Word/Sentence Embeddings – Random (Ecostate) vs Trained Models

Vijay Mallidi

19635574

A report submitted for

300597 Master Project 1

in partial fulfillment of the requirements for the degree of

Master's in Data science

Supervisor: Laurence Park

School of Computing, Engineering and Mathematics
Western Sydney University

June 2020

ABSTRACT

Word embeddings is one of the major advancements in the fields of machine learning and Natural Language processing systems. word embedding is the conversion words to high dimensional vectors, where all the words with similar meaning stays close to each other in the vector space. Word embeddings deals with the possibility that the words can have multiple degrees of similarities. One approach to Word embeddings is to train the model using large set of documents corpus that consumes a lot of time that invites scalability issues. The other approach is to use the models that does not require any training (pre trained models) that are called Random Ecostate models. One of the best way to compare these two approaches is to use the SentEval, an Evaluation tool kit for universal sentence representations. SentEval performs set of tests on the results produced by both the approaches to get the accuracy which can be used to compare.

ACKNOWLEDGMENTS

I would like to take time and express my gratitude and appreciation to all who gave me this wonderful platform to complete the report. A special thanks to my mentor, Mr. Laurence Park of the school of Computing, Engineering and Mathematics at the Western Sydney University for continuous suggestions and encouragement throughout this unit and especially in writing this report.

I would like to thank my parents without which none of this would have been possible for me. I am also thankful to my friends who have been continuously assisting me throughout my time in Sydney and been with me during thick and thin.

TABLE OF CONTENTS

Chapter Page

LIST OF TABLES

LIST OF FIGURES

CHAPTER I: INTRODUCTION

In the earlier days most of the Language Processing Systems uses hand coded sets of rules such as grammar rules or heuristic rules. Since late 1980’s and mid 1990’s most of the natural language processing systems depend on machine learning where the system automatically learns the rules for Language processing through analysis of large sets of documents. Although this approach offers simplicity and robustness, it has many limitations such as the amount of relevant data for speech recognition that can be obtained from representing the words with similar meaning closer in a vector space.

word embeddings start the concept that the words can be represented as the dimensional vectors, where all the words with similar meaning stays close to each other in the vector space. According to this approach words can have multiple degrees of similarity. Results obtained by Learning high dimensional embeddings from a large data are the most accurate. Word embeddings has shown the promising results in processing the syntactic and semantic information which makes it useful in wide range of applications. However, this approach offers the best results, training the methods using large data sets makes it very challenging and raises scalability issues. Random Ecostate approach eliminates the complexity of the training the models and may provide similar results while using a very less data as compared to the previous trained model approach.

Testing these 2 models possesses next challenge with easy solution provided by SentEval. SentEval has 1 set of evaluations and evaluation pipeline that has a fixed standard hyperparameters which helps in avoiding the discrepancies in results.

Many researchers have done the work on any one of these two approaches. However, there is no solid evidence that favours the one approach over the other, so I intend to compare the accuracy of these two techniques. Although Random Ecostate isn’t trained, it might be as good as the Trained models because the work in high dimensional spaces, Random Projections provide a good method for dimension reduction with very little loss in accuracy.

CHAPTER II: LITERATURE REVIEW

Many researchers have done solid research and provided many useful insights that led to the development of word embeddings.

2.1 Latent Semantic Analysis

The Authors of the paper on Introduction to Latent Semantic Analysis Landauer, T. K., Foltz, P. W., & Laham, D describes Latent Semantic Analysis as a method that can be used to extract and represent the contextual-usage meaning of the words by statistical computations applied to a large corpus of text. The author describes greatly about the similarity of the meaning of the words by aggregating all the word contexts that does and does not appear. Latent Semantic Analysis (LSA) processes the large sample of words combined together to form meaningful sentences and passages, and represents the words used in these sentences and passages as points in a very high dimensional Semantic space. Latent Semantic Analysis's similarity estimates are derived from a powerful mathematical analysis which can deal with even more deeper relations. Out of many limitations of LSA some of them are that LSA does not use the order of the words, does not have any logic or relations. Even with the many limitations, LSA manages to get the decent results in getting the similarity of the meaning of the words.

Understanding LSA

Authors of this papers explained that the Latent Semantic Analysis (LSA) can be constructed in 2 ways such as "(1) a practical method of obtaining the estimates of words and similarities among the words or (2) as a model of the computational process and representations underlying substantial portions of the acquisition and utilization of knowledge”.

As a practical expedient, LSA is used to measure the relations between word to word, word to passage and passage to passage that can be easily related with human understanding involving semantic similarity. These relations produced from LSA are close to how humans interpret the meaning of the words on daily basis and also the word choice of the writers which further helps in approximating the human judgements of the similarity between words and predict the impact of these similarities has on the similarities between the passages. The predictions made by LSA are based on powerful mathematical analysis and can be used in deeper relations and produces results much better than the conventional Natural language predictors.

Information retrieval

Information retrieval can be explained as a searcher have something in mind which he or she express in their own words and then the system try to find the text that has the same meaning as what he or she searched for. Instead of matching these words with the documents that has same words in them, Latent Semantic Indexing matches with the documents that has the similar meaning as of the words used in the query. Singular Value Decomposition (SVD) offers better results in matching the documents as compared to the other previously used method. For information retrieval, the text of the documents database is represented in matrix form and subjected to SVD where each word and document are represented in small dimensional vector. The search query is also represented in a pseudo document where the weighted average of the vectors of the words used in the query.

Synonym Tests

Latent Semantic Analysis (LSA) represents similar words in similar ways but when it comes to large text corpora its not entirely true as the relationship between some words in LSA space can be mysterious such as the words ‘verbally ’ and ‘sadomasochism’ are very close to each other but are not supposed to. It is hard to point to the exact reason for these occurrences but it is possible that for some words that have more than one contextual meaning have average high dimensional placement and some words are sampled too thin so that they can be placed in proper way. It is also possible because that Latent Semantic Analysis’s bag of words methods does not follow any syntactical, logical, and non-linguistic pragmatic entailments, it misses the complete meaning of the words and get them scrambled across the LSA space.

In an experiment, Singular Value Decomposition (SVD) was performed on text parts containing about Five hundred characters (an average of about 73 words) from the first portions of each of 30,743 articles that accumulate to 4.5 million text words that resulted in sixty thousand words for a vector. In the Test of English as a Foreign Language (TOEFL) Vocabulary text usually contains questions that has only one word and contains the four alternative answers of one word each where the test taker chooses one answer that is most similar to the question. LSA is able to get 65% which is identical to the large sample of students who apply for entrance to USA from non-English speaking background country.

