Department of Electrical and Computer Engineering
Faculty of Engineering and Architectural Science

Predicting Twitter Users' Interests using Link Prediction of Unobserved Latent Spaces

EDP Final Report

Authors: [Ron Tieu, Trent Rand]
Faculty Lab Coordinator: [Dr. E. Bagheri]
Date: [April 2019]

1 Introduction

Social media has exploded over the last decade to be a frontier where individuals share ideas, opinions, memes, and original content. Over time, user behaviour has become a valuable source of data for a variety of fields; primarily marketing, political campaigning, and a plethora of online businesses. In many cases, these fields are not only looking to see what users are doing and saying, but aiming to interpret communities within the userbase to identify niche markets, swing vote candidates, or new customer bases. What's more is that the limitation of their understanding stops at what's already on the platform. How can a political candidate predict which users are developing a political bias? How can marketers identify users who aren't interested in their product today, but are primed to be exposed to it in the future? Could online businesses user Twitter conversations to predict which users are using like services, but not theirs in particular? If a user raves about their online razor subscription service, when will they develop an interest in other services like monthly feminine hygiene subscriptions?

These questions are the cutting edge of data science in social media. They are based in the tangible desires of organizations looking to optimize the way they benefit from social media data. Past work in academia have used a variety of techniques to attempt to predict how users interests will develop over time.

The goal of this project is to predict the interests of Twitter users using link prediction. This involves taking users' current Twitter data and using that information to predict other interests they would be interested in. The reason this is important is because it is relevant in fields such as advertising and recommendations. Creating this model of predicting the future interests of users is something which is relevant in many different fields, from marketers, political campaigns, or social media platforms themselves.

2 Acknowledgements

We would like to express our very great appreciation to Dr. Bagheri for his professional guidance and continued support throughout this project.

3 Certification of Authorship

We hereby certify that we are the author(s) of this document and that any assistance we received in its preparation is fully acknowledged and disclosed in the document. We have cited all of the sources used to assist us during this project. We also certify that the work presented herein is, to the best of our knowledge, original, resulting from research performed by ourselves, except where otherwise acknowledged.

4 Table of Contents

5 Abstract

In this project, we have created a model that would be able to predict interests for users. From Twitter's API, we were able to parse a dataset containing a large set of data. Using this dataset, we applied LDA topic modelling to cluster the data. This was implemented using gensim in Python. The LDA algorithm takes the documents and returns the topics from each document that are the most relevant. From this model, we used the NetworkX library in python to create a dynamic heterogeneous information network graph (DHIN) containing both nodes of both users and topics. There were weighted edges in the graph that were calculated between nodes depending on how strong the relation was between users, topics, or between user to topic. Once this graph was complete, we would applied a temporal network embedding algorithm in order to output an embedding file. The embedding file would be used to predict user to topic relationships which we would use to evaluate the accuracy of our created model against previously existing models.

6 Objectives

In the past there have been many different methods of predicting the interests of users on Twitter. One of the methods that was used in the past to find interests of users was heterogeneous link prediction using meta paths that have been defined. This method involved using different meta paths to create new links between topics and users. Another method that had been used in the past for prediction of relationships was dynamic heterogeneous information networks (DHIN). This method involved using both latent features and meta paths. Based on past research, the end goal for our project is to create a model that will be able to successfully predict the interests of users similar to previous works. What differentiates our projects from those above is that our project is an expansion on past research and is looking to expand on the work that was done. By looking at multiple past research papers, our key objective was attempting to improve the testing results by combining different methods of relationship prediction in order to return better accuracy results when compared to the state-of-the-art models that already exist.

7 Background

Many previous works have already worked on predicting future interests using many different methods from user's Twitter profiles. The information that can be obtained from tweets can be divided into both explicit and implicit information from tweets. Explicit information would be the information that is directly based on the words used in tweets and sentence structure. Implicit information would be implicit information that was implied from the tweets but not explicitly stated.

Three different methods of modeling user interest were researched into, the bag of words, mixture model, and bag of concepts model. [1]

- **Bag of Words**
  Each interest is represented by a term extracted. Because it is syntax based, it fails to detect underlying semantics.
- **Mixture Models**
  Each interest is represented as mixture of various topics. They can be categorized as topic modeling or feature-based methods. Topic modeling uses co-occurrence patterns of terms to extract sets of correlated terms as topics while feature-based methods focus on user content and uses clustering algorithms to extract sets of terms to form topics.
- **Bag of Concepts**
  Sequences of terms mentioned in textual contents are connected to unambiguous concepts from large databases such as Wikipedia. These databases represent concepts and relationships and can provide the means for inferring underlying semantics of content.

