Leveraging the Combinatorial Nature of Networks and Language to Our Advantage

Ciara Gilsenan
Trinity College Dublin
gilsenci@tcd.ie

Ernests Kuzņecovs
Trinity College Dublin
kuznecoe@tcd.ie

Kalaivani Kandasamy
Trinity College Dublin
kandasak@tcd.ie

Yash Shukla
Trinity College Dublin
yshukla@tcd.ie

Ayush Kalra
Trinity College Dublin
kalraay@tcd.ie

Abstract

A large amount of real time data is generated daily by a plethora of social media networks such as Reddit, Twitter, Facebook, etc. A lot of research has been carried out to discover correlations, patterns and similarities in this raw data. A dataset of Twitter users is used for this research and was constructed using the Twitter API. The data is then processed and represented as a network of nodes. Most social media networks have symbolic links between the nodes that can be retrieved as data, for example, followers of a user on Twitter. As well as this, each node in the network usually have a plethora of textual data, for example, a Twitter users history of posts. These symbolic links between the nodes can be harvested as data points and can be used as a base to experiment textual comparison methods between the nodes, such as constructing word vectors for each node and finding the cosine similarity between them. Though if one were to use the symbolic links and text to do more advanced processing, soon they would be troubled by the combinatorial and computationally tricky nature of the two domains, for example, the number of edges grows by a

$$factor = \frac{n(1-n)}{2}$$

and the n-gram tokens to include also grows exponentially. This paper presents a method to leverage this nature to our favour, to feed the data-hungry neural networks, by using stochastic gradient descent to batch both, edge, and words of word vectors.

Keywords— Networks, Social Media, Textual Similarities, Cosine Similarity, Graph Representations, Stochastic Gradient Descent

1 Introduction

Social Media Networks are formed by a network of individuals who are connected with each other on a platform such as Facebook, Twitter, WhatsApp, etc. In the modern world, social media networks have shown a drastic change in human interactions and exchange of information, and allow individuals to stay connected with their friends, family, customers and colleagues. These social networks have become so useful in society that there is now so much research done and still do to in this area. The real time textual data that they offer can be used to test many hypotheses such as predicting personalities (Golbeck et al., 2011) and measuring correlations between the textual features and various life events (Seraj et al., 2021).

The hypothesis to be tested in this research project is to see if there is any textual similarity between two people in a social media network who follow each other or are friends with each other. This can suggest that if two people in a network know each other, then there is a chance they can be linked based on the textual features in their tweets. There can be many applications for this such as suggesting new friends/followers in different social media networks. This will be tested by constructing a graph representation of a social media network, where two nodes are connected if one or both follows the other or are friends with the other in the network. The textual similarity of two connected nodes are then measured using various textual features and similarity scores upon the

nodes WordVectors such as, *Cosine Similarity*, and *Euclidean Distance*, the symbolically connected edge, and the textually connected edge can then be compared to evaluate the performance of the textual similarity measure.

Furthermore, using the symbolic links of the network gives us access to a great deal of "labelled" data, which can be exploited for improving the similarity measure. In this work, a weight is applied to each word in the wordvector, and this weight is assigned a value based on the "ground truth" of the symbolic edges. This is done through stochastic gradient descent.

Twitter was the platform chosen for this research project. It is a micro-blogging site where interactions happen through messages called tweets. Twitter has an active 396.5 million users as of January, 2022. The link between two individuals can be friendship, follower or relationship. The relationship that exists between individuals in the network can be used in many ways. After some investigation it was concluded that there was no open-source corpus available that suited the hypothesis testing, therefore a dataset was manually constructed.

It was decided that two datasets would be constructed for this research in order to have some variance with the data gathering. The first dataset contains users that are directly linked through the following/follower relationship. The second dataset users that are disjointed and therefore there is no follower/following relationship between them. It is expected that the second dataset will result in a more sparsely connected graph than the first dataset. Additional information on the format and construction of this dataset is described in Section 4.2.