Errors of LSA has been compared with the errors of students. When LSA’s selection is wrong and most of the students choose correct, it may be because that the LSA is more sensitive to contextual associations and less to contrastive semantic features.

2.2 Probabilistic Latent Semantic Analysis

The Author of the paper on Unsupervised method called **Probabilistic Latent Semantic Analysis (PLSA)** Hofmann, T describes Probabilistic Latent Semantic Analysis as a novel statistical technique for the analysis of two-mode and co-occurrence data. This has applications in information retrieval, NLP, machine learning, and in related areas. Statistical views presented on Latent Semantic Analysis leads to the Probabilistic Latent Semantic Analysis where the probabilistic variable is well defined as a general model for the data used.

A statistical model which is a Latent variable model for co-occurrence data called aspect model is used in Probabilistic Latent Semantic Analysis (PLSA). Each observation of the class variables is represented as $z \in Z = \{z1, z2, z3, \ \ldots . zk\}$. The joint probability of the model is defined by the equation

$$P(d, w) = P(d)P(w|d), P(w|d) = \sum_{z \in Z} P(w|z)P(z|d)$$

Where d is for documents and

W is for words

All the aspect models start as assumption of conditional independence that d and w are conditioned independently on the state of respective latent variable. And since z has very less number of elements as compared to the number of words in the collection it is easy to predict the words if we use z as a bottle neck variable. Now the model can be parametrized to be perfectly symmetric. The model can now be represented as

$$P(d,w) = \sum_{z \in Z} P(z)P(d|z)P(w|z)$$

Maximum likelihood

Maximum likelihood can be determined by using the Expectation Maximization algorithm. As the name suggests, this algorithm has 2 steps 1) an expectation step – posterior probabilities are calculated and 2) a maximization step – parameters are updated.

Equation for Expectation

$$P(z|d,w) = \frac{P(z)P(d|z)P(w|z)}{\sum_{z^{\circ} \in Z} P(z^{\circ})P(d|z^{\circ})P(w|z^{\circ})}$$

Equation for Maximization

$$P(z) \propto \sum_{d \in D} \sum_{w \in W} n(d,w)P(z|d,w)$$

Main difference between Probabilistic Latent Semantic Analysis and Latent Semantic Analysis is the function in determining the optimal approximation. Where LSA uses the Frobenius norm and PLSA depends upon the Maximum likelihood function.

Evaluation

The performance between the training data and the unseen test data will always be different. The problem with the statistical learning theory is to generate conditions to generalize unseen data. This can be achieved by generalizing the maximum likelihood for the models also known as Tampered Expectation Maximization (TEM). The Estimation step in EM algorithm is re derived as

$$F\beta = -\beta \sum_{d,w} n(d,w) \sum_{z} P^{\circ}(z;d,w) \log P(d,w|z)\, P(Z)$$

$$+ \sum_{d,w} n(d,w) \sum_{z} P^{\circ}(z;d,w) log P^{\circ}(z;d,w)$$

Where $P^{\circ}(z;d,w) = \frac{[P(z)P(d|Z)P(w|Z)]^{\beta}}{\sum_{\grave{z}}[P(z)P(d|\grave{z})P(\grave{w}|\grave{z})]^{\beta}}$

To genialize the unseen test data, the TEM algorithm is implemented by following the below 4 steps

Step 1 – Set $\beta < -1$ then perform Estimation Maximisation.

Step 2 – Reduce $\beta < -n\beta$ (where n<1) and perform an iteration

Step 3 – As long as performance on help-out data is non-negligible continue TEM iterations otherwise go to step 2.

Step 4 – Stop when decreasing of β does not show any further improvements.

Experimental Results

Hofmann concentrated mainly on two tasks as part of evaluating the model such as 1) automatic indexing of the documents and 2) Perplexity Evaluation.

For automatic indexing of the documents Hofmann has used 4 different medium sized collection of documents namely MED, CRAN, CACM and CISI. The precision recall graphs has been shown in Figure 1 while the results of average precision recall for 9 levels one for each variance of 10% till 90% is shown in figure 2.

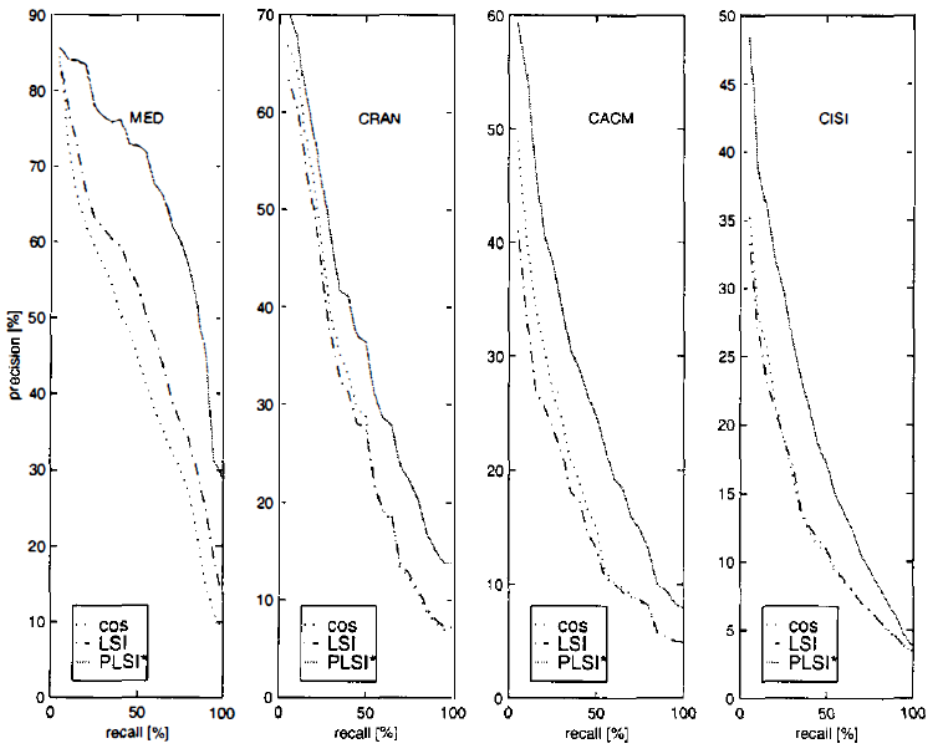
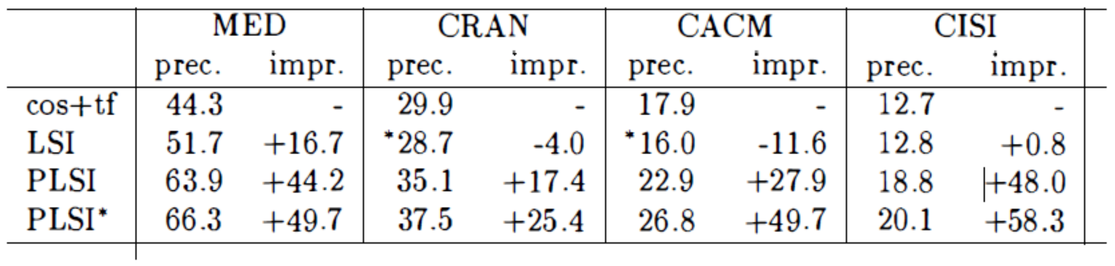


