Lab4-p2-report

1. Message embedding

We used the doc2vec library for the messages scraped from the Reddit website.

```
# get vectors
def get_model(df):
    print('Getting vectors for messages...')
    model = Doc2Vec(vector_size=150, window=10, min_count=1, epochs=100)
    documents = []
    for i, row in df.iterrows():
        documents.append(TaggedDocument(words=row['content'].split(), tags=[i]))
    model.build_vocab(documents)  # 'documents' is your list of TaggedDocument objects
    model.train(documents, total_examples=model.corpus_count, epochs=model.epochs)
    document_vectors = [model.infer_vector(doc.words) for doc in documents]
    df['document_vecs'] = document_vectors
    return [model, df]
```

I returned the model and the processed dataframe with vector for future use.

2. Information clustering

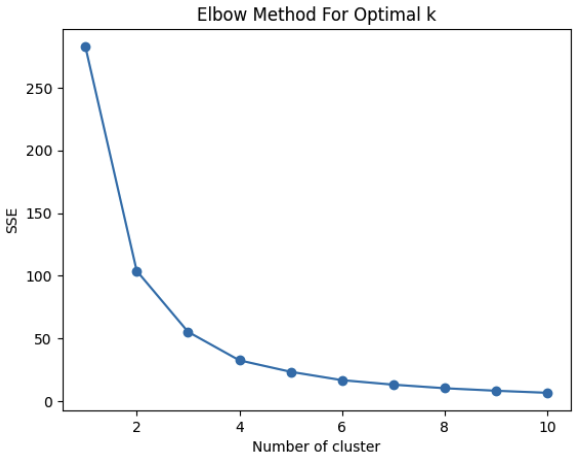
We used Jupyter Notebook for our initial analysis. First things first, we read the result_data_with_vectors by using doc2vec

:

	Unnamed: 0	timestamp	content	Keyword	Image_text	document_vecs
0	0	2023-09-21T10:48:34.018000+0000	Bionic silkworms with spider genes spin fibers...	['Bionic', 'silkworms', 'spider genes', 'spin ...	NaN	[-0.08168349 0.17292932 0.11704982 0.014653...
1	1	2023-09-21T10:25:25.589000+0000	Noisecanceling robots to mute loud conversatio...	['Noisecanceling robots', 'mute loud conversat...	_ &	[-0.11199117 0.23236823 0.16182089 0.016255...
2	2	2023-09-22T00:38:37.747000+0000	This 90000 fireproof tankbot will scout burnin...	['fireproof', 'tankbot', 'scout', 'people', 'r...	NaN	[-0.08025853 0.17095023 0.11647952 0.012742...
3	3	2023-09-20T00:29:45.012000+0000	Project Gutenberg releases 5000 free audiobook...	['Project', 'free audiobooks', 'neural texttos...	NaN	[-0.11156592 0.23531285 0.16593556 0.010631...
4	4	2023-09-19T14:31:35.691000+0000	Intels glass substrate promises 1T transistors...	['Intels glass', 'substrate', 'T transistors',...	NaN	[-0.0885584 0.1888337 0.12871514 0.007256...

Our methodology uses K-means clustering to calculate the numerical value of document_vecs, once we group by each cluster, then each cluster is associated with the keyword. In addition, we calculate the frequency of the most common keyword in the cluster and then use those keywords to do the visualization.

1) we use the Elbow method for finding the optimal K, as you can see k=6 is the value of k at the “elbow” i.e. the point after which the distortion/inertia starts decreasing in a linear fashion



2)We input the data to the k mean model and return the keywords.

[31]:
```
from sklearn.cluster import KMeans

# Set number of clusters
num_clusters = 6  # Adjust this value based on your data

# Fit K-Means
kmeans = KMeans(n_clusters=num_clusters)
df['cluster'] = kmeans.fit_predict(vectors)
```

```
/Users/iceiceice/anaconda3/lib/python3.10/site-packages/sklearn/cluster/_kmeans.py:1416: FutureWarning: The default v
alue of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
```

