Text Summarization

The idea of Text summarization is a bit different from keyphrase extraction or topic modeling. In this case, the end result is still in the form of some text, but with a few sentences based on the length we might want the summary to be. This is similar to an abstract or an executive summary in a research paper. The main objective of automated text summarization is to perform this summarization without involving human input, except for running computer programs. Mathematical and statistical models help in building and automating the task of summarizing documents by observing their content and context.

There are two broad approaches to Text summarization using automated techniques. They are described as follows:

- **Extraction-based techniques:** These methods use mathematical and statistical concepts like SVD to extract some key subset of the content from the original text such that this subset of content contains the core information and acts as the focal point of the entire text. This content can be words, phrases, or even sentences. The end result from this approach is a short executive summary of a couple of lines extracted from the original text. No new content is generated in this technique, hence the name extraction-based.
- **Abstraction-based techniques:** These methods are more complex and sophisticated. They leverage language semantics to create representations and use natural language generation (NLG) techniques where the machine uses knowledge bases and semantic representations to generate text on its own and create summaries just like a human would write them. Thanks to deep learning, we can implement these techniques easily but they require a lot of data and compute.

We will cover extraction based methods here due to constraints of needed a lot of data + compute for abstraction based methods. But you can leverage the seq2seq models you learnt in language translation on an appropriate dataset to build deep learning based abstractive summarizers

Install necessary dependencies

```
import nltk
nltk.download('punkt')
nltk.download('stopwords')
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to /root/nltk_data...
```

```
[nltk_data]   Package stopwords is already up-to-date!
True
```

Get Text Document

We use the description of a very popular role-playing game (RPG) Skyrim from Bethesda Softworks for summarization.

```
DOCUMENT = """coronaviruses are a type of virus. There are many different kinds, and some
 This is why mask-wearing, hand hygiene and physical distancing are essential to preventin
 Symptoms show up in people within two to 14 days of exposure to the virus. A person infec

Cough
Fever or chills
Shortness of breath or difficulty breathing
Muscle or body aches
Sore throat
New loss of taste or smell
Diarrhea
Headache
New fatigue
Nausea or vomiting
Congestion or runny nose
Some people infected with the coronavirus have mild COVID-19 illness, and others have no s
If you have a fever or any of the symptoms listed above, call your doctor or a health care
COVID-19 is diagnosed through a laboratory test. Diagnosis by examination alone is difficu
Treatment for COVID-19 addresses the signs and symptoms of the infection and supports peop
Yes, severe COVID-19 can be fatal. For updates of coronavirus infections, deaths and vacci

Coronaviruses are named for their appearance: “corona” means “crown.” The virus’s outer la

Since the 2019 coronavirus is related to the original coronavirus that caused SARS and can
"""


import re

DOCUMENT = re.sub(r'\n|\r', ' ', DOCUMENT)
DOCUMENT = re.sub(r' +', ' ', DOCUMENT)
DOCUMENT = DOCUMENT.strip()
```

Summarization with Gensim

Let’s look at an implementation of text summarization by leveraging Gensim’s summarization module. It is pretty straightforward.

```
from gensim.summarization import summarize

print(summarize(DOCUMENT, ratio=0.2, split=False))
```

```
A coronavirus identified in 2019, SARS-CoV-2, has caused a pandemic of respiratory il
SARS-CoV-2 may have originated in an animal and changed (mutated) so it could cause i
A person infected with the coronavirus is contagious to others for up to two days bef
Treatment for COVID-19 addresses the signs and symptoms of the infection and supports
Like other viruses, the coronavirus that causes COVID-19 can change (mutate).
Mutations may enable the coronavirus to spread faster from person to person, and may
```

```
print(summarize(DOCUMENT, word_count=75, split=False))
```

```
A coronavirus identified in 2019, SARS-CoV-2, has caused a pandemic of respiratory il
Treatment for COVID-19 addresses the signs and symptoms of the infection and supports
Mutations may enable the coronavirus to spread faster from person to person, and may
```

```
sentences = nltk.sent_tokenize(DOCUMENT)
len(sentences)
```

```
32
```

This summarization implementation from Gensim is based on a variation of a popular algorithm called TextRank.

