NLP PROJECT

TEXT SUMMARIZER

Final Review

By
Group - 07

Group Number 07

SAAM PRASANTH DEEVEN PEDAPALLI
(S20190010136 - saamprasanthdeevan.p19@iiits.in)

PIDAKALA ABHINANDAN BABU
(S20190010140 - abhinandanbabu.p19@iiits.in)

PREMA VIGNESH
(S20190010143 - prema.g19@iiits.in)

SRI LAKSHMI PRASANNA KONERU
(S20190010168 - srilakshmiprasanna.k19@iiits.in)

Motivation

- NLP text summarizer is one of the cool applications of NLP (Natural Language Processing), which shortens the long document into a shorter one while retaining all important information from the document.
- Nowadays, Short news is popular everywhere because people can get the important information in very short time, hence this saves time for the people.
- Abstractive summarization methods aim at producing summary by interpreting the text using advanced natural language techniques in order to generate a new shorter text.

Our work plan for 1st Review

For Presentation 1 on August 31st (**COMPLETED**)

- Proposal for the project among group.
- Collecting relevant information about the proposed project.
- Presenting entire project idea as a first review. In this every one of our team is Involved.
- We should learn new topics that are relevant for this project.

Our work plan for 2nd Review

For **Presentation 2 (Completed)**

- We will submit a midterm report for our project with relevant documents.
- We will get a clear Information about project and number of ways to Implement this.
- We will choose the best method for summarisation and Implement it as a trail run.

Contribution

Sri Lakshmi Prasanna Koneru

- Information collection
- Code for Implementing Word Tokenization
- Code for Implementing Sentence Tokenization.
- Code for generating Text summary.

Saam Prasanth Deeven Pedapalli

- Information collection
- Code for Implementing Absolute Word Frequency
- Code for Implementing Normalized frequencies.
- Code for generating text summary.

Abhinandhan babu

- Information collection
- Creating Stopwords.
- Code for length of text and summary.
- Updating codes and Test Cases

Prema Vignesh

- Information collection
- Implementing Sentence Scores.
- Test Cases

Information relevant to the project collected by Sri Lakshmi Prasanna , Saam Prasanth Deeven, Abhinandhan Babu and Vignesh.

- There are many ways to implement text summarization.
- Few approaches for this task are as follows.
- Text Summarization using Bert’s Google model.
- Text summarization using T5 transformer model.
- Text Summarisation in nlp using nltk library in Python.
- Text Summarization in NLP using spaCy.

- We have decided to Implement Text Summarisation by our own process using python and without using any libraries.

Proposed Approach

Text Summarisation “with” using NLP techniques and “without” using any Machine learning and Deep learning Libraries.

Working Architecture of our project:

1. Giving Text Input.
2. Creating Stopwords.
3. Tokenizing the words.
4. Find absolute frequency of occurrence.
5. Calculating Normalized frequencies
6. Tokenizing the Sentence.
7. Sentence Score.
8. Summarizing the Text.

Giving Text Input

- In our project the user can enter the data in text format without any limitations like number of words or sentences.

```
# Entering the Text
text = input("Enter your text : ")
```

Creating Stopwords

- Stop words are function (filler) words, which are words with little or no meaning that help form a sentence.
- These words do not have much role in retrieving information or getting to know which sentence is more important.
- We ignore these stopwords so that we can get to know how important a sentence really is.

```
# Creating a list for the Stopwords
stopwords1 = ['anything', 'upon', 'whereafter'
stopwords1
```

```
['anything',
 'upon',
 'whereafter',
```

Tokenization of words

- Tokenization is essentially splitting a phrase, sentence, paragraph, or an entire text document into smaller units, such as individual words or terms.
- Each of these smaller units are called tokens.
- We have used the Split function for this process.
- In this process we are replacing the punctuation marks with a space in a split function, so that each sentence/ paragraphs/entire document splits into single words/Tokens.
- These tokens help in understanding the context or developing the model for the NLP.
- The tokenization helps in interpreting the meaning of the text by analyzing the sequence of the words.

