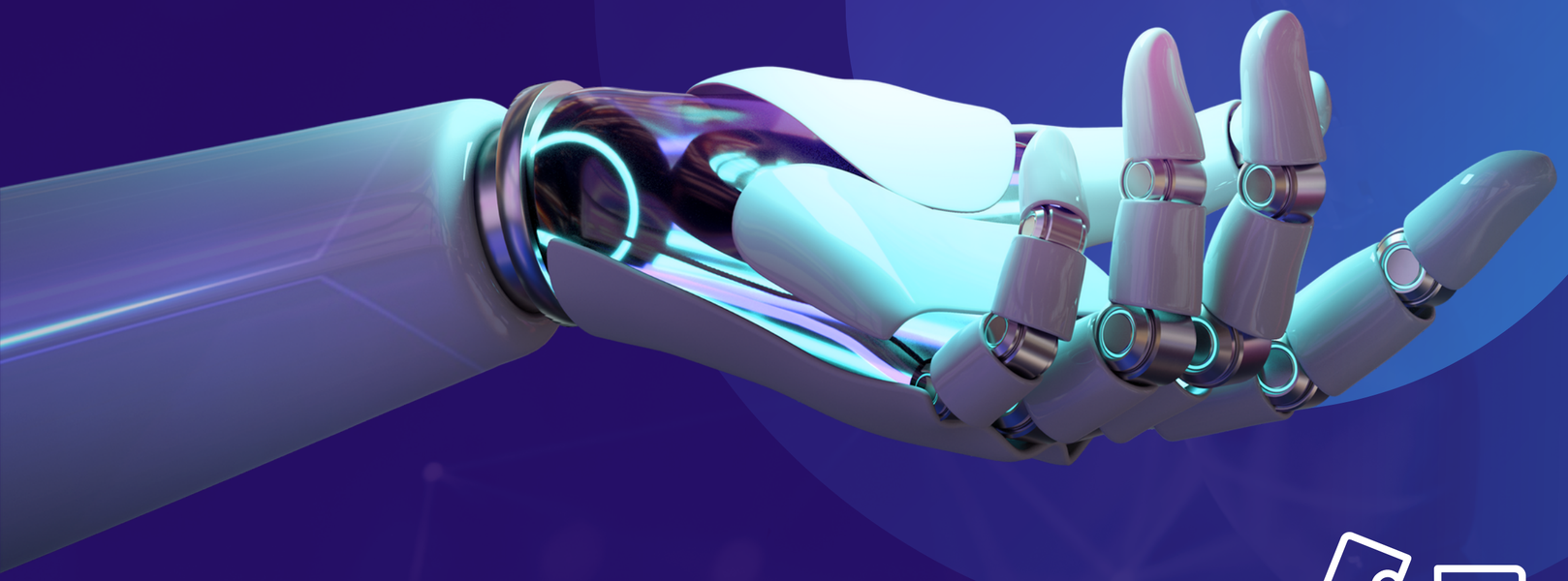
Introduction

Text summarisation is the act of transforming a lengthy sentence into a brief and condensed version.


THE
SUMMARIZER

Do you use it often?



Our purpose

Productivity

Reduce the time and effort required to read and understand large amounts of news articles

Speed

Improving the efficiency of information gathering and analysis

Reliability

Facilitating decision-making by providing concise and actionable summaries of news articles

Agenda

01

Sentiment Analysis

process of identifying positive or negative sentiment in text. Four types of sentiments—negative, neutral, positive, and compound—are given scores by the model.

02

The Model - Summariser

determine whether the two sentences are similar, generate a matrix containing NxN zeros, where N is the number of sentences, using Cosine distance in this mdoel

03

Information Extraction

a technique for automatically extracting particular types of information from text, such as named entities (people, organizations, and places), dates, and facts.

