wikipedia_summarizer.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

RAM Disk Editing

[8]
```
#parsing the URL content and storing in a variable
article_parsed = BeautifulSoup.BeautifulSoup(article_read,'html.parser')
```

[9]
```
#returning <p> tags
paragraphs = article_parsed.find_all('p')
article_content = ''
```

[10]
```
#looping through the paragraphs and adding them to the variable
for p in paragraphs:
    article_content += p.text
```

```
if __name__ == '__main__':
    summary_results = _run_article_summary(article_content)
    print(summary_results)
```

```
 He would also finish fourth in the sprint race the next day, giving him a 50-point championship lead over Rowland. For Leclerc's final race
In China, Leclerc qualified fourth behind Vettel. He would finish third again in France. He ultimately finished the race in fourth place. He
```

1s completed at 1:38 PM