Advanced ML - Exercise 4 Report

Joel Liurner ID 346243579

Gil Zeevi ID 203909320

Introduction

Based on Covid19 research papers, during the project we investigated methods to:

1. Compare two different papers related to covid19.
2. Represent the papers with different logics such as embedding.
3. Implement comparison algorithms based on the representations and on NLP tools.
4. Analyze the results and evaluate the methodology.

The full code can be found in the notebook repository.
Also, we worked with a training dataset from Donalds Trump twitter account in order to train a classification algorithm that would predict if he was the real author and publisher of a tweet or his staff was behind it. The algorithms and decisions can me found on the notebook repository.

Building the dataset

These are the steps taken during this section in order to build a dataset that we would work on.

- **Download and extract** the full set of papers from Kaggle on CORD19 challenge
- **Data filtering**: based on the metadata csv that has information about all available papers:
    - Sort by publish date in order to get from the newest ones.
    - Remove inconsistent and corrupted datapoints such as
        - future publish dates
        - missing abstracts or actual file
        - missing sha identifier (unique index hash per paper on the dataset).
- **Paper selection**: chose the 20000 recently published papers and copied the pubMed central file and saved the text from them.
- **Dataset building:** dropped many columns from metadata that were not useful for the project and kept features such as sha identifier, title, text, abstract.

	sha	title	abstract	journal	text
0	c86d5946ce78b0c8fc0e43fb1a33c1a5ee3feeef; c0e2...	Assessing face masks in the environment by mea...	The use of face masks outside the health care ...	Sci Total Environ	As the name implies, single-use face masks are...
1	819e829dbefd87a2eaf166bca8dfdf8476aed245	Pharmaceutical compounds used in the COVID-19 ...	During the COVID-19 pandemic, high consumption...	Sci Total Environ	The presence of pharmaceutical compounds and t...
2	1840b4d970cd26945d9d11c26fa043f44aa26c19; ce63...	Health consequences of disinfection against SA...	Individuals who get involved in the disinfecti...	Sci Total Environ	COVID-19, a disease caused by the severe acute...

Part 1: model for paper representation

In this part of the assignment we created a function that given two papers, or part of it such as title or abstract, would return a similarity measure between them. The models used for word representations were a BERT transformer model and Word2Vec. After embedding the papers into a vector, similarity was calculated by using cosine similarity.

Transformers

The main representation used for the task was based on Transformers was SPECTER which is Document-level Representation Learning using Citation-informed Transformers. From prior experience in the field we know that it is a robust transformers model which is fitted for the exact task of dealing with scientific documents. It is cable of getting an article's abstract as an input sequence, but is limited up to 512 tokens. This is a limitation that affects mostly on the abstract but not on the title. It is a pretrained model that can be initiated and used as follows:

```
from sentence_transformers import SentenceTransformer
model_specter = SentenceTransformer('allenai-specter')
embedded_text = model_specter.encode(input_text)
```

Word2Vec

This word embedding representation algorithm or technique based in a neural network and used to represent different words in a document as a vector.
Together with the word tokens, we used spaCy library and one of its most common pipelines “en_core_web_sm”. This is a pipeline that serves as a pretrained model and is mostly used for web text such as blogs.
The implementation includes:

1. Conversion of input paper to word tokens by lowercasing and tokenizing words.
2. Apply the chosen spacy model
3. Vector averaging within the word tokens

It can be initiated and used as follows:

```
import spacy, gensim
spacy_model = spacy.load('en_core_web_sm')
words = gensim.utils.simple_preprocess(input_text)
vectors_text = model(words)
embedded_ext = sum(token.vector for token in vectors_text) /
float(len(vectors_text))
```

It is important to note that this is a basic implementation and could be enhanced by removing stopwords, normalizing, and other tools while preprocessing. In addition, the “en_core_web_sm” model is basic and not the most suitable for academic papers. It was chosen this way to learn to work with the models and library but mostly to compare and validate the results from SPECTER which is the main model of the project - we will observe this by the model evaluation section.