The amount of data generated every minute in social media is very huge. This made a need for using advanced techniques to process large volume of data. Considering the data volume, richness and amount of information inferred from the data generated in social media, graphical representation of data is becoming as an active research topic in data mining Samatova et al. (2013). Graphs are chosen for representing our data set most importantly for the reasons that they are considered as an easy and natural way to represent the connections among entities and its structural simplicity. Graphs suits our need as our study is based on the connection between the Twitter users - followers and followings where each users can be represented as nodes and the relationship between them can be represented as edges. After collecting all the data, we need to process the tweets to collect information present in it. A most popular representation is n-grams which represents a continuous n tokens from the given text where n ranges from 1 to size of the sentence. Let's understand n-grams practically with the help of the following sentence: "I reside in Dublin" 1-gram or unigram - "I", "reside", "in", "Dublin"
2-gram or bigram - "I reside", "reside in", "in Dublin"
When the amount of text is large, the number of new tokens increases exponentially which implies our word vector will be huge and making the word vector computationally expensive. In our work we show how using Stochastic Gradient Descent(SGD) uses subsets of the unique tokens and perform computations on them.

This paper will discuss a literature review on related work in this area in Section 2, followed by a detailed description of implementing the hypothesis testing in Section 4, and a presentation of results and evaluation in Section 5 and Section 6 respectively.

2 Literature Review

There are many research articles and papers out there that investigate the similarity between users in social networks such as Twitter. A paper by Singh et al. (2016) describes the social network as a "social structure between people", and mentions the term *Social Network Analysis*, which looks into the relationship between people and/or groups. This paper investigates the textual similarity between people on Twitter, therefore this is very relevant to this project. The paper included some data preprocessing such as stopword removal and stemming. The textual similarity between two users in the social network is calculated using lexical features such as nouns, verbs, adverbs, etc. The similarity strength between to users is calculated using total words used and total common words used. These similarities were then entered into two different k-means clustering algorithms, and the results of the experiment show that the calculated similarity strength using lexical features can cluster users in a social network together. This paper shows how lexical features of text can be important when analysing the similarity between users of social networks.

Other research looks into similarity between social network users based on other features that are not just text. AlMahmoud and AlKhalifa (2018) introduces a framework that measures the similarity between twitter users based on 7 features including number of followers, retweets, hashtags, profiles, etc. In the paper they created a similarity formula that sums up the scores of each feature between two users. The formula is then transformed into a processing framework using the MapReduce model (Dean and Ghemawat, 2008). The paper explains thoroughly the contribution each feature makes to the similarity measurement, and how each feature can establish some sort of communication between two users, which can then be classed as a similarity. Some of the features in the paper are measured using some textual analysis, although these methods may not be similar to those that will be used in this project. For the *hashtag* feature, sentiment analysis is carried out to ensure that two users that use the same hashtag don't have conflicting opinions of it. The framework is then evaluated by comparing the results to Twitter's Who

To Follow service (Gupta et al., 2013), and the paper states that these results are fairly accurate. This paper shows that factors other than text can come into play when measuring similarity between social network users.

There can be various applications to this type of research, from follower recommendations to business recruitment. An interesting application written in an article by Rizio (2021) describes the use of textual similarity in tweets to find users with similar opinions on the COVID-19 pandemic which can then be linked to political views. For this they used the hashtags, text, location, and username of the tweets, and measured the similarity using *Cosine Similarity* and *TF-IDF*.

The research by Healey, Vogel, and Eshghi (Healey et al.) discusses about the group sub languages that evolve quickly among subgroups. This paper investigates on the language similarity among the individuals who are in same group or community. This research is relevant to us as we can also get the similarity of languages among the users who are in same network or follow each other. This research uses n-gram method with chi square test to get these similarities. It also states that dialogue mechanisms such as interactive alignment, grounding or local repair and clarification do not account for the patterns of similarity in language use, as measured by letter uni-grams.

To make a machine learning algorithm perform with high accuracy, optimization algorithms are used. The most popular optimization algorithm is gradient descent. Gradient descent algorithm fine tunes the parameters of the model after processing all the data points present in the training dataset. To generate a graph with n number of nodes, 2^n amount of data must be processed in our system. As the number of nodes in the graph increases, the amount of data to be processed will also increase. When all the data is processed the algorithm will converge at a very slow rate and it results in the gradient descent algorithm performing at a slower pace. Therefore an alternative optimization algorithm that suits our dataset size more efficiently and to reduce the computational cost had to be chosen. Stochastic gradient descent algorithm is found to be efficient in handling large data sets (Zhang, 2004). It is an approximation of the gradient descent algorithm where it replaces the entire data set with a random subset of the data (GeeksforGeeks, 2021). In general it confers a decrease in training time without sacrificing the model accuracy (Tsuruoka et al., 2009).

