Minor Project

Problem Statement:-

Text Summarization: Taking a paragraph as the input and getting a one-line summary for that paragraph.

ALORITHM:-

1. Remove all special character,digit,Multipal space and square brackets from paraghaph for which we find summary using Regular expression module.

2. Remove stopwords from paragraph and Find frequency occurance of each word.

3. Find weighted frequency of each word(means simply divide the number of occurances of words by the frequency of the most occurring word(highest frequency)).

4. Calculating the score of all sentences(by adding weighted frequencies of the words that occur in that particular sentence).

5. We can take top N sentences with the highest scores which is the summary of given paragraph.

CODE EXPLANATION:-

1. Import all the library which is mention below

   import nltk
   This library import all Natural Language processing Module

   import re
   This library import all Regular Expression Module

   from nltk.tokenize import word_tokenize
   This is for word tokenization

   from nltk.tokenize import sent_tokenize
   This is for sentance tokenization

   import heapq
   This library is for retrieve top N sentance with highest scores

2.

   Taking the paragraph for which we find summary.

3. Remove all special character,digit,Multipal space and square brackets from paraghaph

   paragraph_text = re.sub(r'\[[0-9]*\]', ' ', text)
   This code removes square brackets from paragraph(which is store in variable text) and store into paragraph_text.

paragraph_text = re.sub(r'\s+', ' ', text)
This code replaces the resulting multiple spaces by a single space from paragraph(which is store in variable text) and store into paragraph_text.

formatted_paragraph_text = re.sub('[^a-zA-Z]', ' ', paragraph_text)
formatted_paragraph_text = re.sub(r'\s+', ' ', formatted_paragraph_text)
This code is used for removeing special characters and digits from paragraph.

sentence_list = sent_tokenize(paragraph_text)
This code tokenize the paragharph into list of sentences and store in sentence_list

4. Find frequency occurance of each word

we used formatted_paragraph_text to find the frequency occurrence since it doesn't contain punctuation, digits, or other special characters.

```
stopwords = nltk.corpus.stopwords.words('english')
word_frequencies = {}
for word in nltk.word_tokenize(formatted_paragraph_text):
    if word not in stopwords:
        if word not in word_frequencies.keys():
            word_frequencies[word] = 1
        else:
            word_frequencies[word] += 1
```

In the code above, we first store all the English stopwords from the nltk library into a stopwords variable. Next, we loop through all the sentences and then corresponding words to first check if they are stopwords. If not, we proceed to check whether the words exist in word_frequency dictionary i.e. word_frequencies , or not. If the word is encountered for the first time, it is added to the dictionary as a key and its value is set to 1. Otherwise, if the word previously exists in the dictionary, its value is simply updated by 1.

5. Find weighted frequency of each word

weighted frequency:- means simply divide the number of occurances of words by the frequency of the most occurring word(highest frequency).

```
maximum_frequncy = max(word_frequencies.values())
for word in word_frequencies.keys():
    word_frequencies[word] = (word_frequencies[word]/maximum_frequncy)
```

This code calculate the weighted frequency of all word and store into word_frequencies.

6. Calculating the score of all sentences

To calculate the scores for each sentence by adding weighted frequencies of the words that occur in that particular sentence.

```
sentence_scores = {}
for sent in sentence_list:
```

```
for word in word_tokenize(sent.lower()):
    if word in word_frequencies.keys():
      if len(sent.split(' ')) < 50:
        if sent not in sentence_scores.keys():
            sentence_scores[sent] = word_frequencies[word]
        else:
            sentence_scores[sent] += word_frequencies[word]
```

In the code above, we first create an empty sentence_scores dictionary. The keys of this dictionary will be the sentences themselves and the values will be the corresponding scores of the sentences. Next, we loop through each sentence in the sentence_list and tokenize the sentence into words.
We then check if the word exists in the word_frequencies dictionary. This check is performed since we created the sentence_list list from the paragraph_text object. on the other hand, the word frequencies were calculated using the formatted_paragraph_text object, which doesn't contain any stop words, numbers,etc.

We do not want very long sentences in the summary, therefore, we calculate the score for only sentences with less than 50 words (although you can tweak this parameter for your own use-case). Next, we check whether the sentence exists in the sentence_scores dictionary or not. If the sentence doesn't exist, we add it to the sentence_scores dictionary as a key and assign it the weighted frequency of the first word in the sentence, as its value. On the contrary, if the sentence exists in the dictionary, we simply add the weighted frequency of the word to the existing value.

7. Getting the Summary

```
summary_sentences = heapq.nlargest(1, sentence_scores, key=sentence_scores.get)
summary = ' '.join(summary_sentences)
print(summary)
```

Now we have the sentence_scores dictionary that contains sentences with their corresponding score. To summarize the paragraph, we can take top N sentences with the highest scores. The following code retrieves top 1 sentences and prints them on the screen. For this we use heapq library.
If we take 3-4 line then we get meaningful summary.I took 1 line as summary because in problem statement it was given as 1.

Prepared by:-
Name:-Rahetul Asquin
E-Mail:-rahetulasquin12345@gmail.com