[32]:

```
Cluster 0:
Original Keywords: ["['Noisecanceling robots', 'mute loud conversations', 'cafe', 'smart speaker', 'deeplearning', 'a
lgorithms', 'detect', 'loud', 'locations', 'room']", "['Project', 'free audiobooks', 'neural texttospeech technolog
y', 'audiobook', 'voice']", "['Soft Robot', 'Walks', 'Repeatedly Blowing Itself', 'explosive', 'actuation', 'insectsc
ale robots jumps']", "['UC', 'chemists', 'battery', 'water', 'power', 'design', 'membraneseparator', 'parts', 'batter
ies', 'energy storage material', 'improves', 'performance', 'cost']", "['Radio quiet boxes', 'worlds', 'telescope  En
gineers', 'Small Modular Aggregation', 'RFoF Trunk', 'SMART boxes']", "['Food fraud', 'cost', 'year', 'food makers',
'microchips', 'silicon', 'chips', 'grain', 'sand', 'cost', 'pChip', 'blockchain technology', 'trace products', 'growe
rs', 'grocery shelves']", "['Blamed', 'fouling', 'environment', 'polyester', 'team', 'Cornell', 'reuse', 'compounds',
'fabrics', 'fire', 'resistant', 'antibacterial', 'wrinklefree', 'proliferation', 'garment', 'waste', 'landfills']", "
['Paralyzed', 'Patients', 'Speak', 'AIPowered Brain Implants', 'restore', 'speech', 'people', 'silenced', 'brain inju
```

3)After obtaining the data, we conduct basic data processing, which includes removing stopwords and ensuring uniform formatting. Then, based on the frequency of each keyword, we visualize the top 50 common keywords using Python's WordCloud.

```
]: om wordcloud import WordCloud
   port matplotlib.pyplot as plt
   om collections import Counter

   opwords = set(['new', 'people', 'researchers', 'size', 'and', 'the', 'scientists','human','day'])

   m_clusters = df['cluster'].nunique()  # Ensure you have the correct number of clusters

   r cluster in range(num_clusters):
     print(f"Cluster {cluster}:")
     keywords_in_cluster = df[df['cluster'] == cluster]['Keyword']

     # Convert all text to lowercase for consistency
     all_keywords = ' '.join(keywords_in_cluster).lower().split()

     # Filter out stopwords
     filtered_keywords = [word for word in all_keywords if word not in stopwords]

     word_freq = Counter(filtered_keywords)
     #print(word_freq)  # Debugging line: Print out the word frequencies

     # Get top keywords (adjust indices as needed)
     top_keywords = dict(word_freq.most_common()[3:54])  # Adjust as needed

     # Generate WordCloud
     wordcloud = WordCloud(stopwords=stopwords, width=800, height=40, background_color='white').generate_

     plt.figure(figsize=(10,5))
     plt.imshow(wordcloud, interpolation='bilinear')
     plt.title(f"Top Keywords in Cluster {cluster}")
     plt.axis('off')
     plt.show()
```

Cluster 0:

Cluster 1:

Cluster 2:

Cluster 3:

Cluster 4:

3. Automation

In order to implement automation, we first modularize our scripts into three parts, scraper.py, clustering.py and automation.py. Inside automation.py, we also defined storing and fetch function in order to store and retrieve data from MySQL database.

```
def storing(df):
    print('Storing to database...')
    pymysql.install_as_MySQLdb()
    engine = sqlalchemy.create_engine("mysql+mysqldb://root:Dsci-560@localhost/
information_clustering")

    #df.to_csv('result_data.csv')
    df.to_sql('reddit_result', engine, if_exists = 'replace', index=True)
    print('Successfully updated!')

def fetch():
    print('Getting data from database...')
    pymysql.install_as_MySQLdb()
    engine = sqlalchemy.create_engine("mysql+mysqldb://root:Dsci-560@localhost/
information_clustering")
    with engine.connect() as conn:
        result = conn.execute(text("SELECT * FROM reddit_result")).fetchall()
    return result
```

In our main function, we call scraper, clustering, store function sequentially if the user input is a number (indicating time intervals). And if user input quit in the process (use sys.stdin to detect), the script would stop updating.

The screenshot below shows the output of user input quit after one iteration:

```
if __name__ == "__main__":
    #pd.set_option('display.max_colwidth', None)
    if sys.argv[1] != 'quit' and sys.argv[1].isnumeric():
        while True:
            print('Updating... Please type the word quit and press enter if you want to stop. (The data would be automatically
updated if no input in next 5 secs)')
            time.sleep(5)
            ready_to_read, _, _ = select.select([sys.stdin], [], [], 0)
            if ready_to_read:
                user = sys.stdin.readline().strip()
                if user == 'quit':
                    break
            else:
                # print('System sleeping... Please type your keyword if you want to check the results.')
                df = scraper.get_data()
                #df = pd.DataFrame(df)
                model, df = get_model(df)
                df = clustering.kmeans_clustering(df)
                storing(df)
                print('System sleeping... Please type the word quit and press enter if you want to stop after sleeping.')
                time.sleep(60*int(sys.argv[1]))
                ready_to_read, _, _ = select.select([sys.stdin], [], [], 0)
                if ready_to_read:
                    user = sys.stdin.readline().strip()
                    if user == 'quit':
                        break
```

```
cindy@cindy-QEMU-Virtual-Machine:~/Desktop/XinyiZhang_9328705976/lab04$ python3
automation.py 1
Updating... Please type the word quit and press enter if you want to stop. (The
data would be automatically updated if no input in next 5 secs)
Extracting keywords...
Getting vectors for messages...
Storing to database...
Successfully updated!
System sleeping... Please type the word quit and press enter if you want to stop
 after sleeping.
quit
cindy@cindy-QEMU-Virtual-Machine:~/Desktop/XinyiZhang_9328705976/lab04$
```