Grandjean (2016) represents Twitter users in graph model to identify digital humanities community. In this work they have analyzed the type of users present in an linguistic community graph and structural overlap of different language users. This work not only studies about the structural representation but also the vertices in the network such as in-degree, out-degree, betweenness and eigenvector. With the graphical representation the information about the user such as position and their influence in the network is known.

3 Research Questions

Below is a list of research questions that are derived from this project:

- Investigate the different ways a ground truth graph network can be constructed given corpora and meta-data about it.
- Investigate different methods of comparing similarity between documents by comparing against some ground truth.
- Compare the different similarity measures against each other.
- Investigate metrics to measure overlap between graphs.
- Investigate a similarity measure that can be optimised to make two graphs maximally overlap.
- Investigate how large amount of data can be used in our favour.

4 Research Methods

4.1 Overview of Implementation

The various components of the investigation was implemented in Python. The dataset was also be read into memory using Python, and appropriate data-structures and Python functions were used to manipulate the data and perform operations in a modular way. The NLTK[1] library will be used to perform textual pre-processing on the corpus.

[1] `https://www.nltk.org/`

4.2 Dataset Construction

An ideal dataset for this research project would contain a list of users where each row would contain the user id along with a collection of their tweets, a list of followers and followings. After doing some initial research it became difficult to find an open source dataset in this format. Most of the available Twitter datasets would have rows of singular tweets with the user id, tweets from a singular user, tweets processed for sentiment analysis, or tweets related to a specific genre such as COVID, various TV shows, airlines, presidential elections, etc (iMerit, 2021). Therefore the appropriate choice was to manually construct the dataset using the Twitter API[2] and the Python library Tweepy[3]. Figure 1 shows the planned format of each row in the dataset. For this research two datasets were constructed, one for the Ground Truth and one for the Artificial Connections.

```
{userId: "1234", corpus: "", followers: set<userId>, followings: set<userId>}
```

Figure 1: Format for each row in dataset

The main difficulty when constructing this dataset was finding an initial list of random Twitter users since the Twitter API does not have such an endpoint available. The initial plan was to generate a list of random numbers between 111111111 and 999999999 as it was thought that the assignment of the Twitter user ids was done sequentially, therefore these random numbers were likely to match a user. However this method was found to be impractical as it was difficult to find numbers in such a large range that matched a user id, and when a user was matched there were other constraints to consider such as making sure the user tweeted in English and that the user had enough tweets to build a corpus out of. If the user had less than 60 - 100 tweets then that would not be useful. Twitter also in recent years changed their ID formation from a 32-bit sequential assignment because of the exponential growth of the number of users. Twitter now assigns the user ID as a 64-bit unsigned integer made up of a timestamp, a worker number and a sequence number[4].

A second and more practical solution was instead used for the dataset construction. This plan was to obtain a list of random tweets using the *Cursor* object and the *Search* API endpoint from Tweepy. This is similar to using the Search functionality in the Twitter app. With this method the language of the tweets could be specified. A query string had to be specified for this API call therefore a topical one that was being tweeted about very often at the time was chosen. The chosen query string was “Ukraine”. Figure 2 shows the snippet of code that obtains a list of these tweets. The API call returns a list of tweet objects and a property of each of these objects is the user ID, therefore a list of random user IDs could be extracted from these random tweets. Once the list of user IDs was extracted, the datasets could then be built.

```
# search for tweets by keyword, then get users
search_tweets = tweepy.Cursor(api.search, q='ukraine', lang='en').items(80)
```

Figure 2: Obtaining list of random tweets based on query string.

4.2.1 Dataset 1

The first dataset was built by iterating through each ID in the initial list and obtain a list of the user’s tweets using the *user_timeline* endpoint. 60 - 100 tweets were collected for each user. This number was altered often as there is a limit of tweets that one can extract at a time from the Twitter API. The user’s corpus was then built by concatenating each of the extracted tweets. The followers and followings of each user were then collected using the *followers* and *friends* endpoints respectively. 5 - 10 of followers and followings were collected for each user. Each user’s data was then put into a JSON object shown in Figure 1 and was appended to the dataset which was also in JSON format. At this point, the dataset does not contain any links between users, unless that was the case coincidentally.

