ELC Activity-4 ‘22

Submitted for ELC-4, 2022 By:
102003088 Kunal Demla
102003087 Gaurav Pahwa

Text Summarization

Automatic summarization is the process of shortening a text document with software, in order to create a summary with the major points of the original document. Technologies that can make a coherent summary take into account variables such as length, writing style and syntax.

Automatic data summarization is part of real machine learning and data mining. The main idea of summarization is to find a subset of data which contains the "information" of the entire set. Such techniques are widely used in industry today. Search engines are an example; others include summarization of documents, image collections and videos. Document summarization tries to create a representative summary or abstract of the entire document, by finding the most informative sentences, while in image summarization the system finds the most representative and important (i.e. salient) images. For surveillance videos, one might want to extract the important events from the uneventful context.

There are two general approaches to automatic summarization: Extraction and Abstraction.
1. *Extractive Summarization*: These methods rely on extracting several parts, such as phrases and sentences, from a piece of text and stack them together to create a summary. Therefore, identifying the right sentences for summarization is of utmost importance in an extractive method.

2. *Abstractive Summarization*: These methods use advanced NLP techniques to generate an entirely new summary. Some parts of this summary may not even appear in the original text. Such a summary might include verbal innovations.
Research to date has focused primarily on extractive methods, which are appropriate for image collection summarization and video summarization.

In this Jupyter notebook, TextRank algorithm for extractive text summarization is implemented using Google's PageRank search algorithm to generate correlations among sentences.

Libraries Used

- Numpy
- Pandas
- Natural Language Toolkit

Algorithms and Concepts

- TextRank
- PageRank
- Cosine Similarity

How to run

- Install the required libraries using pip, virtual environment or conda.
- Run `jupyter notebook` in your terminal.

Working of the Code:

- Importing of the required libraries and packages.

```
import numpy as np
import pandas as pd
import nltk
import re
```

```
import matplotlib.pyplot as plt
from nltk.tokenize import sent_tokenize
from nltk.corpus import stopwords
from sklearn.metrics.pairwise import cosine_similarity
import networkx as nx
```

- Extraction of word vectors.

```
word_embeddings = {}
file = open('glove.6B.100d.txt', encoding='utf-8')
for line in file:
    values = line.split()
    word = values[0]
    coefs = np.asarray(values[1:], dtype='float32')
    word_embeddings[word] = coefs
file.close()
len(word_embeddings)
```

- Reading the file and creating an object for the same

```
filename="C:\\Users\\Gaurav
Pahwa\\Downloads\\datasets\\news_articles\\002.txt"
f = open((filename), "r")
df=f.read()
f.close()
```

- Convert dataframe into dictionary

```
text_dictionary = {}
text_dictionary[1] = df
```

- Remove Stop-words

```
def remove_stopwords(sen):
    stop_words = stopwords.words('english')
    sen_new = " ".join([i for i in sen if i not in stop_words])
    return sen_new
```

- Vectorizing sentences

```
def sentence_vector_func (sentences_cleaned) :
```

```
    sentence_vector = []
    for i in sentences_cleaned:
        if len(i) != 0:
            v = sum([word_embeddings.get(w, np.zeros((100,))) for w in
i.split()])/(len(i.split())+0.001)
        else:
            v = np.zeros((100,))
        sentence_vector.append(v)
    return (sentence_vector)
```