Basic Text pre-processing

```
import numpy as np

stop_words = nltk.corpus.stopwords.words('english')

def normalize_document(doc):
    # lower case and remove special characters\whitespaces
    doc = re.sub(r'[^a-zA-Z\s]', '', doc, re.I|re.A)
    doc = doc.lower()
    doc = doc.strip()
    # tokenize document
    tokens = nltk.word_tokenize(doc)
    # filter stopwords out of document
    filtered_tokens = [token for token in tokens if token not in stop_words]
    # re-create document from filtered tokens
    doc = ' '.join(filtered_tokens)
    return doc

normalize_corpus = np.vectorize(normalize_document)

norm_sentences = normalize_corpus(sentences)
norm_sentences[:3]
```

```
array(['coronaviruses type virus', 'many different kinds cause disease',
       'coronavirus identified sarscov caused pandemic respiratory illness called cov
      dtype='<U407')
```

Text Representation with Feature Engineering

We will be vectorizing our normalized sentences using the TF-IDF feature engineering scheme. We keep things simple and don't filter out any words based on document frequency. But feel free to try that out and maybe even leverage n-grams as features.

```
from sklearn.feature_extraction.text import TfidfVectorizer
import pandas as pd

tv = TfidfVectorizer(min_df=0., max_df=1., use_idf=True)
dt_matrix = tv.fit_transform(norm_sentences)
dt_matrix = dt_matrix.toarray()

vocab = tv.get_feature_names()
td_matrix = dt_matrix.T
print(td_matrix.shape)
pd.DataFrame(np.round(td_matrix, 2), index=vocab).head(10)
```

```
(281, 32)
/usr/local/lib/python3.7/dist-packages/sklearn/utils/deprecation.py:87: FutureWarning
  warnings.warn(msg, category=FutureWarning)
```

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	1
accumulate	0.0	0.0	0.0	0.24	0.0	0.00	0.0	0.0	0.0	0.00	0.0	0.0	0.0	0.0	0.00	0
aches	0.0	0.0	0.0	0.00	0.0	0.00	0.0	0.0	0.0	0.13	0.0	0.0	0.0	0.0	0.00	0
acute	0.0	0.0	0.0	0.00	0.0	0.00	0.0	0.0	0.0	0.00	0.0	0.0	0.0	0.0	0.00	0
addresses	0.0	0.0	0.0	0.00	0.0	0.00	0.0	0.0	0.0	0.00	0.0	0.0	0.0	0.0	0.00	0
affected	0.0	0.0	0.0	0.00	0.0	0.00	0.0	0.0	0.0	0.00	0.0	0.0	0.0	0.0	0.00	0
africa	0.0	0.0	0.0	0.00	0.0	0.00	0.0	0.0	0.0	0.00	0.0	0.0	0.0	0.0	0.00	0
air	0.0	0.0	0.2	0.21	0.0	0.00	0.0	0.0	0.0	0.00	0.0	0.0	0.0	0.0	0.00	0
alone	0.0	0.0	0.0	0.00	0.0	0.00	0.0	0.0	0.0	0.00	0.0	0.0	0.0	0.0	0.35	0
also	0.0	0.0	0.0	0.00	0.0	0.00	0.0	0.0	0.0	0.00	0.0	0.0	0.0	0.0	0.00	0
animal	0.0	0.0	0.0	0.00	0.0	0.37	0.0	0.0	0.0	0.00	0.0	0.0	0.0	0.0	0.00	0

Latent Semantic Analysis

Here, we summarize our game description by utilizing document sentences. The terms in each sentence of the document have been extracted to form the term-document matrix, which we observed in the previous cell.

We apply low-rank Singular Value Decomposition to this matrix. The core principle behind Latent Semantic Analysis (LSA) is that in any document, there exists a latent structure among terms that are related contextually and hence should also be correlated in the same singular space.

The main idea in our implementation is to use SVD (recall M = USVT) so that U and V are the orthogonal matrices and S is the diagonal matrix, which can also be represented as a vector of the singular values.

The original matrix can be represented as a term-document matrix where the rows are terms and each column is a document, i.e., a sentence from our document in this case. The values can be any type of weighting like Bag of Words model-based frequencies, TF-IDFs, or binary occurrences.

1. Get the sentence vectors from the matrix **V** (**k** rows).
2. Get the top **k** singular values from **S**.
3. Apply a threshold-based approach to remove singular values that are less than half of the largest singular value if any exist. This is a heuristic and you can play around with this value if you want. Mathematically, it's $S_i = 0 \; iff \; S_i < \frac{1}{2} S$.
4. Multiply each term sentence column from **V** squared with its corresponding singular value from **S**, also squared, to get sentence weights per topic.
5. Compute the sum of the sentence weights across the topics and take the square root of the final score to get the salience scores for each sentence in the document.