The next step was to iterate through each user in the dataset, get the followers and followings list in the JSON and repeat the process described above of obtaining the tweets, followers and followings for each user in the followers/followings list and appending to the dataset. This can then provide direct links in the dataset which will

[2] https://developer.twitter.com/en/docs/twitter-api
[3] https://www.tweepy.org/
[4] https://developer.twitter.com/en/docs/twitter-ids

be used when constructing the network graphs for the hypothesis testing. This was done iteratively through the dataset and it was also done randomly by initialising a list of random indices and adding the followers/followings of the user at that index to the dataset. Of course this process could not be executed for each member of the dataset as then it would never stop being build, therefore the construction ended when there was approximately 300 users as that was thought to be an appropriate amount for the hypothesis testing. This was also due to time constraints as Twitter only allows a certain amounts of tweets to be extracted at a time and the code had to be re-run every 15 minutes. Figure 3 shows a snippet of the constructed dataset.

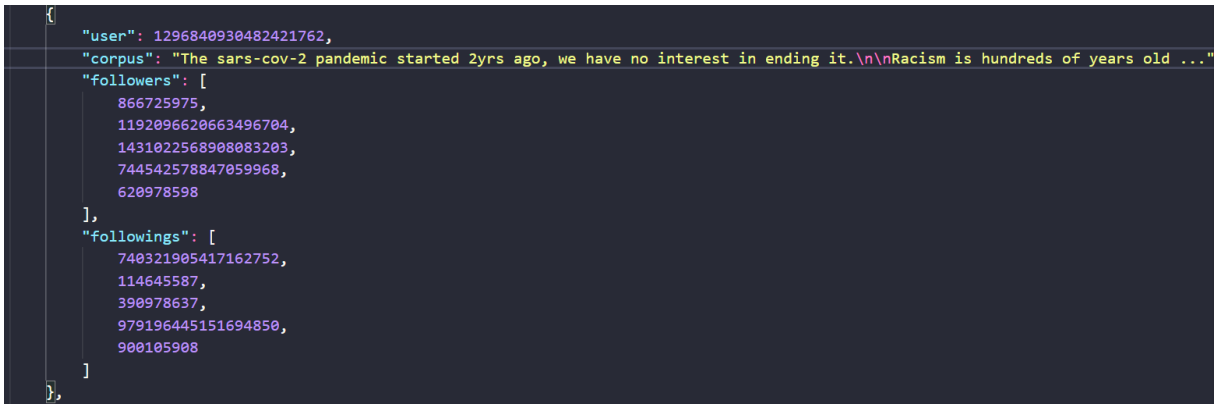

Figure 3: Example row in JSON dataset.

4.2.2 Dataset 2

This second dataset is contains disjointed Twitter users who have all tweeted about a specific topic. It was constructed similarly to Dataset 1 as seen in Section 4.2.1 and each row has the same format as seen in Figure 1. The main difference with this dataset is that the subsequent followers and followings of the users in the dataset are not added, therefore there are no links present between them. When constructing this dataset the chosen query string used in the Tweepy *Search* endpoint came from a list of topics in the area of computer science. Table 1 shows the list of these query strings used in the construction. The dataset is also in JSON format as seen in Figure 3.

Query Strings	
Machine Learning	Java
Python Programming	Javascript
Web development	JQuery
Node.js	Golang
Functional Programming	Dynamic Programming
NumPy	Software Engineering
App Development	Python Flask

Table 1: Query Strings for Artificial Connections Dataset

4.3 Constructing Network Graphs

4.3.1 Overview

Figure 4 and Figure 5 illustrates the network construction of the Twitter data for hypothesis testing. Further information on this is discussed in Section 4.3.3 and Section 4.3.4.