- Printing of the Summary

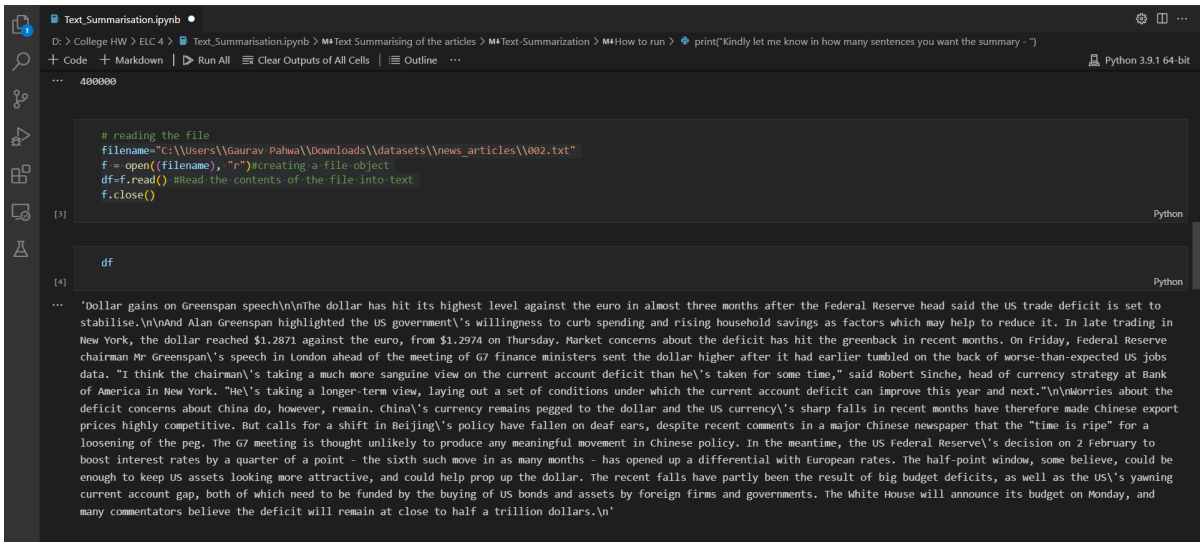
```
def summary_text (test_text, n = 5):
    sentences = []
    sentences.append(sent_tokenize(test_text))
    sentences = [y for x in sentences for y in x]
    clean_sentences = pd.Series(sentences).str.replace("[^a-z A-Z 0-9]", "
")
    clean_sentences = [s.lower() for s in clean_sentences]
    clean_sentences = [remove_stopwords(r.split()) for r in
clean_sentences]
    sentence_vectors = sentence_vector_func(clean_sentences)
    sim_mat = np.zeros([len(sentences), len(sentences)])
    for i in range(len(sentences)):
        for j in range(len(sentences)):
            if i != j:
                sim_mat[i][j] =
cosine_similarity(sentence_vectors[i].reshape(1,100),
sentence_vectors[j].reshape(1,100))[0,0]


    nx_graph = nx.from_numpy_array(sim_mat)
    scores = nx.pagerank(nx_graph)

    ranked_sentences = sorted(((scores[i],s) for i,s in
enumerate(sentences)))
    summarised_string = ''
    for i in range(n):
        try:
            summarised_string = summarised_string +
str(ranked_sentences[i][1])
        except IndexError:
```

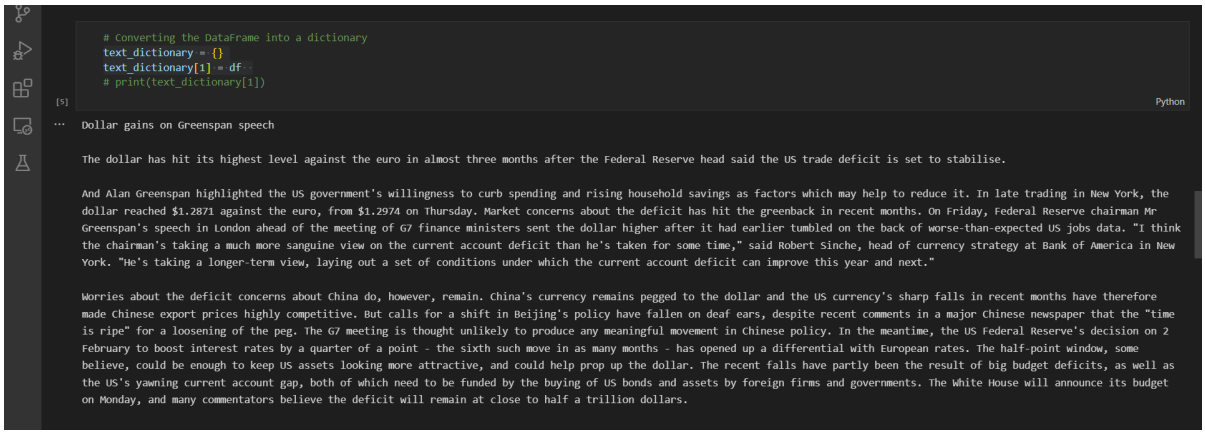
```
            print ("Summary Not Available")

    return (summarised_string)
```