Figure 1: Precision recall curves for matching the terms on all four document collections



	MED		CRAN		CACM		CISI	
	prec.	impr.	prec.	impr.	prec.	impr.	prec.	impr.
cos+tf	44.3	-	29.9	-	17.9	-	12.7	-
LSI	51.7	+16.7	*28.7	-4.0	*16.0	-11.6	12.8	+0.8
PLSI	63.9	+44.2	35.1	+17.4	22.9	+27.9	18.8	+48.0
PLSI*	66.3	+49.7	37.5	+25.4	26.8	+49.7	20.1	+58.3

Figure 2: Average Precision results for all four document collections

Two documents collections that has been used for perplexity evolution are MED that has 1033 documents and LOB corpus that has 1674 documents. The goal is to predict the occurrences of the words in a document and to predict the conditioned nouns for the adjectives. Figure 3 shows the probability curves for both the dataset collections (a) for MED and (b) for LOB corpus.

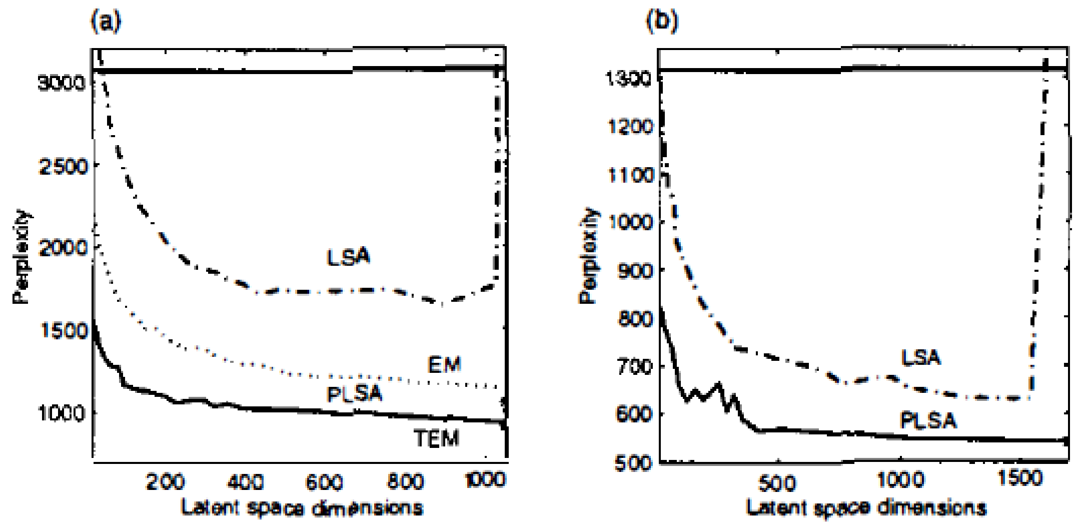


Figure 3:Perplexity results for LSA and PLSA on the dataset collections (a) for MED and (b) for LOB corpus

2.3 Latent Dirichlet Allocation

The Authors of the paper on Latent Dirichlet Allocation Blei, D.M., Ng, A.Y. and Jordan, M.I describes LDA as a model for collections of discrete data that provides full generative probabilistic semantics for documents. In LDA documents are modelled using a Dirichlet random variable which can be describes as a probability distribution on a latent low-dimensional topic space. The distribution of the words of a document which hasn't been seen yet is treated as a continuous mixture over all the documents space and a discrete mixture over all possible topics. This paper mostly concentrates on finding the correct document during the search using the words rather than just similarity between the words.

Through out the paper authors text modelling as an example as its broadly applicable to discrete data. The general assumption in LDA is that there are k latent topics based on which all the documents are generated, and topic is represented as multinomial distribution over the |v| words in the vocabulary. Then the document can be generated by selecting a sample for these mixtures of topics and words from that mixture.

To create a document with N words w = {w1, w2, w3 . . .wN}. From Dirichlet distribution θ is sampled. Then for each word in the document a topic $z_n \in \{1,..,k\}$ is sampled from multinomial distribution. The probability of a document can be written as the following equation. The graphical model is represented in figure 4.

$$p(w) = \int_{\theta} (\prod_{n=1}^{N} \sum_{z_n}^{k} p(w_n|z_n;\beta)p(z_n|\theta))p(\theta;\alpha)d\theta$$

Where $p(\theta;\alpha)$ is Dirichlet,

$p(z_n|\theta)$ is multinomial parameterized by θ

$p(w_n|z_n;\beta)$ is multinomial over the words

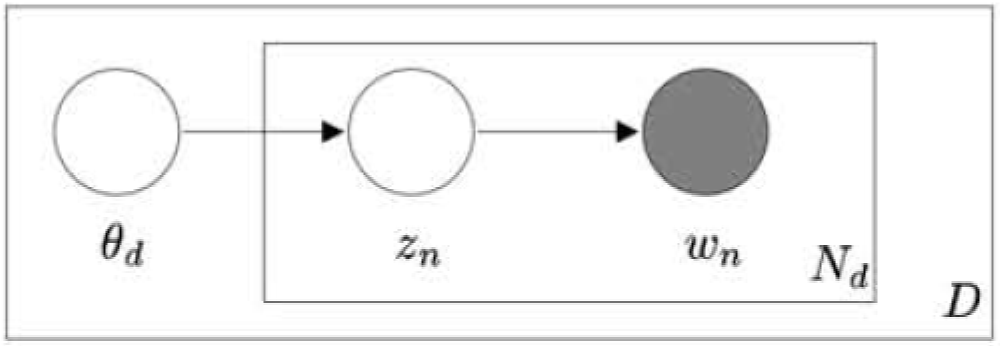


Figure 4:Graphical model representation of LDA

Inference and Learning

Expanding Equation 2.3.1 we have:

$$p(W;\alpha,\beta) = \frac{\tau(\sum_i \alpha_i)}{\prod_i \tau(\alpha_i)} \int_\theta \left(\prod_{i=1}^{k} \theta_i^{\alpha_{i-1}}\right) (\prod_{n=1}^{N} \sum_{i=1}^{k} \prod_{j=1}^{|V|} (\theta_i \beta_{ij})^{w_n^i})\, d\theta$$