The data from the twitter

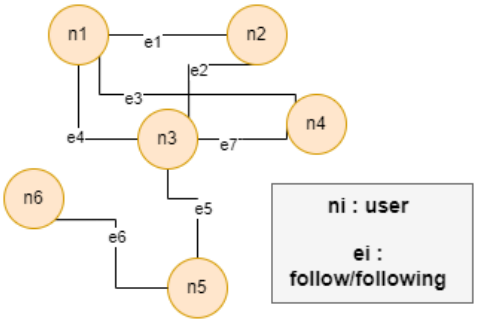

Figure 4: Graph 1 - Ground Truth

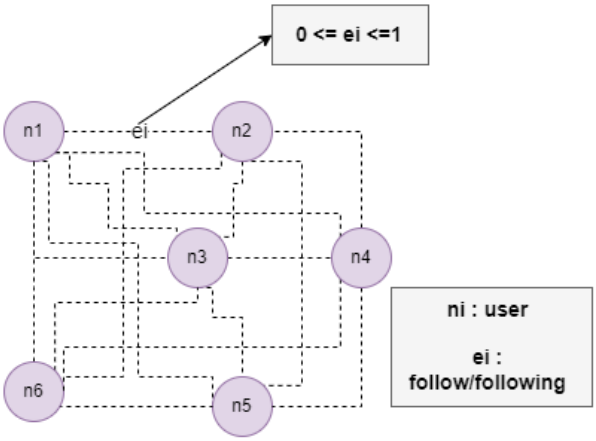

Figure 5: Graph 2 - Artificial Connections

- A Graph 1 will be constructed using the symbolic links in the data as ground truth.
- Graph 2 will be artificially constructed using a function f applied to all pairs of users corpora.
- Then Graph 1 and Graph 2 will be compared and the difference will be observed.

4.3.2 Data Preprocessing

The dataset was stored in a JSON format. The JSON file contained data in key-value pairs, and the primary data related to tweets was contained in the corpus. On analyzing the dataset, it was found that apart from the user tweet, the dataset also contained much redundant information such as emojis, punctuations, links, usernames, and hashtags. Apart from this, some of the tweets from users contained text from other languages. To make the data in a useful format pre-processing was required.

Initially, before pre-processing the dataset, the removal of ASCII values and text from other languages was a tough challenge that was overcome by using the Unicode library of python. To remove the emojis from the corpus, an array containing the ASCII values of the emojis is used. Apart from that, to remove other things such as links punctuations and usernames, the ”sub” function of string was used. The data still contained some redundant information like hashtags. Initially, all the hashtags were removed along with the word, but it was later realized that the hashtags also contained critical words which play a vital role in the formation of tokens. All the pre-processed data was converted into lowercase to maintain the consistency of the text. To extract essential words from the corpus, tokenization was performed to separate each word by commas. Later, The Python Natural Language Toolkit (nltk) library removed the list of stopwords.

To get some insights out of the data we also checked the whole preprocessed corpus and retrieved unique tokens from it. After getting the unique tokens we calculated the probability of the occurence of each unique token for a user.

4.3.3 Graph 1: Ground Truth

- In this graph, each node represents a twitter user.
- Each edge represents a connection between the users.
- We consider an edge of strength 1 between two nodes if there exists a follow in both directions. A strength of 0.5 if there exists only one follow between the two nodes, and a strength of 0 if there is no symbolic link.

4.3.4 Graph 2: Artificial Connections

- In this graph, the nodes are the same as in Graph 1.
- Each edge between 2 nodes in this graph is constructed by taking the first nodes corpus and comparing the second nodes corpus.
  - A nodes corpus is given by concatenating a users twitter posts into one document.

  - We use a function f to compute the strength of an edge.

- Multiple f's we can use.
  - f is a function that takes 2 different documents and computes a similarity score, the similarity score is a number between 0 and 1.

4.4 Possible f Comparison Functions

1. f transforms the input documents into word vectors of word counts, then computes the cosine similarity between them.
2. f transforms the input documents into word vectors of TF-IDF scores, then compute the cosine similarity between them.
3. f transforms the input document into word vectors with n-grams of counts/TF-IDF scores, then compute the cosine similarity between them.

- A possible pre-processing step applied to the corpus would be to eliminate the tokens that occur a small number of times to remove the size of the word vectors.

4.5 Comparisons Between Graph 1 and Graph 2

- A graph can represented as a vector, where each index represents an edge, and the value is the edge strength.
- Given graph vectors Graph1, Graph2, a cosine similarity can be computed between them to give back the degree of overlap between the graphs.