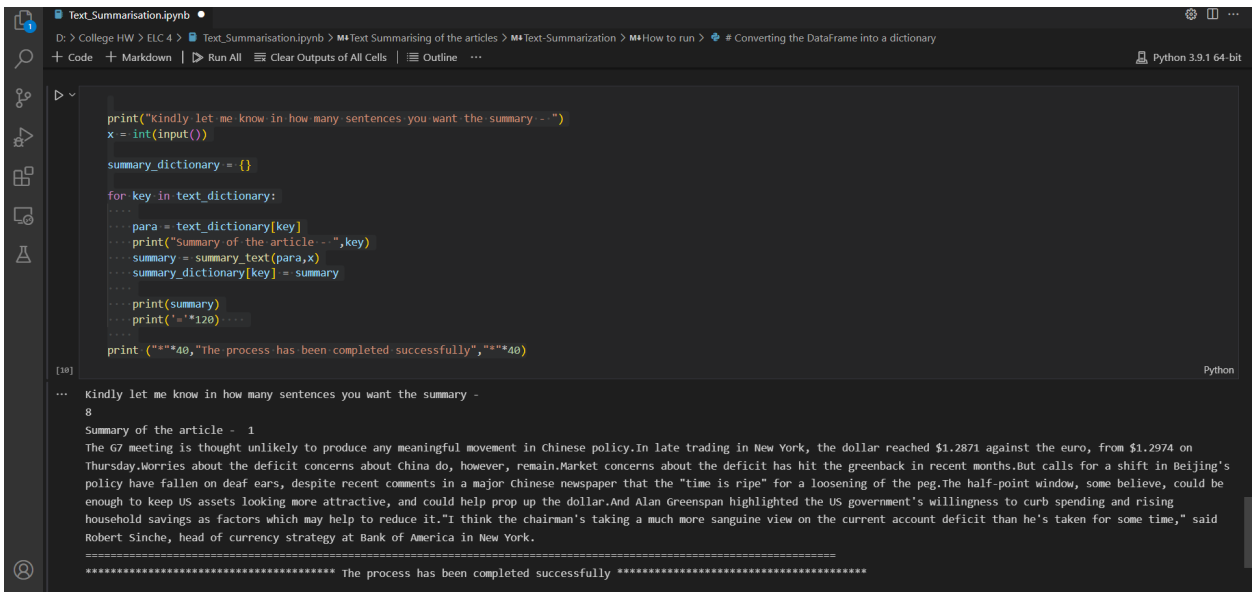
- Driver Code

```
print("Kindly let me know in how many sentences you want the summary - ")
x = int(input())

summary_dictionary = {}

for key in text_dictionary:

    para = text_dictionary[key]
    print("Summary of the article - ",key)
    summary = summary_text(para,x)
    summary_dictionary[key] = summary

    print(summary)
    print('='*120)

print ("*"*40,"The process has been completed successfully","*"*40)
```

Outputs:


