NLP : TEXT SUMMARIZATION

Natural Language Processing, in short NLP, is subfield of Machine learning / AI which deals with linguistics and human languages. NLP deals with interactions between computers and human languages. In other words, it enables and programs computers to understand human languages and process & analyse large amount of natural language data.

PREPROCESSING STEPS

REMOVAL OF HTML TEXT:

HTML tags don't add much value towards understanding and analysing text so they should be removed.

This can be done by importing the Beautiful Soup library. This module provides get_text() function that takes HTML as input and returns text as output.

REMOVAL OF ACCENTED (NON ASCII) CHARACTERS:

Accented characters are important elements which are used to signify emphasis on a particular word during pronunciation or understanding. we need to make sure that these characters are converted and standardized into ASCII characters. A simple example — converting é to e.

We can remove accents from the string by using a Python module called Unidecode. This module consists of a method that takes a Unicode object or string and returns a string without ascents.

Syntax: output_string = unidecode.unidecode(target_string)

REMOVAL OF URLS:

URLs (or Uniform Resource Locators) in a text are references to a location on the web, but do not provide any additional information. We take our sample text and analyse each word, removing words or strings starting with http.

Shouldn't be too hard to write such a predicate, perhaps something as simple as

str(word).startswith('http') would suffice.

TOKENIZATION:

This involves Splitting the sentence into words.

For eg:

```
from nltk.tokenize import word_tokenize

sentence = "Books are on the table"

words = word_tokenize(sentence)

print(words)
```

REMOVAL OF STOPWORDS:

Apart from URLs, HTML tags and special characters, there are words that are not required for tasks such as sentiment analysis or text classification. Words like I, me, you, he and others increase the size of text data but don't improve results dramatically and thus it is a good idea to remove those.

For the task, we can use pre-defined stopwords collection (e.g. from NLTK or any other NLP library), or we can define our own set of stopwords based on our task.

LOWERCASING:

Converting a word to lower case

This can be done through inbuilt lower() function

Syntax: string .lower()

EXPANSION OF CONTRACTED WORDS :

In our everyday verbal and written communication, a lot of us tend to contract common words like "you are" becomes "you're". Converting contractions into their natural form will bring more insights. This can be done through 'import re'

```
Eg:  phrase = re.sub(r"can\'t", "can not", phrase)
     phrase = re.sub(r"won\’t", "will not", phrase)
     phrase = re.sub(r"can\’t", "can not", phrase)
```

LEMMATIZATION :

It is a process of transforming a word to its root form.

```python
import nltk
from nltk.stem import WordNetLemmatizer

lemmatizer = WordNetLemmatizer()

print(lemmatizer.lemmatize("Machine", pos='n'))
# pos: parts of speech tag, verb
print(lemmatizer.lemmatize("caring", pos='v'))
```

lemmatizer.py hosted with ❤ by GitHub view raw

Output: machine, care
Explanation: The word Machine transforms to lowercase and retains the same word unlike Stemming. Also, the word caring is transformed to its lemma 'care' as the parts of speech variable (pos) is verb(v)

METHODS TO ANALYZE AND SUMMARIZE

Text summarization methods can be grouped into two main categories: **Extractive** and **Abstractive methods**

EXTRACTIVE TEXT SUMMARIZATION

It is the traditional method developed first. The main objective is to identify the significant sentences of the text and add them to the summary. You need to note that the summary obtained contains exact sentences from the original text.

ABSTRACTIVE TEXT SUMMARIZATION

It is a more advanced method. The approach is to identify the important sections, interpret the context and reproduce in a new way. Note that here, the sentences in summary are generated, not just extracted from original text.

TEXT RANK ALGORITHM

Implementing the TextRank algorithm
Required Libraries

- **Numby**
- **Pandas**
- **Ntlk**
- **Re**

The following is an explanation of the code behind the extraction summarization technique:

Step 1

Concatenate all the text you have in the source document as one solid block of text. The reason to do that is to provide conditions so that we can execute step 2 more easily.

Step 2

We provide conditions that define a sentence such as looking for punctuation marks such as period (.), question mark (?), and an exclamation mark (!). Once we have this definition, we simply split the text document into sentences.

Step 3

Now that we have access to separate sentences, we find vector representations (word embeddings) of each of those sentences. (Converting sentences to numbers is known as vector representations)

Step 4

Once we have the vector representation for our words, we have to extend the process to represent entire sentences as vectors. To do so, we may fetch the vector representations of the terms that constitute words in a sentence and then the mean/average of those vectors to arrive at a consolidated vector for the sentence.

Step 5

At this point, we have a vector representation for each individual sentence. It is now helpful to quantify similarities between the sentences using the cosine similarity approach.

Cosine similarity = dot product of two vectors/cross product of two vectors

We can then populate an empty matrix with the cosine similarities of the sentences.

Step 6

Now that we have a matrix populated with the cosine similarities between the sentences.

We can convert this matrix into a graph wherein the nodes represent the sentences, and the edges represent the similarity between the sentences.

It is on this graph that we will use the handy PageRank algorithm to arrive at the sentence ranking.

PageRank Algorithm

This algorithm helps search engines like google rank web pages. Let's understand the algorithm with an example. Assume you have four web pages with different levels of connectivity between them. One may have no links to the other three; one may be connected to the other 2, one may be correlated to just one, and so on.

We can then model the probabilities of navigating from one page to another by using a matrix with n rows and columns, where n is the number of web pages. Each element within the matrix will represent the probability of transitioning from one webpage to another. By assigning the right probabilities, one can iteratively update such a matrix to come to a web page ranking.

Step 7

We now have ranked all sentences in the article in order of importance. We can now extract the top N (say 10) sentences to create a summary

CONCLUSION AND REASON FOR APPROACH

TextRank is very easy to use and implement because it's unsupervised.

TextRank can be used to find the most relevant sentences and also keywords

Hence I have opted for Text Rank algorithm for text summarization.