4.6 Improving Comparison Function f

- The ground truth of Graph1 can be used to train a comparison function f to achieve higher graph overlap.
- For each word count, wc_i in the wordvector, a weight x_i is attached to the word count, giving $wc_i * x_i$. An ordinary word count wordvector can be thought of as each x_i being set to 1. x_i should be adjusted such that when the similarity is applied to all the edges in the graph, it gives the maximum similarity between Graph1 and Graph2.
- An x_i of 0 would denote that the word counts have no significance in determining if 2 nodes are symbolically connected.
- Using ordinary gradient descent to optimise such a function would require doing an exponentially large vector multiplications i.e For each pair of nodes, compute the cosine similarity of their wordvectors, and sum them up. And this is just for a single iteration to update the weights once, whereas perhaps a lot more than 1000 may be needed.
- A modified stochastic gradient descent is used instead, with the premise being that only a subset of the edges *and* words are used to compute overlap between ground truth. The following subsection explains the details of this subset batching.

4.7 Loss Function For SGD

- let M = Set of all edge indices in an edge vector
- let K = Set of all word vector indices in a word vector
- let m = Subset of M
- let k = Subset of K
- let b_m = size of m, i.e batch size of m
- let b_k = size of k, i.e batch size of k
- let GT_m = an edge vector of the ground truth values. i.e each index corresponds to an edge, and the value corresponds to the edges strength. Indexed by m.

- let $EF_{x,m,k}$ = an edge vector where values are the cosine similarity of the two word vectors associated with the edges, where x and the word vectors are indexed by k (giving us a subset of words in the word vectors), and edges are indexed by m (giving us a subset of the pairs of word vectors out of the total set of edges). And each word vector is weighted by x.
- let $f(x, m, k)$ = the similarity between $EF_{x,m,k}$ and GT_m. The code implementation is shown in 7, a5 corresponds to $EF_{x,m,k}$, and dataGT[m] corresponds to GT_m
- On each iteration of SGD, a new random set of indices k and m are chosen to be used in the function f, the gradients with respect to x for the function is calculation, and the weights are updated such that the function f becomes smaller. Implementation of SGD is shown in 6.
- For comparing the accuracy of the overlap in the loss function f, for SGD, mean squared error is used as cosine similarity would have difficulties having it's gradients computed.

5 Results

- Upon inspection of the datasests, dataset 1 was used to perform the textual analysis.
- The higher the cosine similarity, the higher the similarity between vectors.
- For the baseline similarity, the similarity between ground truth and various baseline graphs are listed in 10. The first listing is the graph constructed from the word count word vector cosine similarity between nodes. The mean one is the graph constructed by setting all edges to the mean value of the edges of the ground truth graph.
- The Adam variant of SGD is also used to inspect performance.
- 11 shows the similarity measure of using weighted word vectors trained by SGD and Adam with 100,000 iterations against the unweighted word vectors. We see that weighted word vectors of SGD performs better than unweighted word vectors. Even though Adam has a low mean squared error, it performs poorly in the cosine similarity, which, might be a more meaningful measure than mean squared error.
- 8, 9 shows the similarity measures during the training process, we can see Constant step SGD is continually climbing cosine similarity, but Adam does not. This could be the momentum aspect of Adam which is residually updating the weights of values which dont have a gradient for that batch on that iteration. We notice the cosine similarity is still increasing for SGD, which means it can become more accurate if it was ran for longer.
- 12, 13, 14, shows what the highest and lowest weighted words are. We can see that Adam is eager to give words a higher than 1 weight, while SGD Constant keeps the weights under 1. The bottom words are practically 0 as we'd expect, as there are words which don't have impact on the predictability of the ground truth connections.

```
def sgd(x0, alpha, iters, bm, bk, rngKey):
  x = x0
  for i in range(iters):
    key, rngKey = random.split(rngKey)
    m, k = indices(key, bm, bk)
    g = (grad(loss_mse)(x, m, k))
    x = x - alpha * g
  return x
```