DATA EXTRACTION

DATA SCRAPING

from news platforms like CNN, NBC, and BBC, and employed by RSS (Really Simple Syndication)

DATA CLEANING

remove any irrelevant information such as advertisements and duplicate content, and ensure that the data is consistent and in a neat paragraph.

01 Sentiment Analysis

determine the positivity, negativity, and neutrality of textual data.

In [10]:
```
import pandas as pd
from nltk.sentiment import SentimentIntensityAnalyzer
nltk.download('vader_lexicon')

text_column = df['Text']

initial = SentimentIntensityAnalyzer()

# Analyze the sentiment for each text in the column
for text in text_column:
    sentiment = initial.polarity_scores(text)
    print(sentiment)
```

```
[nltk_data] Downloading package vader_lexicon to
[nltk_data]     /Users/sallypang/nltk_data...
[nltk_data]   Package vader_lexicon is already up-to-date!
{'neg': 0.035, 'neu': 0.815, 'pos': 0.149, 'compound': 0.9973}
{'neg': 0.04, 'neu': 0.852, 'pos': 0.108, 'compound': 0.9918}
{'neg': 0.042, 'neu': 0.855, 'pos': 0.103, 'compound': 0.9969}
{'neg': 0.004, 'neu': 0.9, 'pos': 0.096, 'compound': 0.9956}
{'neg': 0.042, 'neu': 0.849, 'pos': 0.109, 'compound': 0.9868}
{'neg': 0.087, 'neu': 0.83, 'pos': 0.083, 'compound': -0.5071}
{'neg': 0.069, 'neu': 0.818, 'pos': 0.113, 'compound': 0.9934}
{'neg': 0.005, 'neu': 0.877, 'pos': 0.118, 'compound': 0.9981}
{'neg': 0.054, 'neu': 0.793, 'pos': 0.153, 'compound': 0.9976}
{'neg': 0.042, 'neu': 0.813, 'pos': 0.145, 'compound': 0.999}
{'neg': 0.128, 'neu': 0.791, 'pos': 0.081, 'compound': -0.9957}
{'neg': 0.076, 'neu': 0.829, 'pos': 0.095, 'compound': 0.9559}
{'neg': 0.128, 'neu': 0.806, 'pos': 0.066, 'compound': -0.9985}
{'neg': 0.236, 'neu': 0.66, 'pos': 0.103, 'compound': -0.9998}
{'neg': 0.116, 'neu': 0.764, 'pos': 0.121, 'compound': 0.3255}
{'neg': 0.086, 'neu': 0.747, 'pos': 0.168, 'compound': 0.9979}
{'neg': 0.021, 'neu': 0.905, 'pos': 0.075, 'compound': 0.9893}
{'neg': 0.013, 'neu': 0.833, 'pos': 0.155, 'compound': 0.9984}
{'neg': 0.016, 'neu': 0.796, 'pos': 0.188, 'compound': 0.9965}
{'neg': 0.048, 'neu': 0.861, 'pos': 0.091, 'compound': 0.9966}
{'neg': 0.054, 'neu': 0.857, 'pos': 0.089, 'compound': 0.9963}
{'neg': 0.163, 'neu': 0.749, 'pos': 0.088, 'compound': -0.9994}
{'neg': 0.048, 'neu': 0.883, 'pos': 0.07, 'compound': 0.9968}
{'neg': 0.108, 'neu': 0.775, 'pos': 0.116, 'compound': 0.0963}
{'neg': 0.055, 'neu': 0.854, 'pos': 0.091, 'compound': 0.9489}
{'neg': 0.07, 'neu': 0.846, 'pos': 0.084, 'compound': 0.775}
{'neg': 0.243, 'neu': 0.722, 'pos': 0.035, 'compound': -0.9997}
```

Method of summarizing

Extractive Approach

concatenating important sentences or paragraphs without understanding the meaning of those sentences.

Abstractive Approach

generating a summary of a text from its main ideas, not by copying verbatim most salient sentences from text.