The salience score computations for each sentence can be mathematically represented as follows:

$$SS = \sqrt{\sum_{i=1}^{k} S_i V_i^T}$$

```
from scipy.sparse.linalg import svds

def low_rank_svd(matrix, singular_count=2):
    u, s, vt = svds(matrix, k=singular_count)
    return u, s, vt
```

```
num_sentences = 8
num_topics = 3

u, s, vt = low_rank_svd(td_matrix, singular_count=num_topics)
print(u.shape, s.shape, vt.shape)
term_topic_mat, singular_values, topic_document_mat = u, s, vt
```

```
(281, 3) (3,) (3, 32)
```

```
# remove singular values below threshold
sv_threshold = 0.5
min_sigma_value = max(singular_values) * sv_threshold
singular_values[singular_values < min_sigma_value] = 0


salience_scores = np.sqrt(np.dot(np.square(singular_values),
                                 np.square(topic_document_mat)))
salience_scores
```

```
array([0.11348978, 0.44531554, 0.4021708 , 0.23756049, 0.29954116,
       0.48245859, 0.20870657, 0.52951637, 0.62341143, 0.53362203,
       0.13122933, 0.38377585, 0.17513233, 0.1946512 , 0.35763233,
       0.7548632 , 0.53007674, 0.39557111, 0.30740027, 0.35564197,
       0.26730856, 0.25780765, 0.26440005, 0.44086094, 0.71430503,
       0.64056528, 0.23948274, 0.34131334, 0.48129078, 0.08825606,
       0.6907127 , 0.25507288])
```

```
top_sentence_indices = (-salience_scores).argsort()[:num_sentences]
top_sentence_indices.sort()


print('\n'.join(np.array(sentences)[top_sentence_indices]))
```

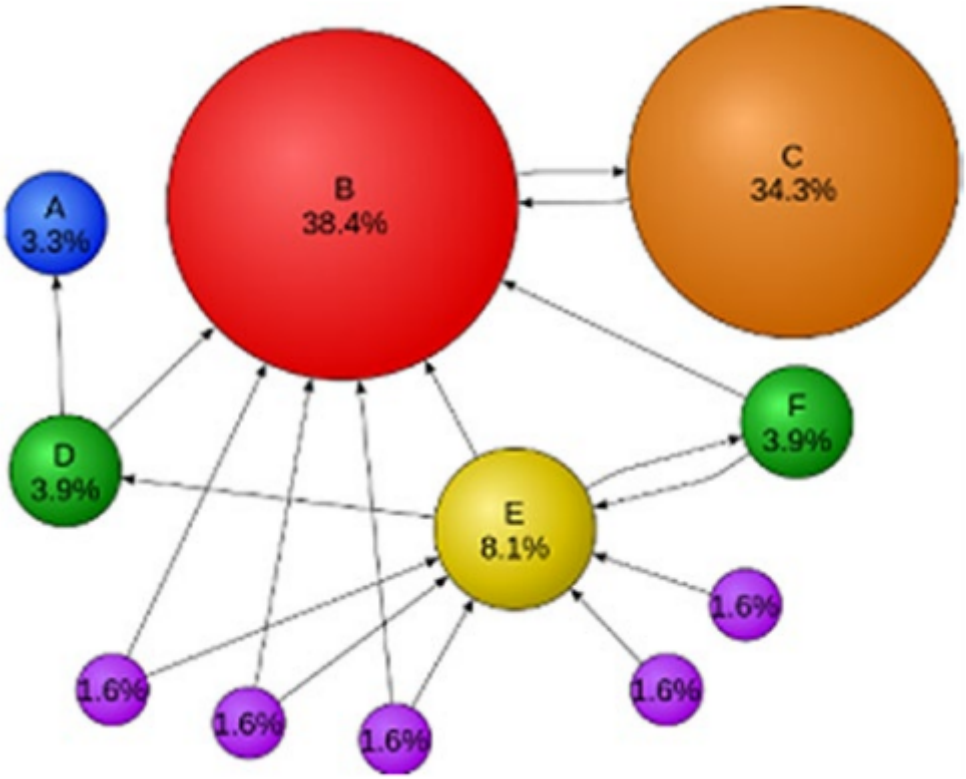
```
Research continues, and more study may reveal how and why the coronavirus evolved to
Symptoms show up in people within two to 14 days of exposure to the virus.
A person infected with the coronavirus is contagious to others for up to two days bef
Some people with the coronavirus do not have symptoms at all.
Treatment for COVID-19 addresses the signs and symptoms of the infection and supports
The coronavirus that causes COVID-19 is similar to the one that caused the 2003 SARS
Since the 2019 coronavirus is related to the original coronavirus that caused SARS ar
Mutations may enable the coronavirus to spread faster from person to person, and may
```

TextRank

The TextRank summarization algorithm internally uses the popular PageRank algorithm, which is used by Google for ranking websites and pages. This is used by the Google search engine when providing relevant web pages based on search queries. To understand TextRank better, we need to understand some of the concepts surrounding PageRank. The core algorithm in PageRank is a graph-based scoring or ranking algorithm, where pages are scored or ranked based on their importance.