Figure 6: SGD

```
@jit
def loss_mse(x, m, k):
  a1 = dataM.at[m].get()
  a2 = dataWV.at[a1].get()

  a3 = a2.T.at[k].get().T
  x1 = x.at[k].get()

  a4 = a3 * x1
  a5 = vmse(a4)
  return mse(a5, dataGT[m])
```

Figure 7: Batched Loss Function

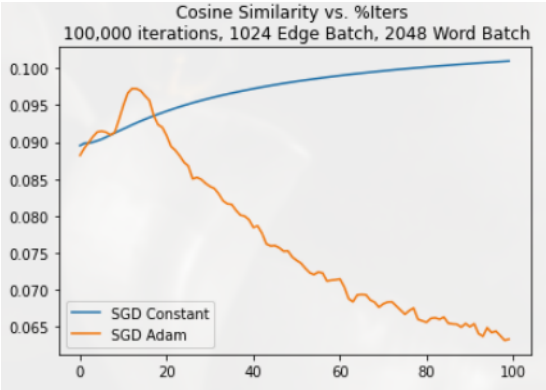

Figure 8: SGD Cosine Similarity

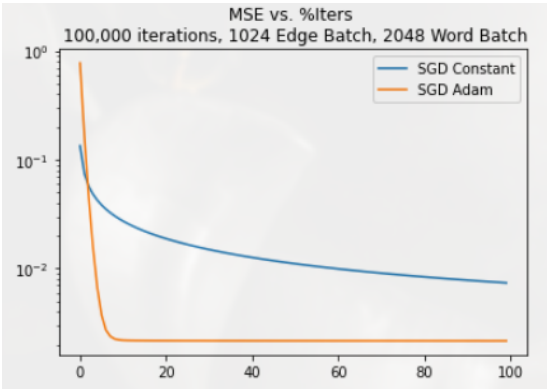

Figure 9: SGD Mean Squared Error

```
Ground Truth vs Cosine Similarity
Cosine Similarity:      0.08787288
Euclidean Distance:     19.994143

Idenities
Cosine Similarity:      0.99999994
Euclidean Distance:     0.0

Zeroed
Cosine Similarity:      0.0
Euclidean Distance:     9.899494

Mean
Cosine Similarity:      0.08073495
Euclidean Distance:     9.867179
```

Figure 10: Comparison of Baselines

```
 Cosine Similarity
Word Count:      0.083588995
SGD:             0.101690404
Adam:            0.05863548

 Mean Squared Error
Word Count:      2.256657
SGD:             0.0076674037
Adam:            0.0023069987
```

Figure 11: SGD vs. Baseline Word Count

```
[('laugh', 0.9802864),
 ('replace', 0.97996575),
 ('hide', 0.97991294),
 ('feeding', 0.97977316),
 ('goodness', 0.9795386),
 ('cry', 0.97935903),
 ('beating', 0.9786494),
 ('meaning', 0.97863996),
 ('sponsors', 0.97840333),
 ('apt', 0.9783374),
 ('speedy', 0.9780893),
 ('barely', 0.9779822),
 ('math', 0.9779201),
 ('disabled', 0.97778326),
 ('ashamed', 0.97768736)]
```

Figure 12: Top 15 Words SGD

```
[('strict', 1.7358292),
 ('magnitude', 1.6299992),
 ('lying', 1.4925587),
 ('path', 1.4883835),
 ('soldiers', 1.4870441),
 ('iron', 1.3758439),
 ('iraq', 1.3636775),
 ('gods', 1.3189857),
 ('projects', 1.3185166),
 ('believes', 1.3124821),
 ('nofly', 1.3078176),
 ('failures', 1.3036107),
 ('planes', 1.2055637),
 ('lived', 1.2036895),
 ('stress', 1.1933296)]
```