Method of summarizing

Extractive Approach

concatenating important sentences or paragraphs without understanding the meaning of those sentences.

Abstractive Approach

generating a summary of a text from its main ideas, not by copying verbatim most salient sentences from text.

02 Summariser

Summarize the news article, by using the pagerank algorithm

```python
def text_summary(article_id, top_n=3):
   # nltk.download("stopwords")
    stop_words = stopwords.words('english')
    summarize_text = []

    try:
        # Step 1 - Read text and split it to setences.
        sentences =  read_article_text(article_id)

        # Step 2 - Generate Similary Martix across sentences
        sentence_similarity_martix = create_similarity_matrix(sentences, stop_words)

        # Step 3 - Rank sentences in similarity martix
        sentence_similarity_graph = nx.from_numpy_array(sentence_similarity_martix)

        scores = nx.pagerank(sentence_similarity_graph) # PageRank (PR) is an algorithm used by Google Search to rank website
      # score is dictionary with key = node and value is its rank

        # creating the graph for similarity matrix.
        nx.draw(sentence_similarity_graph,with_labels=True)

        # Step 4 - Sort the rank and pick top sentences
        ranked_sentence = sorted(((scores[i],s) for i,s in enumerate(sentences)), reverse=True)
        # ranked_sentences are the list sorted in descending order sentences with their rank.

        for i in range(top_n):
          summarize_text.append(" ".join(ranked_sentence[i][1]))

        # Step 5 - Offcourse, output the summarize text
        print("Summarize Text: \n", ". ".join(summarize_text) + ".")

        # Step 6 writing the summarize text into a file
        with open('summary.txt', 'w') as f:
            for text in summarize_text:
                text = text.encode('utf-8')
                print(text, file=f)
        # Step 7 generating word cloud of summarize text
        word_cloud_generate('./summary.txt')
    except IndexError:
        print("This article is not in English language")
```

03 Information Extraction

Extracting all the entities that are present in the text

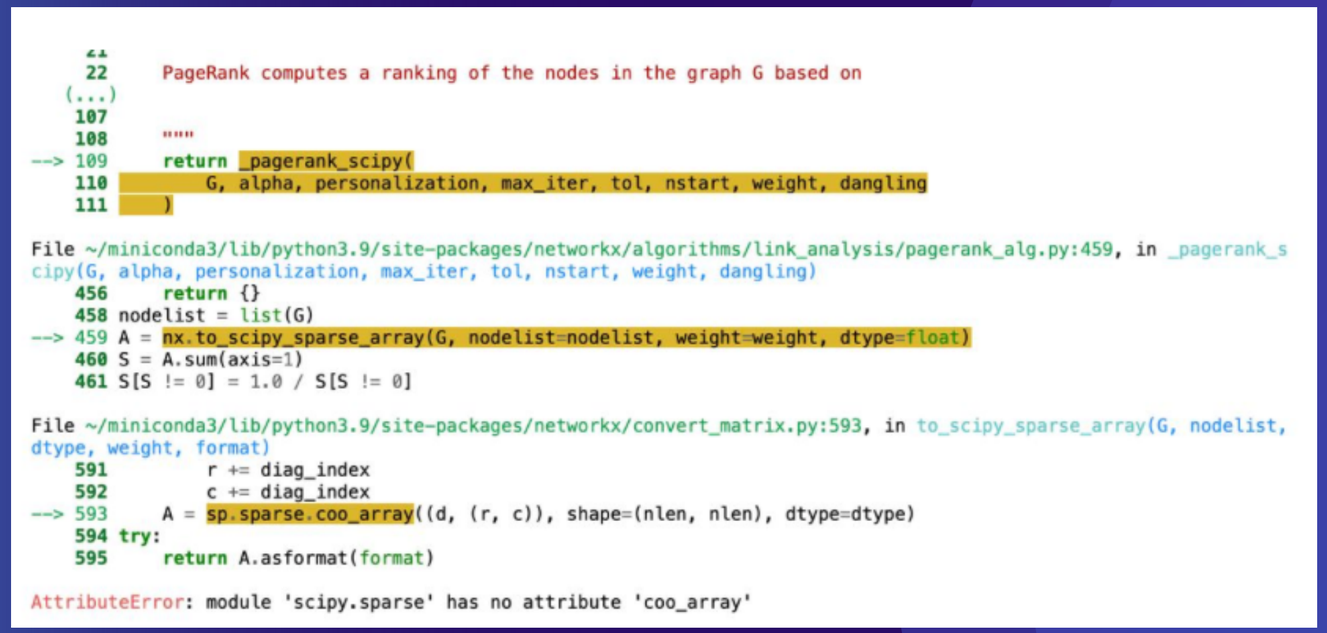
```
nlp = spacy.load("en_core_web_sm")

# Load txt file to read the summarised text
with open('/Users/richardreynard/Downloads/Final/summary.txt', 'r') as file:
    article = file.read()

doc = nlp(article)

for ent in doc.ents:
    print('{} - {}'.format(ent.text, ent.label_))
print(f"Number of Entities: {len(doc.ents)} - {doc.ents}")
```