Since the text collection is large and requires fast inferencing variational approach has been utilized in approximating the likelihood in equation 2.3.2 and the equation is re written as below:

$$\log p(w;\alpha,\beta) = \log \int_\theta \sum_z p(w|z;\beta)p(z|\theta)p(\theta;\alpha) \frac{q(\theta,z;\gamma,\emptyset)}{q(\theta,z;\gamma,\emptyset)} d\theta$$

$$\geq E_q[\log p(w|z;\beta) + \log p(z|\theta) + \log p(\theta;\alpha) - \log q(\theta,z;\gamma,\emptyset)]$$

Where $q(\theta,z;\gamma,\emptyset)$ is fully factorized variational distribution

$q(\theta;\gamma)$ is Dirichlet and

$q(z_n;\emptyset_n)$ is Multinomial

To obtain the best approximation $p(w;\alpha,\beta)$, bounds of γ *and* $\emptyset$ can be maximized. Since the Dirichlet distribution is not computable nor differentiable, (k-1) dimensional integral over θ which is computable is used. The two below equations will be used further:

$$\emptyset_{ni} \propto \beta_{iw_n} e^{(\varphi(\gamma_i) - \varphi\,(\sum_{j=1}^{k} \gamma_j))}$$

$$\gamma_i = \alpha_i + \sum_{n=1}^{N} \emptyset_{ni}$$

Where φ is the first derivative of the log τ function and the resulting variational parameters can also be used to interpret as an approximation of the parameters of the true posterior. By variating Estimation and maximizing the lower bound on log likelihood of the Estimation Maximization (EM) Algorithm for the collection of documents D = {w1, w2 . . ., wM}. The equation 2.3.3 can be re written as below:

$$\log p(D) \geq \sum_{m=1}^{M} E_{q_m}[logp(\theta, z, w)] - E_{q_m}[logq_m(\theta, z)]$$

The Maximization step of EM algorithm on the above equation can be calculated with the help of the M step update equation as below:

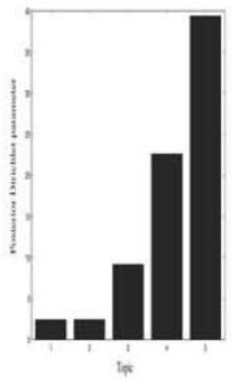
$$\beta_{ij} \propto \sum_{m=1}^{M} \sum_{n=1}^{|w_m|} \emptyset_{mni} w_{mn}^{j}$$

Applying Newton-Raphson

$$\frac{\partial l}{\partial \alpha_i} = \sum_{m=1}^{M} (\varphi(\sum_{j=1}^{k} \alpha_j) - \varphi(\alpha_i)) + (\varphi(\gamma_{mi}) - \varphi(\sum_{j=1}^{k} \gamma_{mj}))$$

Experiments with LDA

LDA was tested using 2 corpora’s (1) TREC AP corpus that consisting 2500 news articles and 37,871 words. And (2) is the CRAN corpus, consisting of 1400 abstracts and 7747 words. After examining the posterior distribution on the topic mixture, topics that contributes most words to a particular document. Examining these multinomials gives more information about these topics.



Topic 1	Topic 2	Topic 3	Topic 4	Topic 5
SCHOOL	MILLION	SAID	SAID	SAID
SAID	YEAR	AIDS	NEW	NEW
STUDENTS	SAID	HEALTH	PRESIDENT	MUSIC
BOARD	SALES	DISEASE	CHIEF	YEAR
SCHOOLS	BILLION	VIRUS	CHAIRMAN	THEATER
STUDENT	TOTAL	CHILDREN	EXECUTIVE	MUSICAL
TEACHER	SHARE	BLOOD	VICE	BAND
POLICE	EARNINGS	PATIENTS	YEARS	PLAY
PROGRAM	PROFIT	TREATMENT	COMPANY	WON
TEACHERS	QUARTER	STUDY	YORK	TWO
MEMBERS	ORDERS	IMMUNE	SCHOOL	AVAILABLE
YEAROLD	LAST	CANCER	TWO	AWARD
GANG	DEC	PEOPLE	TODAY	OPERA
DEPARTMENT	REVENUE	PERCENT	COLUMBIA	BEST

Figure 5: The Dirichlet Parameters (on the left) where γ_i>0 and top 15 words from the respective topic.

To compare the performance of Latent Dirichlet Allocation with other models, perplexity is calculated for the datasets, AP and CRAN corpora. For any M number of documents, perplexity (D_{test}) can be calculated as:

$$\exp\,(-\sum_{m} logp(w_m)/\sum_{m}|w_m|)$$

CHAPTER III: GATHERING THE DATA

Now, our project to do the comparison for the two approaches the data required is very large. So, I choose to work with one of the most popular social networking sites, Twitter. Getting Tweet data is very easy and very understandable to most of the audience. Twitter data have five data objects that comprises of different fields:

- **Tweet** has the fields id, text, attachments, author_id, created_at, entities, geo, in_reply_to_user_id, lang, possibly_sensitive, referenced_tweets, source, public_metrics, withheld.

- **Media** has the fields media_key, type, duration_ms, height, preview_image_url, url, width.

- **Poll** has the fields id, options, duration_minutes, end_datetime, voting_status.

- **Place** has the fields id, full_name, contained_within, country, country_code, geo, name, place_type.

- **User** has the fields id, name, username, created_at, description, entities, location, pinned_tweet_id, profile_image_url, protected, url, verified, withheld.