Due to the nature of the bag of words and topic modeling methods, they are not good for short informal tweets and are better used for formal documents.

Some methods for finding relationships [2] between emerging topics include:

- **Semantics Relatedness**
  The relatedness of topics are determined based on semantic similarity of concepts.
- **Collaborative Relatedness**
  The relatedness of topics are determined based on a collaborative filtering strategy where relatedness is measure based on users' overlapping contributions to the topic.

- **Hybrid Approach**

  A combination of both the semantics and collaborative relatedness.

One way of obtaining semantic relatedness is by using semantic information from Wikipedia's category structure. This can be combined with the temporal evolution of user's interests in order to predict user's future interests. The Wikipedia category structure to model high level user interests and takes the temporal evolution of users interests to predict users' future interests. Using a hierarchical approach, when a user is interested in a category it can be assumed they're interested in broader related categories. The Wikipedia category structure can be transformed into a Wikipedia Category Hierarchy (WCH). [3]

Implicit interests can be identified based solely on topic association without semantics. Three steps are used to find the implicit interests of users [4] are:

1. Inferring User Explicit Interests:

Annotating tweets with wikipedia concepts and applying the LDA to discover topics and explicit user interest profiles.

2. Discovering Frequent Topic Patterns:

Frequent Pattern Mining is data mining technique that can be used to extract association between topics on twitter and infer implicit interests by finding closely related topics that frequently co-occur.

3. Interest Profile Augmentation:

The explicit interest profile of the user is augmented using the frequent topic patterns at each time interval with additional interests from frequent topic patterns.

8 Theory & Design

In this chapter we discuss the theory behind our project and how we achieved our goal of predicting topics that users would be interested in. After using Twitter's API to parse data from tweets, there were multiple steps that needed to be done in order to create our model that predicted user interests. Chapter 8.1 discusses how we went from the parsed data from Twitter and applied the Latent Dirichlet Allocation (LDA) in order to cluster the data and find topics. Creating connections between users is discussed in Chapter 8.2, connections between users and topics in Chapter 8.3, and between topics in Chapter 8.4. Chapter 8.5 discusses how we created dynamic heterogeneous information networks and created graphs from this information. Chapter 8.6 contains information on how we applied temporal network embedding to find the latent spaces in the model and used those to find user to topic predictions.

8.1 Software Design and Architecture

The final software design was structured as a data pipeline. The objectives and requirements of the software would be met through three main stages, each implemented separately. The first segment of the pipeline would focus on fetching the required data and prepare it for computations by the LDA Model.

The second stage would take this data and generate a Dynamic Heterogeneous Information Network by identifying the User-to-User, User-to-Topic and Topic-to-Topic relationships. User-to-User relationships were found by retrieving followership information from the dataset. User-to-Topic relationships were assigned by the LDAModel using a consolidation of a user's tweets as the input document to query. Finally, Topic-to-Topic relationships were generated using Word Mover's Distance. Once the edges had been generated, the network graph was formatted into a JSON data format that would be accepted by the third stage of the pipeline.

Temporal Network Embedding through a standalone binary was used as the third stage of the pipeline. Here, the temporally segmented network graphs would be fed through the algorithm which would generate embedding files for each subsequent timeslice.

The final stage of the pipeline was the evaluation of the ability of the software to predict future interests of the users. This involved comparing each embedded timeslice prediction with the future time sliced network graph as generated by the authentic dataset. This involved the Area Under Receiver Operating Characteristic evaluation method. This method compares the model's ability to correctly and incorrectly make future predictions using the rate at which false positives, true positives, false negatives, and true negatives are made by the model.

Due to the large capacity of the dataset it was necessary to execute the software using a cloud computing platform. In this case, we used Google Cloud Platform. The hardware used included a MySQL database to host and manage the datasets. The software, including the LDA Model and Temporal Network Embedding toolkits were executed on a Virtual Machine. This machine featured the following hardware specifications. Intel Skylake with 16 vCPUS and 60Gb of available RAM.

8.2 Datasets and Parsing

The first step in the development was to parse the user data into lists of tweets organized by time and user. In doing so, a simple export of the tweets of 5 random users would allow us to cross check manually whether the robot had properly accomplished the data parsing task. The process for parsing natural language data and producing a model under which topics may be identified is as follows. A corpus (collection of all documents) of natural language is prepared. This preparation involves removing stop words, tokenization, generating bigram and trigram tokens, and finally lemmatization. Stop words in natural language processing are words that are "useless" when it comes to producing insight. This includes words such as "the, a, in, an, so, etc." These terms do not help our model in determining topics and can be removed from the data prior to utilization.