Our Journey



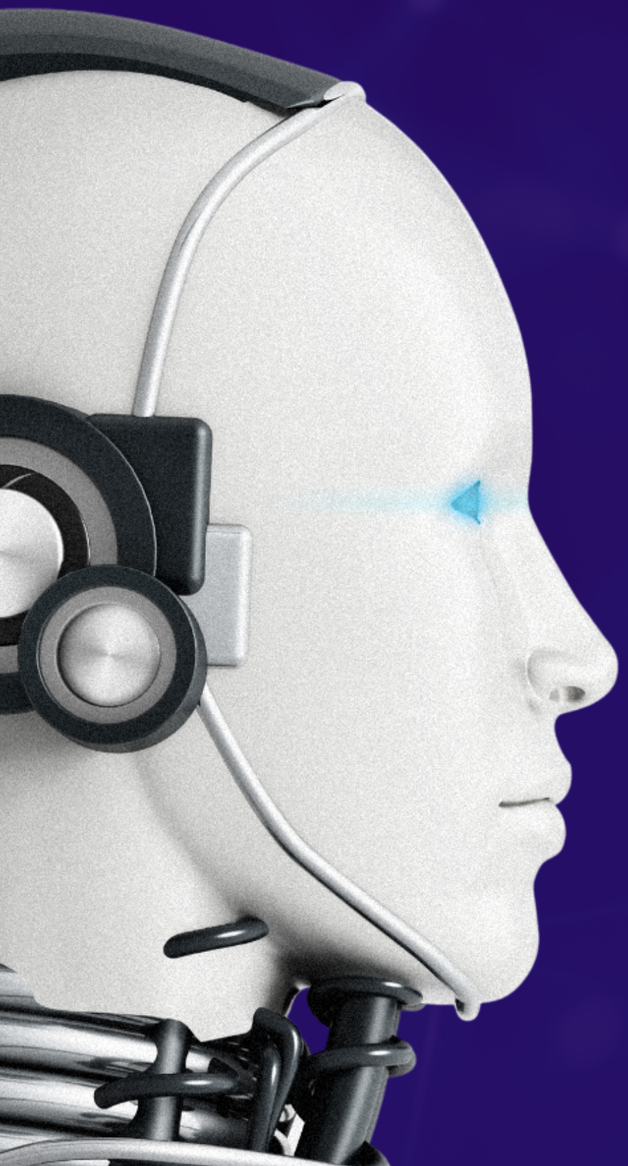
```
    21
    22     PageRank computes a ranking of the nodes in the graph G based on
   (...)
   107
   108     """
--> 109     return _pagerank_scipy(
   110         G, alpha, personalization, max_iter, tol, nstart, weight, dangling
   111     )

File ~/miniconda3/lib/python3.9/site-packages/networkx/algorithms/link_analysis/pagerank_alg.py:459, in _pagerank_s
cipy(G, alpha, personalization, max_iter, tol, nstart, weight, dangling)
   456     return {}
   458 nodelist = list(G)
--> 459 A = nx.to_scipy_sparse_array(G, nodelist=nodelist, weight=weight, dtype=float)
   460 S = A.sum(axis=1)
   461 S[S != 0] = 1.0 / S[S != 0]

File ~/miniconda3/lib/python3.9/site-packages/networkx/convert_matrix.py:593, in to_scipy_sparse_array(G, nodelist,
dtype, weight, format)
   591         r += diag_index
   592         c += diag_index
--> 593     A = sp.sparse.coo_array((d, (r, c)), shape=(nlen, nlen), dtype=dtype)
   594 try:
   595     return A.asformat(format)

AttributeError: module 'scipy.sparse' has no attribute 'coo_array'
```