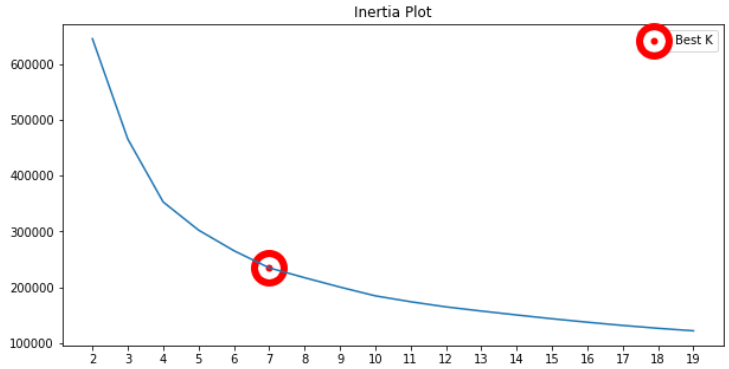
Part 2 - Model analysis and evaluation

SPECTER Model analysis

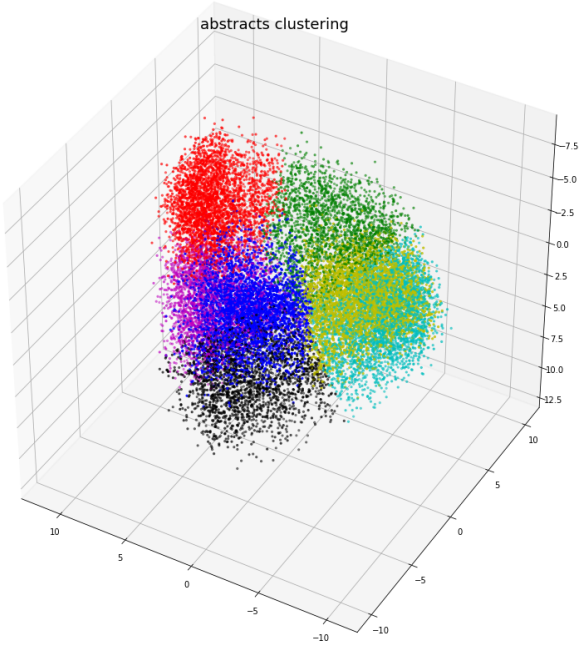
When applying the model to each of the abstract documents, the output vector has 768 dimensions. The analysis is done by reducing those dimensions to be able to interpret its results. This is done by applying PCA and retrieving the 3 principal components of the vectors.

Clustering

The next step was to perform clustering with KMeans algoriths and different number of clusters. The following result was plotted based on the inertia of each of the clustering attemps. With the concepts of the elbow method, we chose the best amount of clusters to be 7 in order to minimize inertia with meaningful groups but not compromise the amount of clusters.



The clustering results are plotted again on the following figure. The obtained silhouette coefficient was of the order of 0.3 which was not great but decent and far better than the one obtained if PCA were not to be performed (aprox 0.05).



TFIDF

Finally, the analysis was concluded by applying TFIDF to obtain the most important terms per cluster.
As per the results, we could say that the topics of the documents for each cluster are:
- Cluster 0: related to treatments
- Cluster 1: data on vaccination results
- Cluster 2: technical details on the virus spike protein.
- Cluster 3: related to students researches and results
- Cluster 4: related to high risks, acute infections and mortality
- Cluster 5: seems to be a less technical group, more related to social and online results
- Cluster 6: similar to cluster 4 on acute infections but related found results on studies.

SPECTER Model evaluation

There is no absolute way to perform evaluation on the model as it is unsupervised learning, we have no label on the data and we are not experts in the field. However, we will perform different comparisons and analysis that will qualitatively allow us to assess the results. It must be noted that to perform such analysis, we are considering as reference the performed clustering of the papers, evaluating them with other tools as well.

1. Evaluate grouped papers by SPECTER model with each other and check if the results are consistent.
2. Compare to results yielded by other models on same clustering.
3. Repeat similarity calculation of the model within clusters but enrich text samples.

Evaluate clusters similarity

The first intuitive way to see if the model is good is by accessing the clusters we identified. Several paper titles within the cluster are expected to "match" most relevant words inside the cluster. As an example, “cluster 0” words were related to treatments and covid19 effects and included [“effects, results, clinical, studies”]

Some random sampled titles from the cluster are:
- “Myasthenia Gravis Masquerading as Status Asthmaticus”
- “3D Printing of Customizable Phantoms to Replace Cadaveric Models in Upper Extremity Surgical Residency Training”
- “Site-Specific Labeling of Endogenous Proteins Using CoLDR Chemistry”
- “Pathogen Peak “Averaging” in Potable Reuse Systems: Lessons Learned from Wastewater Surveillance of SARS-CoV-2”

All of these titles look quite technical and somehow discuss health problems, possible secondary effects and medical techniques.

Evaluate clusters with other models

Another way of evaluating the model is by checking its results consitency with other models. To do so we bring back the described Word2Vec model and NCD (normalized compression distance) calculation based on gzip compression from homework 3.

The following results are calculated by using a reference paper from cluster 0 and comparing its similarity with two groups: (a) 10 papers in the same cluster and (b) one paper of each of the others clusters - 7 papers in total. What we present is the variance of the similarity for each group and model.

	Same Cluster	Specter	W2V	NCD
0	Yes	0.004545	0.000079	0.000248
1	No	0.029558	0.000871	0.103545

Results are immediate but not revealing, they only help confirm our model behaviour. Word2Vector and NCD confirm the results of Specter: the variance of similarity within one cluster is lower than with other cluster. This indeed makes sense and shows that for all models, papers in different clusters are more separate.