Tokenization can be done to either separate words or sentences.

```
text28 = text27.lower()
data = text28.split()
print(data)
```

```
['i', 'have', 'lots', 'of', 'friends', 'from',
```

Finding Absolute frequency of occurrence

- Creating a dictionary for the word frequencies.
- Taking the data from the text, we calculate the term frequency of words.
- we will not calculate the term frequencies for the Stopwords and Punctuation marks.
- If any word is introduced for the first time then the frequency of that word will be 1.
- If the word is being occured for the second time then it will be incremented by one.

```
print("Frequency for each word is:", word_frequencies)
```

```
Frequency for each word is: {'lots': 1, 'friends': 3, 'childhood': 1,
```

Calculating Normalized Frequency

(i) Maximum Frequency

- Obtaining the word with highest frequency from the word Frequency.

```
print("The maximium frequency is:", max_frequency)
```

```
The maximium frequency is: 5
```

```
print(word_frequencies)
```

```
{'lots': 0.2, 'friends': 0.6, 'childhood': 0.2,
```

(ii) Normalized frequency

We will divide each word frequency with maximum frequency, this is to bring the normalized frequency.

Sentence Tokenization

- Sentence tokenization is the process of splitting text into individual sentences.
- These tokenizers work by separating the words using punctuation and spaces.
- These tokens help in understanding the context or developing the model for the NLP.
- The tokenization helps in interpreting the meaning of the text by analyzing the sequence of the words.

```
# Sentence tokenisation
sentence_tokens = [sent for sent in text.split(".")]
print(sentence_tokens)
if len(sentence_tokens) >1:
  first_sent=sentence_tokens[0]
  if sentence_tokens[-1] == '':
    last_sent=sentence_tokens[-2]
  else:
    last_sent=sentence_tokens[-1]
  print(first_sent)
  print(last_sent)
else:
  print("There is only one sentence/word in the text input given")
```

Sentence Score

- We calculated the sentence scores by adding up the normalized word frequencies of the words present in the sentence after ignoring the stopwords.

```
# Printing the sentence scores
print(sentence_scores)
len(sentence_scores)
```

```
s my best friend forever': 1.9999999999999998,
```

Summarization

- We are generating the summary by initializing it with the first sentence.
- We are adding the sentences with high sentence scores to the summary and finally adding the last sentence to the summary.

```
if summary_lines > len(sentence_scores):
  print("Summary lines cannot be greater than the lines
  summary = ""
elif summary_lines < 0:
  summary = ""
  print("Invalid summary lines")
elif summary_lines == 0:
  summary = ""
elif summary_lines == 1:
  summary = first_sent
else:
  summary_lines = summary_lines-2
  summary = first_sent
  key_list = list(sentence_scores.keys())
  val_list = list(sentence_scores.values())
  del key_list[0]
  del val_list[0]
  del key_list[-1]
  del val_list[-1]
  while summary_lines is not 0:
    position = val_list.index(max(val_list))
    summary += ". " + key_list[position]
    del val_list[position]
    del key_list[position]
    summary_lines -= 1
  summary = summary + ". " + last_sent
  print(summary)
```

Output

Here there are 1629 total words in a Document, after the summarisation process the length of the Summary is 403 words. The Length of the summarization can be changed by changing the summary lines required.

```
print("Length of Original text :",len(text))
print("Length of Summary  :",len(summary))
print("_"*215)
print("\t\t\t\t\t\t SUMMARY GENERATED")
print("_"*215)
print(summary)
```

```
Length of Original text : 1629
Length of Summary  : 403
```

Conclusion

Text Summarization Saves Time:

By generating automatic summaries, text summarization helps content editors save time and effort, which otherwise is invested in creating summaries of articles manually.

Text Summarization gives Instant Response:

It reduces the user's effort involved in extracting the relevant information. With automatic text summarization, the user can summarise an article in just a few seconds by using the software, thereby decreasing their reading time.

Future work:

We will implement Abstractive Summarisation and update code. We will also look to update the code.

Thank you