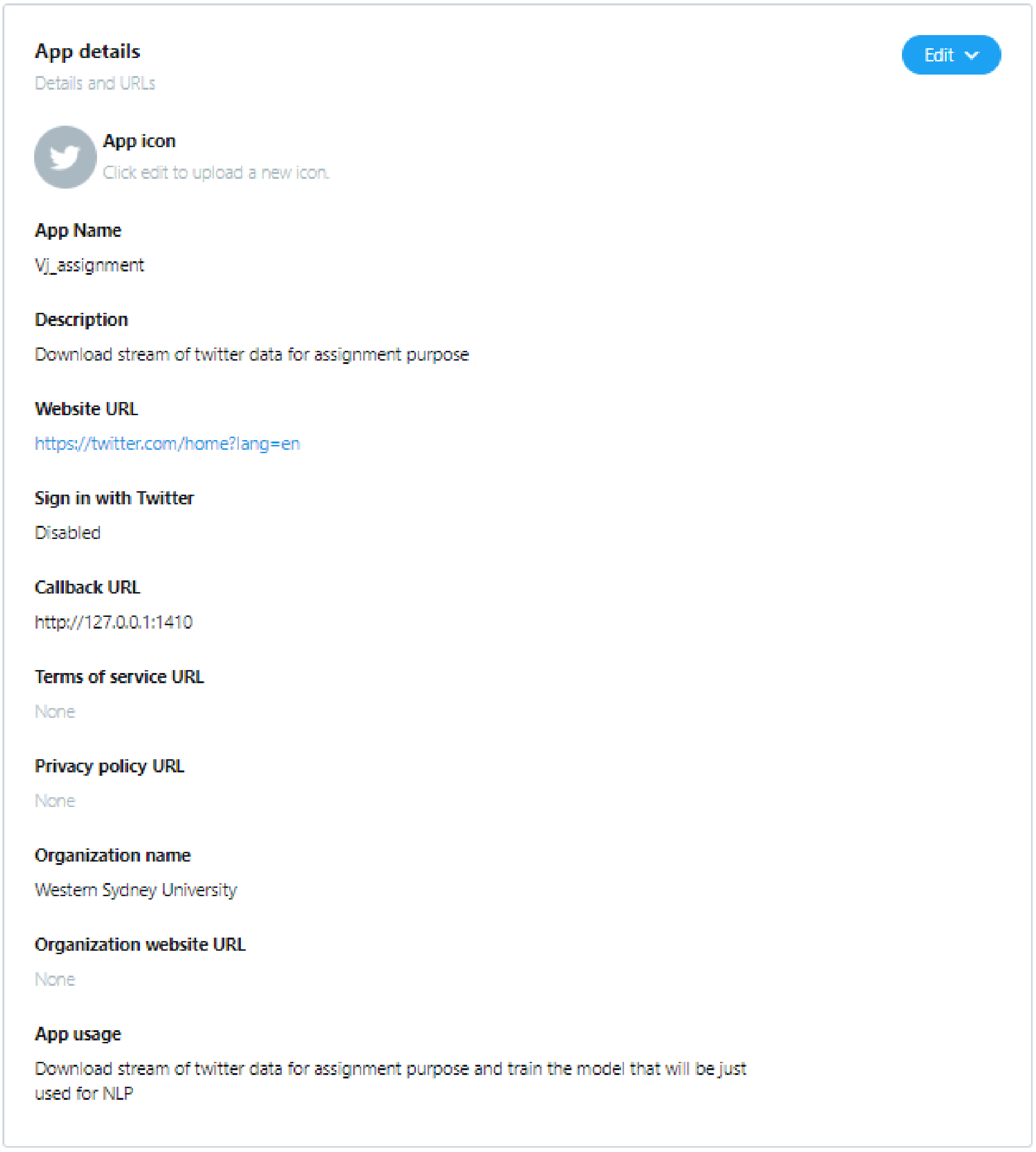
For the trained models, first we need to divide the twitter data into 2 parts as training data set and testing dataset. Training data set is used to train the model we are going to use, and testing data set is used to get the output. Whereas for random ecostate we just run the testing dataset through the random ecostate to obtain the results.

Python offers tweepy to easily access twitter API. This API provides access to entire twitter restful API methods. Tweepy supports both authentication processes such as (1) application-user that uses ‘tweepy.OAuthHandler()’ and (2) application-only that uses ‘tweepy.AppAuthHandler()’ but for our purpose we will use ‘tweepy.OAuthHandler()’ twitter authentication process. A twitter Application has been created that will allow us to download the data from twitter. The application details are as shown in figure 6. Once the application has been created we now have the details required for authentication such as consumer key, consumer secret, access token, access token secret. Using these details we can authorise the twitter account to download the data.

The data downloaded is in json format and has a lot of unwanted fields and noise that are not essential for our analysis. For our analysis we are only interested in tweet text so all the tweet texts has been extracted from twitter data and saved separately in a text file. Still the tweet text is not ready for analysis as the tweets contain a lot of data that cannot be interpreted by the Natural language processors.

Cleaning the Data

Most of the tweets contains data that is unclear to NLP’s as the tweets contains emoticons, emojis, numbers, hashtags, web addresses etc., To deal with this unwanted data we have used regular expressions. Regular Expression is used to check if a particular string matches with a given expression and can be included or excluded for the data based on the requirement. Using regular expression, we have deleted all the emoticons, emojis, numbers, hashtags, web addresses etc., In order to make it easy for the system to undersand the words we have also converted all the capital letters to small letters. Now the data is finally in form which we can analyse.



App Name
Vj_assignment

Description
Download stream of twitter data for assignment purpose

Website URL
https://twitter.com/home?lang=en

Sign in with Twitter
Disabled

Callback URL
http://127.0.0.1:1410

Terms of service URL
None

Privacy policy URL
None

Organization name
Western Sydney University

Organization website URL
None

App usage
Download stream of twitter data for assignment purpose and train the model that will be just used for NLP