Text_Summarisation.ipynb
D: > College HW > ELC 4 > Text_Summarisation.ipynb > Text Summarising of the articles > Text-Summarization > How to run > print("Kindly let me know in how many sentences you want the summary - ")
+ Code + Markdown | Run All Clear Outputs of All Cells | Outline
Python 3.9.1 64-bit
400000
# reading the file
filename="C:\\Users\\Gaurav Pahwa\\Downloads\\datasets\\news_articles\\002.txt"
f = open((filename), "r")#creating a file object
df=f.read() #Read the contents of the file into text
f.close()
[3]
Python
df
[4]
Python
'Dollar gains on Greenspan speech\n\nThe dollar has hit its highest level against the euro in almost three months after the Federal Reserve head said the US trade deficit is set to stabilise.\n\nAnd Alan Greenspan highlighted the US government\'s willingness to curb spending and rising household savings as factors which may help to reduce it. In late trading in New York, the dollar reached $1.2871 against the euro, from $1.2974 on Thursday. Market concerns about the deficit has hit the greenback in recent months. On Friday, Federal Reserve chairman Mr Greenspan\'s speech in London ahead of the meeting of G7 finance ministers sent the dollar higher after it had earlier tumbled on the back of worse-than-expected US jobs data. "I think the chairman\'s taking a much more sanguine view on the current account deficit than he\'s taken for some time," said Robert Sinche, head of currency strategy at Bank of America in New York. "He\'s taking a longer-term view, laying out a set of conditions under which the current account deficit can improve this year and next."\n\nWorries about the deficit concerns about China do, however, remain. China\'s currency remains pegged to the dollar and the US currency\'s sharp falls in recent months have therefore made Chinese export prices highly competitive. But calls for a shift in Beijing\'s policy have fallen on deaf ears, despite recent comments in a major Chinese newspaper that the "time is ripe" for a loosening of the peg. The G7 meeting is thought unlikely to produce any meaningful movement in Chinese policy. In the meantime, the US Federal Reserve\'s decision on 2 February to boost interest rates by a quarter of a point - the sixth such move in as many months - has opened up a differential with European rates. The half-point window, some believe, could be enough to keep US assets looking more attractive, and could help prop up the dollar. The recent falls have partly been the result of big budget deficits, as well as the US\'s yawning current account gap, both of which need to be funded by the buying of US bonds and assets by foreign firms and governments. The White House will announce its budget on Monday, and many commentators believe the deficit will remain at close to half a trillion dollars.\n'


# Converting the DataFrame into a dictionary
text_dictionary = {}
text_dictionary[1] = df
# print(text_dictionary[1])
[5]
Python
Dollar gains on Greenspan speech
The dollar has hit its highest level against the euro in almost three months after the Federal Reserve head said the US trade deficit is set to stabilise.
And Alan Greenspan highlighted the US government's willingness to curb spending and rising household savings as factors which may help to reduce it. In late trading in New York, the dollar reached $1.2871 against the euro, from $1.2974 on Thursday. Market concerns about the deficit has hit the greenback in recent months. On Friday, Federal Reserve chairman Mr Greenspan's speech in London ahead of the meeting of G7 finance ministers sent the dollar higher after it had earlier tumbled on the back of worse-than-expected US jobs data. "I think the chairman's taking a much more sanguine view on the current account deficit than he's taken for some time," said Robert Sinche, head of currency strategy at Bank of America in New York. "He's taking a longer-term view, laying out a set of conditions under which the current account deficit can improve this year and next."
Worries about the deficit concerns about China do, however, remain. China's currency remains pegged to the dollar and the US currency's sharp falls in recent months have therefore made Chinese export prices highly competitive. But calls for a shift in Beijing's policy have fallen on deaf ears, despite recent comments in a major Chinese newspaper that the "time is ripe" for a loosening of the peg. The G7 meeting is thought unlikely to produce any meaningful movement in Chinese policy. In the meantime, the US Federal Reserve's decision on 2 February to boost interest rates by a quarter of a point - the sixth such move in as many months - has opened up a differential with European rates. The half-point window, some believe, could be enough to keep US assets looking more attractive, and could help prop up the dollar. The recent falls have partly been the result of big budget deficits, as well as the US's yawning current account gap, both of which need to be funded by the buying of US bonds and assets by foreign firms and governments. The White House will announce its budget on Monday, and many commentators believe the deficit will remain at close to half a trillion dollars.