- **Determine the appropriate level of summarization (e.g. sentence, paragraph, document)**
- **Evaluate the quality of the summary generated**
- **Determine web scraping algorithm**
- **Handle noise data**



Dive in the code

Utilised Jupyter Notebook is a web-based interactive computing platform.

Limitations

- Unable to summarise other languages besides English

In [23]:
```
article_id = input("Enter the index of Article index ID for Summarization: ")
article_id=int(article_id)
print("-" * 114)
print(f"Orginal Text:\n{df['Text'][article_id]}")
print("-" * 114)
text_summary(article_id)
print("-" * 114)
```

```
Enter the index of Article index ID for Summarization: 1847
------------------------------------------------------------------------------------------------------------------
Orginal Text:
ଗଣମାଧ୍ୟମରେ ଏମିତି ଅନେକ ଖବର ସାମ୍ନାକୁ ଆସିଛି, ଯିଏ ଥରେ ଦେଶ ପାଇଁ ଖେଳୁଛି ତାଙ୍କ ପାଇଁ ସରକାରୀ ଚାକିରିଟିଏ ଥୁଆ ହେଉଛି । ସେ ରାଜ୍ୟର ହେଉ ବା ଦେଶର । ସରକାରୀ ଚାକିରି ଦି
ପାହ ବଢୁଛି । ଖେଳାଳିଙ୍କୁ ପ୍ରୋତ୍ସାହନ ବି ମିଳୁଛି । ଅନ୍ୟପଟେ ଏମିତି ବି ଖବର ସାମ୍ନାକୁ ଆସୁଛି ଯାହା ଶୁଣିଲେ ଏହି ମଣିଷ ଆଖିରୁ ଦୁଇ ଧାର ଲୁହ ନିଗିଡ଼ି ଯାଉଛି । ଆଉ ତାହା ହେଲା ଯିଏ ଦି
ଦୈନନ୍ଦିନ ଜୀବନ ଅତିବାହିତ କରିବା ପାଇଁ ଦିନ ମଜୁରିଆ ଭାବେ ଖଟୁଛି । ଖାଲି ସେତିକି ନୁହେଁ ଏମିତି ବି ଖବର ସାମ୍ନାକୁ ଆସିଛି ଯେ, ଇଣ୍ଟରନାସନାଲ ଖେଳାଳି ଟାକ୍ସି ଡ୍ରାଇଭର ଭାବେ କାମ କରୁ
ବି ଆମେ ସେମିତି କିଛି କାହାଣୀ ନେଇ ଆପଣଙ୍କ ପାଖକୁ ଆସିଛୁ । ଜଣେ ଆନ୍ତର୍ଜାତୀୟ ମହିଳା ଫୁଟ୍‌ବଲ ଖେଳାଳି ଏବେ ନିଜର ଗୁରୁରାଣ ମେଣ୍ଟାଇବା ପାଇଁ ଟୋମାଟୋ ଗର୍ଲ ଭାବେ କାମ କରୁଛନ୍ତି
ର୍ଡର ମୁତାବକ ଖାଦ୍ୟ ପହଞ୍ଚାଉଛନ୍ତି । ସେ ହେଉଛନ୍ତି ଆନ୍ତର୍ଜାତୀୟ ମହିଳା ଫୁଟ୍‌ବଲର ପୌଲାମି ଅଧିକାରୀ । ମାଇକ୍ରୋ ବ୍ଲଗିଂ ସାଇଟ୍ ଟ୍ୱିଟର ରେ ତାଙ୍କର ଏକ ଭିଡିଓ ଭାଇରାଲ ହୋଇଛି । ସେଥିରେ
ବା ନଜର ଆସିଛନ୍ତି । ସେ କୋଲକତାର ବେହାଲା ଅଞ୍ଚଳରେ ରହୁଥିବା ବେଳେ ଚାରୁଚନ୍ଦ୍ର କଳେଜର ତୃତୀୟ ବର୍ଷର ଛାତ୍ରୀ । ଭାଇରାଲ ଭିଡିଓରେ ସେ କହିଛନ୍ତି ସେ କିଭଳି ଜର୍ମାନୀ, ୟୁକେ, ଶ୍ରୀଲ