Figure 6: Application in twitter

CHAPTER IV: MODELS FOR WORD EMBEDDINGS

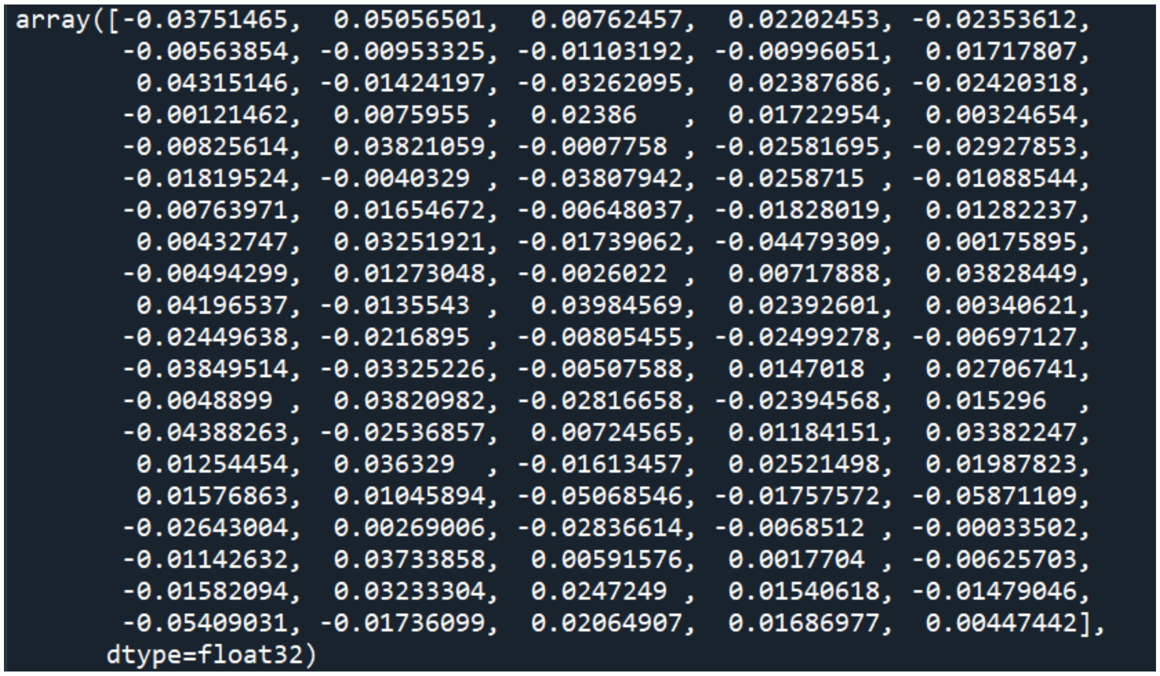
word embedding is the technique of converting the words to high dimensional vectors, where all the words with similar meaning stays close to each other in the vector space. According to this approach words can have multiple degrees of similarity. Results obtained by Learning high dimensional embeddings from a large data are the most accurate. In this paper we concentrate on two different approaches/models (1) Trained models and (2) Random (Ecostate) Model.

4.1 Trained model

Word2Vec is a word embedding technique used to generate vectors from given words. Unlike humans, machines can only understand numbers. The words need to be represented in numeric format and Word2Vec does exactly that. Using deep learning and network based techniques, Word2Vec converts the words into vectors in such a way that the words with similar meaning are close to each other in multi-dimensional vector space. The classic example of perfect implementation of Word2Vec is that when a man is subtracted from a king and women is added gives queen.

King – Man + Women = Queen

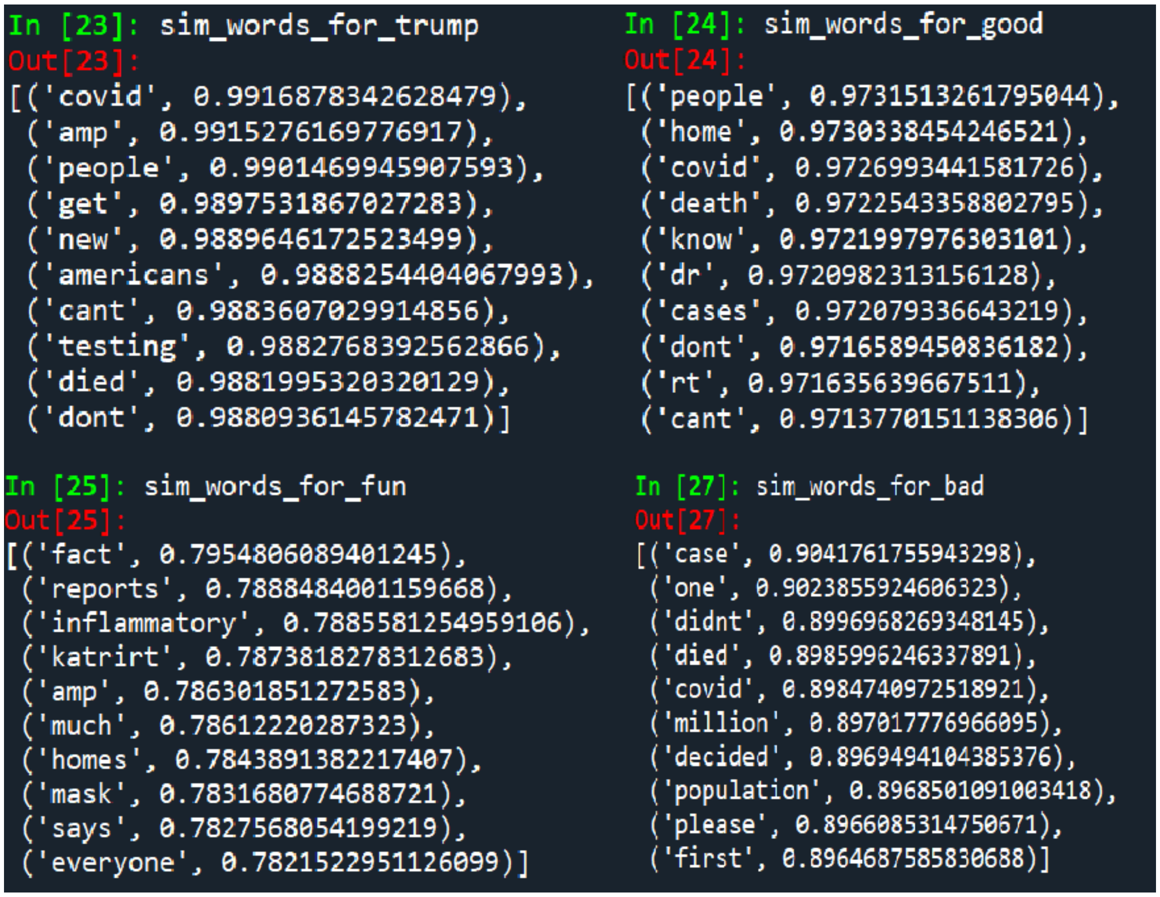
As already mentioned, Word2Vec model is implemented on Python with the help of Python's Gensim library. The data collected and cleaned will now be used to train the Word2Vec model. The clean tweet text data is now tokenized to individual words as the input to the Word2Vec model is words. The list of words collected and cleaned from twitter is passed to the Word2Vec model of Genism package. The Word2Vec model has a min_count parameter that allows you to select the words that needs to be included for the conversion based on the number of times the word has appeared in the text corpus. Now we have our own Word2Vec model and figure 7 shows the vector space in which the word "Trump" stored in.



```
array([-0.03751465,  0.05056501,  0.00762457,  0.02202453, -0.02353612,
       -0.00563854, -0.00953325, -0.01103192, -0.00996051,  0.01717807,
        0.04315146, -0.01424197, -0.03262095,  0.02387686, -0.02420318,
       -0.00121462,  0.0075955 ,  0.02386   ,  0.01722954,  0.00324654,
       -0.00825614,  0.03821059, -0.0007758 , -0.02581695, -0.02927853,
       -0.01819524, -0.0040329 , -0.03807942, -0.0258715 , -0.01088544,
       -0.00763971,  0.01654672, -0.00648037, -0.01828019,  0.01282237,
        0.00432747,  0.03251921, -0.01739062, -0.04479309,  0.00175895,
       -0.00494299,  0.01273048, -0.0026022 ,  0.00717888,  0.03828449,
        0.04196537, -0.0135543 ,  0.03984569,  0.02392601,  0.00340621,
       -0.02449638, -0.0216895 , -0.00805455, -0.02499278, -0.00697127,
       -0.03849514, -0.03325226, -0.00507588,  0.0147018 ,  0.02706741,
       -0.0048899 ,  0.03820982, -0.02816658, -0.02394568,  0.015296  ,
       -0.04388263, -0.02536857,  0.00724565,  0.01184151,  0.03382247,
        0.01254454,  0.036329  , -0.01613457,  0.02521498,  0.01987823,
        0.01576863,  0.01045894, -0.05068546, -0.01757572, -0.05871109,
       -0.02643004,  0.00269006, -0.02836614, -0.0068512 , -0.00033502,
       -0.01142632,  0.03733858,  0.00591576,  0.0017704 , -0.00625703,
       -0.01582094,  0.03233304,  0.0247249 ,  0.01540618, -0.01479046,
       -0.05409031, -0.01736099,  0.02064907,  0.01686977,  0.00447442],
      dtype=float32)
```