Abstract enrichment

Finally, we will test the SPECTER model by enriching some samples of the corpus with other samples. This will make papers more similar thus bringing them closer. The following results show the similarity of a reference paper from cluster 0 with respect to a paper of each of the other clusters, regular similarity and enriched comparison with the reference sample.

	Regular Comparison	Enriched Comparison
0	1.000000	0.984011
1	0.638931	0.801007
2	0.532656	0.579946
3	0.540288	0.802859
4	0.590241	0.705407
5	0.484160	0.626126

We can observe that by enriching the text it will actually bring closer the papers as they will be more similar, and this is consistent with the model. It is worth noting that the only sample that does not follow this is number 0 as it is comparing the paper with itself, and by enriching it we are modifying it. Finally, as sanity check, we should observe that the variance on the distance to the samples should be reduced when these samples are enriched.
Regular Specter Distance to different cluster samples variance: 0.0296
Specter Distance to different cluster enriched samples variance: 0.0177

Part 3 - Tweet prediction model

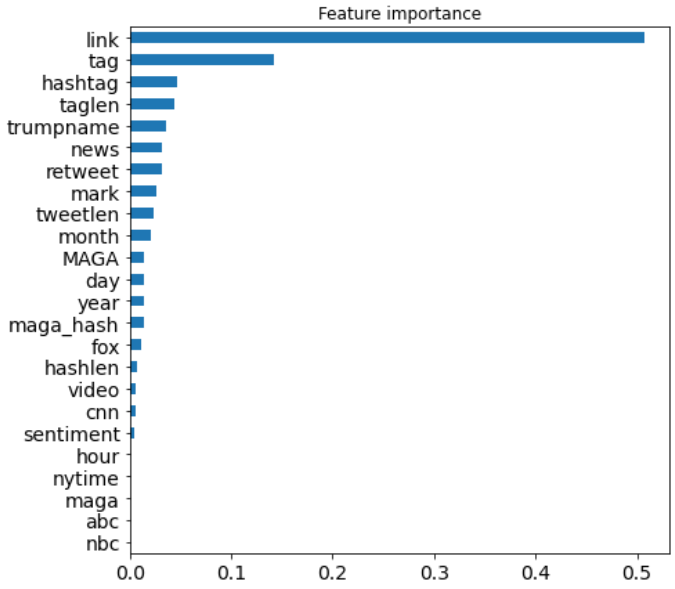
Within this task, we performed some exploratory data analysis to understand the distribution and characteristics of Trump tweets and its staff tweets. Given that, we were able to engineer several features and create models to predict who is the author of each tweet. Also, it was useful to clean the data for some inconsistent samples: non Trump account tweets, non android or iphone publications, and other character cleaning.

Feature engineering

Based on the data, we generated several features to be able to classify the tweets. These included but not limited to:

- date features
- lenght of tweet
- number of hashtags
- if it included certain words such as trump's name, cnn, nytimes, “Make America Great Again” and so on.
- sentiment analysis evaluation
- if it included images or urls

We applied XGBoostClassifier algorithm to decide on the most important engineered features and following are the results. The top ten features were chosen for classification models.



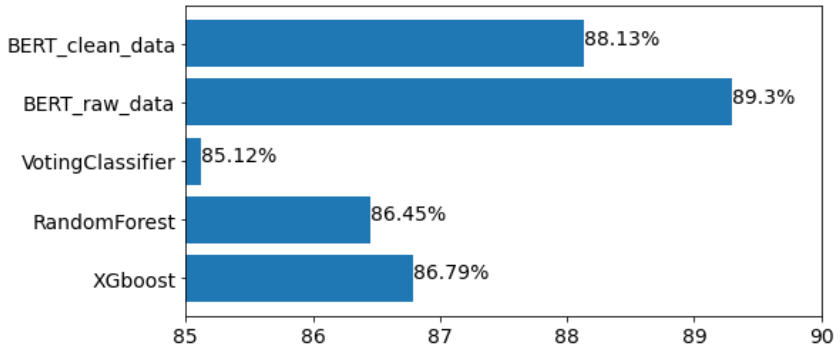
Classification models and results.

Several models were applied to classify the tweets. Here we post the results. To evaluate them, we split the dataset 80/20% into training and validation.

1. Top 10 engineered features with Random Forest
2. Top 10 engineered features with Gradient Boosting
3. Top 10 engineered features with Voting Classifier, based on:
    a. Decision Tree with max depth 12
    b. K-Neighbours with k=5

    c. Support Vector Machine (SVC) with rbf kernel
    d. Gaussian Naive Bayes Classifier
4. BERT-based-uncased classifier, non based on engineered features but on the tweet text. Two models were applied:
    a. To the raw unprocessed text
    b. Cleaned tweet text by removing characters or sequences such as “RT” or “https://”.

The results show that the BERT model with raw tweet text was the most accurate:



It is important to consider that the obtained results were based on a small validation set, and not applied to a test set.