Tokenization reduces a document down to a list of lowercase tokens. This allows for the removal of punctuation, words which are only a single letter, and other noise which is bound to be present in Twitter data.

Often in natural language multiple standalone terms are frequently found together to reference something entirely new from each individual term. The intention or subject of a token can change

drastically if it is found to be part of multiple termed entity. N-gram modelling is a technique used here to provide the model with contextual awareness about this linguistics behaviour. N-gram modelling is a probabilistic approach to predicting the next term in a sequence which takes the form of a (n - 1) order model. This type of modelling is used in DNA and protein sequencing as well as computational linguistics. In our case, bigram and trigram models were queried to give the topic model contextual awareness of 2 and 3 termed topics.

The final step in data preparation prior to the training of the Latent Dirichlet Allocation Model was lemmatization. The lemma of a word is commonly referred to as the root. Lemmatization reduces the terms down to their dictionary definition or base. For any human, the words “democracy”, “democratic”, and “democratization” are all like terms. Should a document be queried for the term “democracy” one should hope that all three terms return true. In order to do so, the data must be reduced to its base. This is also beneficial in cases where tenses and verb conjugations are used. For example, “The birds sang yesterday”, and “I love hearing birds singing in the morning” will both be reduced to “bird sing yesterday” and “love hear bird sing morn”. In both cases, the model should correctly interpret birds and signing as the topics those users were tweeting about. Lemmatization greatly simplifies the natural language to a place where the above is possible. At this stage, the input data of documents and corpus have been adequately prepared to train and test the LDA Model.

8.3 Latent Dirichlet Allocation

Latent Dirichlet Allocation is a mathematical model which is used in computational linguistics to observe trends and spaces within natural language which may not be immediately visible to the human eye. In the shortest terms, LDA is a generative probabilistic model for collections of discrete data. This discrete data, our corpus, is segmented into a series of documents. With the LDA model, each document is represented as a random mixture over latent topics, with each topic defined as a distribution of all words available in the corpus.

Latent in LDA refers to the things we don’t know but are hidden within the data. Dirichlet, and Dirichlet Distribution is the probabilistic model, which in this context is a distribution of distributions. In this way, it is the distribution of topics within documents and distribution of words within the topic.

Allocation refers to the fact that once the distribution is created, topics will be allocated to documents and words of each document back to topics.

The process thus begins by defining a probability distribution of topics within documents. As well as a probability distribution of words in each topic. In this way, the number of topics that are expected in the corpus is already defined as the main control variable of the model, K. With these two distributions it is now possible to define the probability of a word within a given document using a summation of both distributions.

From here, a process known as Gibbs Sampling is invoked. This is the generative process used to sample conditional distributions of variables, with the assumption that the distribution will converge to a true distribution over time.

After all the data had been prepared properly, we used the gensim library's implementation of the Latent Dirichlet Allocation (LDA) in order to cluster the dataset into topics. The LDA is a commonly used algorithm for finding topics inside of documents. The LDA is a method of clustering which takes the group of tweets that we input and clustering them into groups of words that make up topics. Because the data we used was from tweets, the informal data would not have returned optimal results and we had to tokenize them before inserting them into the LDA. which took each tweet as a document.
The steps that the LDA goes through in our LDA Topic model in Python can be seen in Figure 8.1.