Figure 7: Vector space in which the word "trump" is stored in

The similarity of the few words has been provided below in Figure 8. We can see that the results produced are not as expected. This is because the data set chosen is not big enough. So, it is clear that we have to choose the data that's big enough to train the model in order to produce expected results.



```
In [23]: sim_words_for_trump
Out[23]:
[('covid', 0.9916878342628479),
 ('amp', 0.9915276169776917),
 ('people', 0.9901469945907593),
 ('get', 0.9897531867027283),
 ('new', 0.9889646172523499),
 ('americans', 0.9888254404067993),
 ('cant', 0.9883607029914856),
 ('testing', 0.9882768392562866),
 ('died', 0.9881995320320129),
 ('dont', 0.9880936145782471)]
```

```
In [24]: sim_words_for_good
Out[24]:
[('people', 0.9731513261795044),
 ('home', 0.9730338454246521),
 ('covid', 0.9726993441581726),
 ('death', 0.9722543358802795),
 ('know', 0.9721997976303101),
 ('dr', 0.9720982313156128),
 ('cases', 0.972079336643219),
 ('dont', 0.9716589450836182),
 ('rt', 0.971635639667511),
 ('cant', 0.9713770151138306)]
```

```
In [25]: sim_words_for_fun
Out[25]:
[('fact', 0.7954806089401245),
 ('reports', 0.7888484001159668),
 ('inflammatory', 0.7885581254959106),
 ('katrirt', 0.7873818278312683),
 ('amp', 0.786301851272583),
 ('much', 0.78612220287323),
 ('homes', 0.7843891382217407),
 ('mask', 0.7831680774688721),
 ('says', 0.7827568054199219),
 ('everyone', 0.7821522951126099)]
```

```
In [27]: sim_words_for_bad
Out[27]:
[('case', 0.904176175594329 8),
 ('one', 0.902385592460632 3),
 ('didnt', 0.8996968269348145),
 ('died', 0.8985996246337891),
 ('covid', 0.8984740972518921),
 ('million', 0.897017776966095),
 ('decided', 0.8969494104385376),
 ('population', 0.8968501091003418),
 ('please', 0.8966085314750671),
 ('first', 0.8964687585830688)]
```

Figure 8: Similar words for trump, good, fun and bad

4.2 Random Ecostate models

Trained models provide the best results but the scalability of the training method requires a lot of time and introduces scalability issues. To avoid these challenges the Random Ecostate approach can be followed which may provide similar results while using a very less data as compared to the trained model approach. Random ecostate networks are designed for sequence prediction problems. We can diverge the Ecostate state Network from sequence prediction to sentence/word prediction. Since no training is required in this technique, we can directly get the similarities of a particular word.

CHAPTER V: ACCURACY TEST FOR THE TRAINED AND RANDOM MODELS

To test the accuracy of the two approaches, trained model and Random Model we use an evaluation Toolkit for Universal Sentence Representations called SentEval. SentEval performs various tests as shown in the below Table 1. All the sentences manually classified with values from 1 to 5.

Table 1: Various Tests performed in SentEval

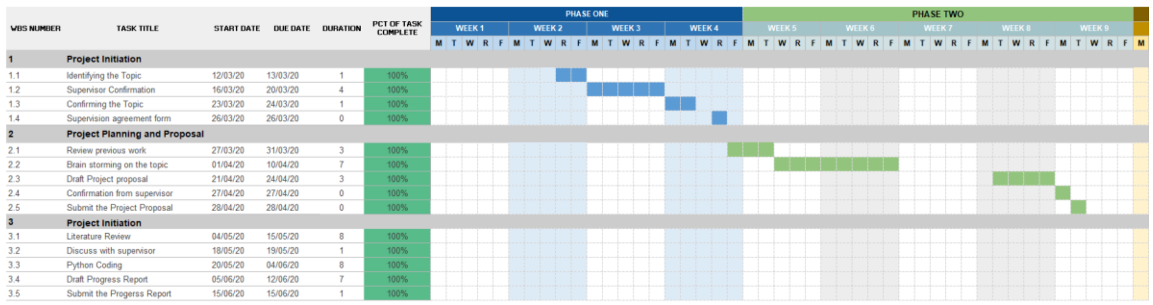
Name	Task
MR	Sentiment(Movies)
CR	Product Reviews
SUBJ	Subjectivity/Objectivity
MPQA	Opinion Polarity
TREC	Question-Type
SST-2	Sentiment(Movies)
SST-5	Sentiment(Movies)

But since our output is represented in vectors for all the words, we will have to convert the words stored in vectors to a meaningful sentence that we have used in SentEval. Then the outputs of both the techniques can be compared with the sentences in SentEval to get the accuracy of both the techniques.

After the comparison of both the techniques I expect that the trained models will be more accurate. If the accuracy of Randomly assigned model (Ecostate) is close to what we have from trained models it would be very efficient option to choose Randomly assigned model for word embeddings.