ପାଖି ୧୬ଟି ପୁଟ୍‌ବଲ୍ ଟୁର୍ଣ୍ଣାମେଣ୍ଟରେ ସେ ଭାରତ ପାଇଁ ପ୍ରତିନିଧିତ୍ୱ କରିଛନ୍ତି । ପୌଲାମି ନିଜର ସ୍ୱପ୍ନକୁ ସାକାର କରିପାରି ନାହାନ୍ତି । ପରିବାରର ଆର୍ଥିକ ଅବସ୍ଥା ଭଲ ନଥିବାରୁ ସେ ବାଧ୍ୟ ହୋଇ ଚାକି
୩୦୦ରୁ ୪୦୦ ଟଙ୍କା ରୋଜଗାର ହେଉଛି । ବେଳେ ବେଳେ ୧୫୦ ଟଙ୍କାରେ ମଧ୍ୟ ମନକୁ ବୁଝାଇବାକୁ ପଡୁଛି । ଏହି ଭାଇରାଲ ଭିଡିଓ କୁ ଏବେ ପ୍ରାୟ ୫୫ ହଜାର ଲୋକ ଦେଖିସାରିଲେଣି । ସମ
ami Adhikary a football player who has represented India at the international level. Today she has to support her
n. #football pic.twitter.com/pGnJ0QOUEg
------------------------------------------------------------------------------------------------------------------
This article is not in English language
------------------------------------------------------------------------------------------------------------------
```

Limitations

- Some minor errors in entity recognition

```
nlp = spacy.load("en_core_web_sm")

# Load txt file to read the summarised text
with open('/Users/sallypang/Library/CloudStorage/OneDrive-JamesCookUniversity/MA 3831/MA 3831 - Assessment02/summary.txt', 'r
    article = file.read()

doc = nlp(article)

for ent in doc.ents:
    print('{} - {}'.format(ent.text, ent.label_))
print(f"Number of Entities: {len(doc.ents)} - {doc.ents}")
```

```
Hyun Yang - PERSON
Epigenetic - NORP
three - CARDINAL
worse,\xe2\x80\x9d Sinclair - ORG
Sinclair - ORG
Number of Entities: 5 - (Hyun Yang, Epigenetic, three, worse,\xe2\x80\x9d Sinclair, Sinclair)
```

Summary

- Text summarizing enables users to **condense** enormous amounts of text into manageable bits without omitting any essential information.
- Offers a **straightforward**, informal overview of lengthy texts.
- **Saves time and cost** while also allowing us to pinpoint key dates, locations, and other details.
- ChatGPT and Grammaly has proven the output accuracy of the model by **80%-93%**