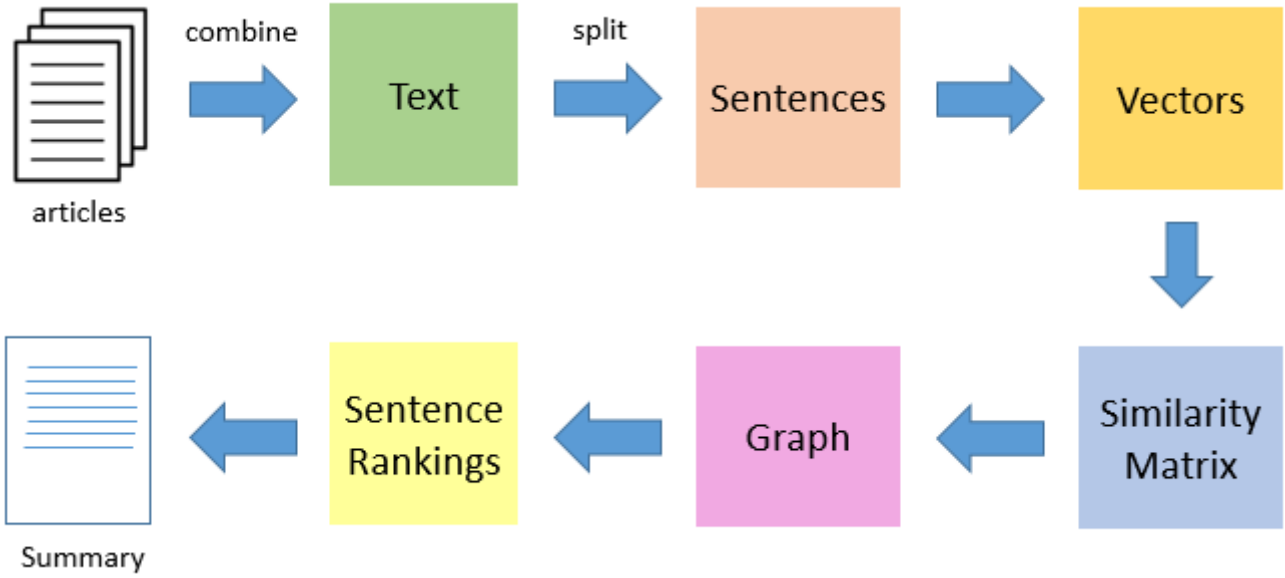
Websites and pages contain further links embedded in them which link to more pages having more links and this continues across the Internet. This can be represented as a graph-based model where vertices indicate the web pages and edges indicate links among them. This can be used to form a voting or recommendation system such so when one vertex links to another one in the graph it is basically casting a vote.

Vertex importance is decided not only on the number of votes or edges but also the importance of the vertices that are connected to it and their importance.



We can see that vertex denoting Page C has a higher score than Page E even if it has fewer edges compared to Page E, because Page B is an important page connected to Page C.

For textrank we will follow a similar process leveraging pagerank



- Tokenize and extract sentences from the document to be summarized.
- Decide on the number of sentences, k, that we want in the final summary
- Build a document-term feature matrix using weights like TF-IDF or Bag of Words.
- Compute a document similarity matrix by multiplying the matrix by its transpose.
- Use these documents (sentences in our case) as the vertices and the similarities between each pair of documents as the weight or score coefficient we talked about earlier and feed them to the PageRank algorithm.
- Get the score for each sentence.
- Rank the sentences based on score and return the top k sentences.