CHAPTER VI: RESEARCH PLAN

The plan for the current session is shown in below Gantt Chart:



WBS NUMBER	TASK TITLE	START DATE	DUE DATE	DURATION	PCT OF TASK COMPLETE
1	**Project Initiation**				
1.1	Identifying the Topic	12/03/20	13/03/20	1	100%
1.2	Supervisor Confirmation	16/03/20	20/03/20	4	100%
1.3	Confirming the Topic	23/03/20	24/03/20	1	100%
1.4	Supervision agreement form	26/03/20	26/03/20	0	100%
2	**Project Planning and Proposal**				
2.1	Review previous work	27/03/20	31/03/20	3	100%
2.2	Brain storming on the topic	01/04/20	10/04/20	7	100%
2.3	Draft Project proposal	21/04/20	24/04/20	3	100%
2.4	Confirmation from supervisor	27/04/20	27/04/20	0	100%
2.5	Submit the Project Proposal	28/04/20	28/04/20	0	100%
3	**Project Initiation**				
3.1	Literature Review	04/05/20	15/05/20	8	100%
3.2	Discuss with supervisor	18/05/20	19/05/20	1	100%
3.3	Python Coding	20/05/20	04/06/20	8	100%
3.4	Draft Progress Report	05/06/20	12/06/20	7	100%
3.5	Submit the Progerss Report	15/06/20	15/06/20	1	100%

Figure 9: Gantt chart for research plan

In the current session as an analysis, small twitter data has been downloaded and model has been trained using the same data, but the results are not as expected because of insufficient data. In the next session I will collect large amount of data from twitter and train the model again. Once we have the trained model, I will have to test the results from the Random Ecostate model and the accuracy of both using SentEval on Python.

REFERENCES

Achlioptas, D., 2001, May. Database-friendly random projections. In Proceedings of the twentieth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems (pp. 274-281).

Blei, D.M., Ng, A.Y. and Jordan, M.I., 2002. Latent dirichlet allocation. In Advances in neural information processing systems (pp. 601-608).

Blei, D.M., Ng, A.Y. and Jordan, M.I., 2003. Latent dirichlet allocation. Journal of machine Learning research, 3(Jan), pp.993-1022.

Conneau, A. and Kiela, D., 2018. Senteval: An evaluation toolkit for universal sentence representations. arXiv preprint arXiv:1803.05449.

Hofmann, T., 2013. Probabilistic latent semantic analysis. arXiv preprint arXiv:1301.6705.

Landauer, T.K., Foltz, P.W. and Laham, D., 1998. An introduction to latent semantic analysis. Discourse processes, 25(2-3), pp.259-284.

Landauer, T.K. and Dumais, S.T., 1997. A solution to Plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. Psychological review, 104(2), p.211.

Li, P., Hastie, T.J. and Church, K.W., 2006, August. Very sparse random projections. In Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 287-296).

Malik, U. (2019). Implementing Word2Vec with Gensim Library in Python. [online] Stack Abuse. Available at: https://stackabuse.com/implementing-word2vec-with-gensim-library-in-python/ [Accessed 15 Jun. 2020].

radimrehurek.com. (2019). gensim: topic modelling for humans. [online] Available at: https://radimrehurek.com/gensim/models/word2vec.html [Accessed 15 Jun. 2020].

Wieting, J. and Kiela, D., 2019. No training required: Exploring random encoders for sentence classification. arXiv preprint arXiv:1901.10444.

APPENDIX: PYTHON CODE USED FOR THE ANALYSIS

```
####################Downloading the data from twitter#####################

#pip install tweepy==3.8.0

import tweepy
#from tweepy import OAuthHandler
import json
from tweepy import Stream
from tweepy.streaming import StreamListener
import re

consumer_key='492T056Vq4TeUnDKDTs01amTC'
consumer_secret='0gDOGaHpVWlFxEYhQxCcSoeAb5ntonIlGOLqyyum1NTP3CtazY'
access_token='1384285812-QvXHCDqfFd5kRTbtpEx7SlGRgvS6mCmqF5GwWNi'
access_token_secret='RhOz3iGtQujSVFrO7PQN2ZIDOqm8LFh2hMz8bOMmO9by9'


auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_token, access_token_secret)
api = tweepy.API(auth)

searchTerm = input("Enter topic: ")
limit = int(input("Enter the maximum number of tweets: "))

tweets = tweepy.Cursor(api.search,q=searchTerm,count=limit, lang="en",
tweet_mode='extended').items(limit)
f3 = open('test.txt', 'w', encoding="utf-8")
with open('tweets.json', 'w', encoding='utf8') as file:
    for tweet in tweets:
        data = tweet._json['full_text']
        print(data)

        f3.write(data)

f3.close()
```

```
#####################Pre-processing the downloaded data#####################

import nltk
import string
import re
from nltk.corpus import stopwords
nltk.download('punkt')


with open('C:/Users/vijju/vj.txt', 'r+', encoding = 'utf-8') as fh:
    data = fh.read()

    fh.close()

data2 = ''.join(re.sub("(@[A-Za-z0-9]+)|([^0-9A-Za-z \t])|(\w+:\/\/\S+)","",data))

data2 = data2.lower()

data2 = re.sub(r'\d+', '', data2)

stop_words = set(stopwords.words('english'))
from nltk.tokenize import word_tokenize
tokens = word_tokenize(data2)
result = [i for i in tokens if not i in stop_words]


from gensim.models import Word2Vec
word2vec = Word2Vec(result, min_count=3)

vocabulary = word2vec.wv.vocab
print(vocabulary)

v1 = word2vec.wv['covid']
```

```
#################### Training the model using Word2Vec #####################

import nltk
import string
import re
from nltk.corpus import stopwords
#nltk.download('punkt')

from gensim.models import Word2Vec


with open('C:/Users/vijju/vj.txt', 'r+', encoding = 'utf-8') as fh:
    data = fh.read()

    fh.close()

data2 = ''.join(re.sub("(@[A-Za-z0-9]+)|([^0-9A-Za-z \t])|(\w+:\/\/\S+)","",data))

data2 = data2.lower()

data2 = re.sub(r'\d+', '', data2)

all_sentences = nltk.sent_tokenize(data2)
all_words = [nltk.word_tokenize(sent) for sent in all_sentences]


for i in range(len(all_words)):
    all_words[i] = [w for w in all_words[i] if w not in stopwords.words('english')]


my_w2v = Word2Vec(all_words, min_count=3)

vocabulary = my_w2v.wv.vocab
print(vocabulary)

v1 = my_w2v.wv['trump']

sim_words_for_trump = my_w2v.wv.most_similar('trump')
sim_words_for_good = my_w2v.wv.most_similar('good')
sim_words_for_fun = my_w2v.wv.most_similar('fun')
sim_words_for_bad = my_w2v.wv.most_similar('bad')
```