Text_Summarisation.ipynb
D: > College HW > ELC 4 > Text_Summarisation.ipynb > Text Summarising of the articles > Text-Summarization > How to run > # Converting the DataFrame into a dictionary
+ Code + Markdown | Run All Clear Outputs of All Cells | Outline
Python 3.9.1 64-bit
print("Kindly let me know in how many sentences you want the summary - ")
x = int(input())
summary_dictionary = {}
for key in text_dictionary:
para = text_dictionary[key]
print("Summary of the article - ",key)
summary = summary_text(para,x)
summary_dictionary[key] = summary
print(summary)
print('='*120)
print ("*"*40,"The process has been completed successfully","*"*40)
[10]
Python
Kindly let me know in how many sentences you want the summary -
8
Summary of the article - 1
The G7 meeting is thought unlikely to produce any meaningful movement in Chinese policy.In late trading in New York, the dollar reached $1.2871 against the euro, from $1.2974 on Thursday.Worries about the deficit concerns about China do, however, remain.Market concerns about the deficit has hit the greenback in recent months.But calls for a shift in Beijing's policy have fallen on deaf ears, despite recent comments in a major Chinese newspaper that the "time is ripe" for a loosening of the peg.The half-point window, some believe, could be enough to keep US assets looking more attractive, and could help prop up the dollar.And Alan Greenspan highlighted the US government's willingness to curb spending and rising household savings as factors which may help to reduce it."I think the chairman's taking a much more sanguine view on the current account deficit than he's taken for some time," said Robert Sinche, head of currency strategy at Bank of America in New York.
**** The process has been completed successfully ****

Question Answering System

We have used similar methods as used in summarization.
We have made use of TF-IDF and determined the top matching sentences.

Code:

- Conversion of directory files to dictionary

```
def load_files(directory):
    dictionary = {}
    for file in os.listdir(directory):
        with open(os.path.join(directory, file), encoding="utf-8") as
ofile:
            dictionary[file] = ofile.read()
    return dictionary
```

- Tokenizing Words into List

```
def tokenize(document):
    tokenized = nltk.tokenize.word_tokenize(document.lower())

    final_list = [x for x in tokenized if x not in string.punctuation and
x not in nltk.corpus.stopwords.words("english")]
    return final_list
```

- Computing inverse document frequencies.

```
def compute_idfs(documents):

    idf_dictio = {}
    doc_len = len(documents)
```

```
    unique_words = set(sum(documents.values(), []))

    for word in unique_words:
        count = 0
        for doc in documents.values():
            if word in doc:
                count += 1

        idf_dictio[word] = math.log(doc_len/count)

    return idf_dictio
```

- Find Top matching sentences

```
def top_sentences(query, sentences, idfs, n):

    scores = {}
    for sentence, sentwords in sentences.items():
        score = 0
        for word in query:
            if word in sentwords:
                score += idfs[word]

        if score != 0:
            density = sum([sentwords.count(x) for x in query]) /
len(sentwords)
            scores[sentence] = (score, density)

    sorted_by_score = [k for k, v in sorted(scores.items(), key=lambda x:
(x[1][0], x[1][1]), reverse=True)]

    return sorted_by_score[:n]
```

- Find the most matching files

```
def top_files(query, files, idfs, n):

    scores = {}
    for filename, filecontent in files.items():
        file_score = 0
```

```
        for word in query:
            if word in filecontent:
                file_score += filecontent.count(word) * idfs[word]
        if file_score != 0:
            scores[filename] = file_score

    sorted_by_score = [k for k, v in sorted(scores.items(), key=lambda x:
x[1], reverse=True)]
    return sorted_by_score[:n]
```

Output