Figure 13: Top 15 Words Adam

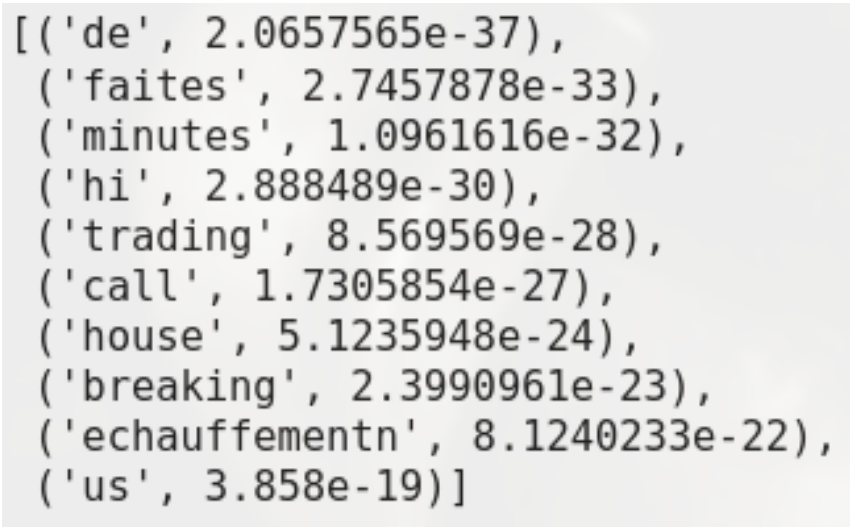

Figure 14: SGD Bottom 10 Words

6 Evaluation & Conclusions

We successfully gathered a Twitter dataset and used the combinatorial nature of the networks towards our advantage by:

- Using the symbolic links in the data as a ground truth, allows us to have a large amount of labelled data.
- Using SGD to be able to harness and process the large amount of data and achieve a higher score than the baseline graphs.

7 Future Work

- The approach can be used on larger and different networks to see what interesting results one may obtain.
- The model used in this work is a linear model, SGD is a great fit for neural networks, and it would be interesting to see SGD identify non-linear features of the text.
- The SGD hyperparameters were not properly chosen, so a global random search can be applied to pick optimal hyperparameters, and furthermore, the model can be ran for longer to see what cosine similarity it can converge to.
- The MSE was used as a rough approxiamtion of the cosine similarity because the gradients were not able to be computed. A better approximation for cosine similarity can be used which may fix Adam.
- Adam can be modified to pause the momentum for the inactive weights, and perhaps this might also fix Adam eagarly adjust the weights.

References

AlMahmoud, H. and S. AlKhalifa (2018). Tsim: A system for discovering similar users on twitter. *Journal of Big Data 5*(1).

Dean, J. and S. Ghemawat (2008). Mapreduce. *Communications of the ACM 51*(1), 107–113.

GeeksforGeeks (2021, 09). ML — Stochastic Gradient Descent (SGD).

Golbeck, J., C. Robles, M. Edmondson, and K. Turner (2011). Predicting personality from twitter. *2011 IEEE Third Int'l Conference on Privacy, Security, Risk and Trust and 2011 IEEE Third Int'l Conference on Social Computing*, 149–156.

Grandjean, M. (2016). A social network analysis of twitter: Mapping the digital humanities community. *Cogent Arts & Humanities 3*(1), 1171458.

Gupta, P., A. Goel, J. Lin, A. Sharma, D. Wang, and R. Zadeh (2013). Wtf. *Proceedings of the 22nd international conference on World Wide Web - WWW '13*.

Healey, P. G. T., C. Vogel, and A. Eshghi. Group Dialects in an Online Community.

iMerit (2021, 12). Top 25 Twitter Datasets for Natural Language Processing and Machine Learning.

Rizio, D. (2021, May). Covid tweets-finding similar twitter users in the first days of the pandemic.

Samatova, N. F., W. Hendrix, J. Jenkins, K. Padmanabhan, and A. Chakraborty (2013). *Practical graph mining with R*. CRC Press.

Seraj, S., K. G. Blackburn, and J. W. Pennebaker (2021). Language left behind on social media exposes the emotional and cognitive costs of a romantic breakup. *Proceedings of the National Academy of Sciences 118*(7).

Singh, K., H. K. Shakya, and B. Biswas (2016). Clustering of people in social network based on textual similarity. *Perspectives in Science 8*, 570–573. Recent Trends in Engineering and Material Sciences.

Tsuruoka, Y., J. Tsujii, and S. Ananiadou (2009). Stochastic gradient descent training for l1-regularized log-linear models with cumulative penalty. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pp. 477–485.

Zhang, T. (2004). Solving large scale linear prediction problems using stochastic gradient descent algorithms. In *Proceedings of the twenty-first international conference on Machine learning*, pp. 116.