Build Similarity Matrix

```
similarity_matrix = np.matmul(dt_matrix, dt_matrix.T)
print(similarity_matrix.shape)
np.round(similarity_matrix, 3)
```

```
(32, 32)
array([[1.   , 0.   , 0.073, ..., 0.   , 0.   , 0.11 ],
       [0.   , 1.   , 0.   , ..., 0.   , 0.164, 0.   ],
       [0.073, 0.   , 1.   , ..., 0.043, 0.18 , 0.037],
       ...,
       [0.   , 0.   , 0.043, ..., 1.   , 0.   , 0.   ],
       [0.   , 0.164, 0.18 , ..., 0.   , 1.   , 0.089],
       [0.11 , 0.   , 0.037, ..., 0.   , 0.089, 1.   ]])
```

Build Similarity Graph

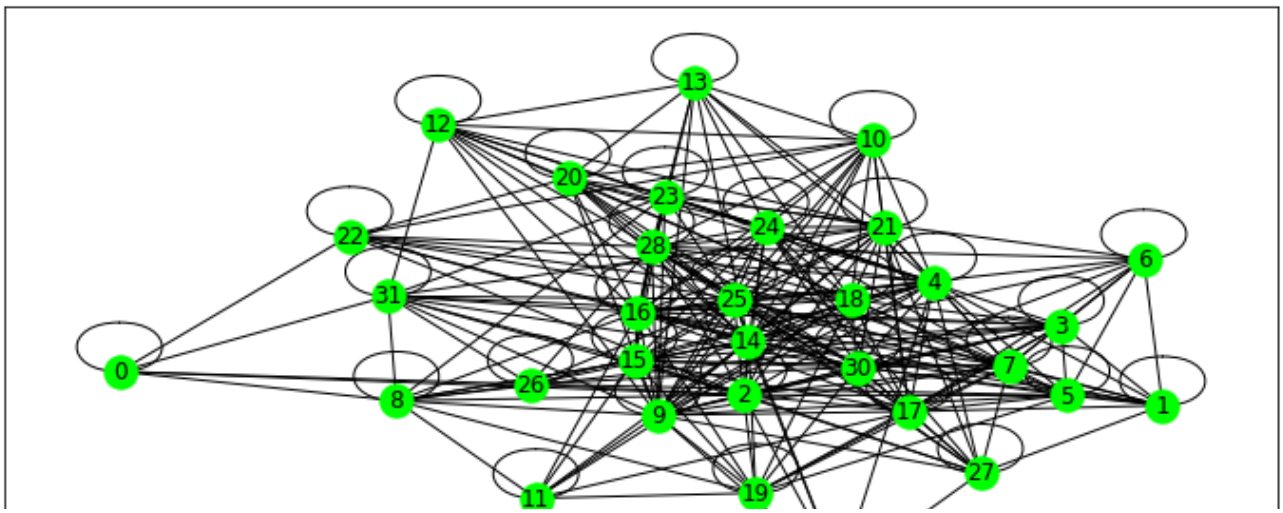
```
import networkx

similarity_graph = networkx.from_numpy_array(similarity_matrix)
similarity_graph
```

```
<networkx.classes.graph.Graph at 0x7f334ed5cdd0>
```

```
import matplotlib.pyplot as plt
%matplotlib inline

plt.figure(figsize=(12, 6))
networkx.draw_networkx(similarity_graph, node_color='lime')
```



Get Sentence Importance Scores

```
scores = networkx.pagerank(similarity_graph)
ranked_sentences = sorted(((score, index) for index, score
                                          in scores.items()),
                          reverse=True)
ranked_sentences[:10]
```

```
[(0.04748316948462508, 15),
 (0.04424923139555843, 30),
 (0.04293081835367438, 25),
 (0.04055411009731093, 16),
 (0.037201579571463125, 9),
 (0.03699551437754038, 24),
 (0.0369743303959401, 2),
 (0.033906322509520796, 28),
 (0.0333659565913274, 7),
 (0.03268575031606259, 14)]
```

```
top_sentence_indices = [ranked_sentences[index][1]
                        for index in range(num_sentences)]
top_sentence_indices.sort()
```

```
print('\n'.join(np.array(sentences)[top_sentence_indices]))
```

```
A coronavirus identified in 2019, SARS-CoV-2, has caused a pandemic of respiratory il
A person infected with the coronavirus is contagious to others for up to two days bef
Some people with the coronavirus do not have symptoms at all.
Treatment for COVID-19 addresses the signs and symptoms of the infection and supports
The coronavirus that causes COVID-19 is similar to the one that caused the 2003 SARS
Since the 2019 coronavirus is related to the original coronavirus that caused SARS ar
Like other viruses, the coronavirus that causes COVID-19 can change (mutate).
Mutations may enable the coronavirus to spread faster from person to person, and may
```

✓ 0s completed at 9:49 PM