If the user input argument is a word or phrase, the script would use the model generated when getting message vectors to get the vector for user input and get the cluster of the most similar message for the phrase and display.

```
if not sys.argv[1].isnumeric():
    print(sys.argv[1])
    search = sys.argv[1:]
    search.append('tech')
    df = fetch()
    df = pd.DataFrame(df)
    #df, cluster_df = clustering.kmeans_clustering(df)
    model, df = get_model(df)
    inferred_vector = model.infer_vector(search)
    #similar_docs = model.docvecs.most_similar([inferred_vector], topn=1)[0]
    similar_doc_index = model.dv.most_similar([inferred_vector], topn=1)[0][0]
    cluster = df.iloc[similar_doc_index][['cluster']].iloc[0]
    cluster_df = df.groupby(['cluster']).agg({'Keyword':list, 'content':list}).reset_index()
    cluster_df['Keyword'] = cluster_df['Keyword'].apply(lambda x: [i for i in x if i not in stopwords])
    cluster_df['Keyword'] = cluster_df['Keyword'].apply(lambda x: Counter(x).most_common(5))
    display = cluster_df[cluster_df['cluster'] == cluster][['Keyword','content']]
    print(display.to_string(header=False))
```

Below is the screenshot of user input a search word and our script shows the keywords and messages associated with the user input:

```
cindy@cindy-QEMU-Virtual-Machine:~/Desktop/XinyiZhang_9328705976/lab04$ python3 automation.py battery
Getting data from database...
Getting vectors for messages...
1  [('help runners sprint faster', 'robotic exoskeleton', 1), ('stimulation system could help quadriplegic patients m
ove', 'arc nerve', 'arms', 1), ('first legal level 3 automated driving system', 'us', 'try', 1), ('regrow teeth enter
s clinical trials', 'world ’', '1st drug', 'national', 1), ('nasa battery tech', 'satellites brings grid', 'battery b
uilt', 'scale storage', 'grid', 1)]  [This robotic exoskeleton can help runners sprint faster, The ARC nerve-stimulat
ion system could help quadriplegic patients move their arms again, We try out the first legal level 3 automated drivi
ng system in the US, World’s 1st drug to regrow teeth enters clinical trials - National, NASA battery tech to deliver
 for the grid. A battery built for satellites brings grid-scale storage down to Earth., Light therapy helps the brain
 clear out toxic Alzheimer’s proteins, Watery material makes windows selectively block light and/or heat, How inverse
 vaccines might tackle diseases like multiple sclerosis, University of Maryland Medical Center performs second pig-to
-human heart transplant, Bionic silkworms with spider genes spin fibers 6x tougher than Kevlar, This insect-sized rob
ot can carry 22 times its own weight | The four-legged miniature machine is powered by tiny explosions., This driverl
ess car company is using chatbots to make its vehicles smarter., How software that tracks covid variants could protec
t us against future outbreaks, Virus-searing gloves may one day replace disposables, Ag tech can cut billions of tons
 of greenhouse gas emissions., Electronic Nose 'Smells' Wildfires for Ultra-Early Detection, Scientists grow whole mo
del of human embryo, without sperm or egg, The Ukraine War has accelerated research into lithium-ion battery alternat
ives, including ones made of sand, Drug-delivery implant thwarts scar tissue by being a moving target, Scientists use
 processed coffee grounds to make stronger concrete, A biotech company says it put dopamine-making cells into people’
s brains, For the first time, AI dominates humanity's best in a real-world sport, New bioink promotes growth & regene
ration of 3D-printed muscle tissue, Quantum computer reveals chemical reaction in 100-billionth-speed slow-mo, Techni
que shows how abnormal RNA splicing leads to disease, Scientists Treat Severe Injuries in One Eye With Stem Cells Fro
```