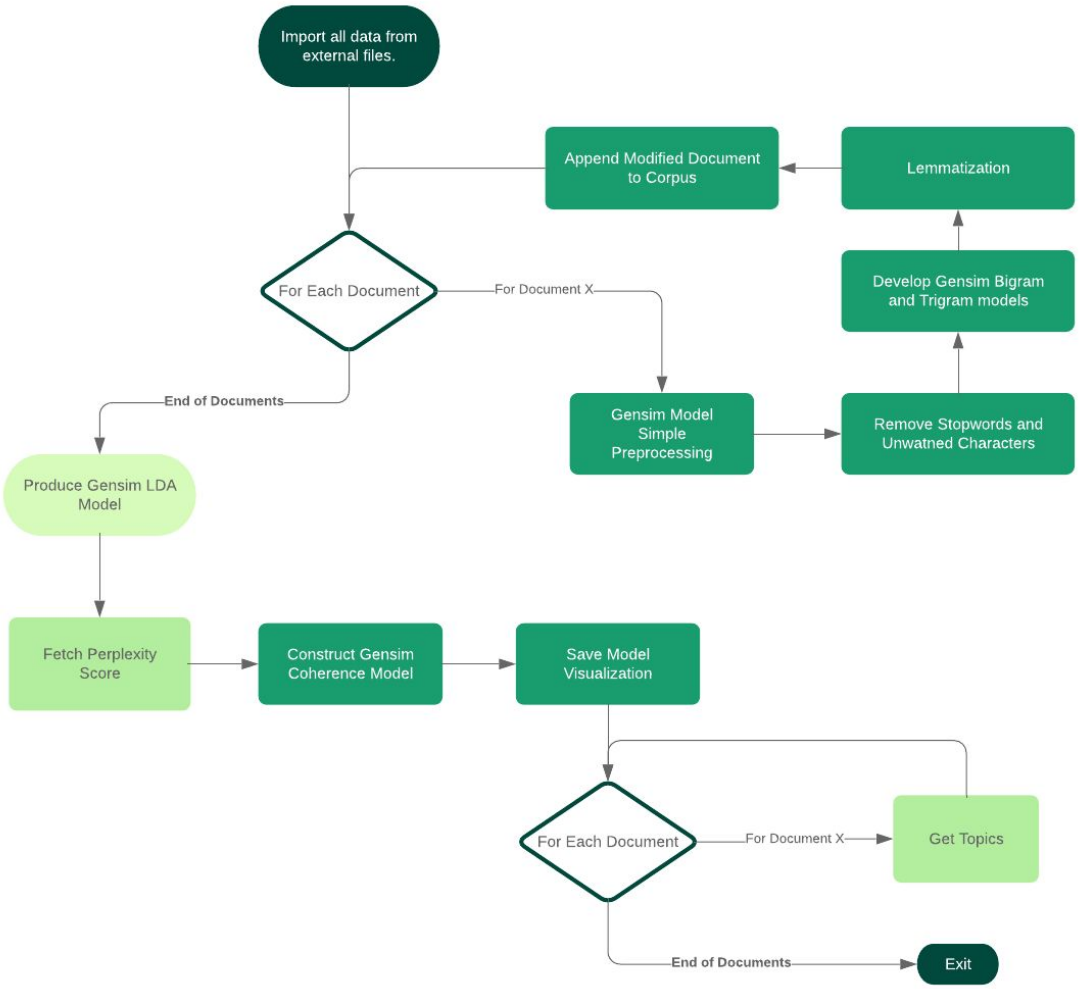


Figure 8.1: Flowchart of the LDA in Python.

8.4 User-to-User Relationships

To find relationships between users for our DHIN, we found the explicit relationships between users using followership, which is already implemented into Twitter. When there was a followership connection between two users, an edge was created between the two user nodes.

8.5 User-to-Topic Relationships

To find user-to-topic relationships, we used the LDA model that was generated. Using the key words that were included in users' tweets, they were given to the LDA model. The LDA model would then output a list of predictions for which topics the user would be most interested in as well as the

prediction. From this, we have a weighted edge between a topic and a node where the weight represents how strong the relationship is between them.

8.6 Topic-to-Topic Relationships

To find the relationships between topics, we used Word Mover's Distance (WMD) in Python. The WMD is a method of calculating the distance it takes for the embedded words of one document to "move" to the embedded words of the other document [5]. We used the WMD because it was a better metric of finding semantic relatedness between documents compared to our initial model which used the cosine similarity. We found that using the WMD resulted in more optimal results and allowed a better model to be generated. Using the word embeddings for the tweets that we already have found, we use WMD to calculate the similarity between different topics. From these similarities we can use this as a weighted edge between the topic nodes in a graph.

8.7 Dynamic Heterogeneous Information Networks (DHIN)

Using the data that we have parsed and the different relationships that we have created, we created a DHIN with the nodes to the graph being topics and users, and the edges of the graph being the relatedness between them. The outputs that were found in Chapters 8.2-8.4, we created these networks by combining them into a single graph. The final result from our DHIN was a graph with nodes of the user or topic types and weighted edges that would connect both the same and different node types.

8.8 Temporal Network Embedding

From our DHIN, we applied a temporal network embedding program to it. The process for this was inputting multiple time-slices of our graph that we found from our DHIN. Once our graphs are input through the program, the temporal network embedding uses the block-coordinate gradient descent algorithm in order to infers the temporal latent spaces for our network that we have built [6]. From this, the program would output a directory of embedding files where we would be able to test our results by making predictions for topics that users are interested in.

Figure 8.2 shows the algorithm that is used to find the optimal solution for the prediction model. For each of the time slices, for each node, it updates the latent position of the node using the update rule and normalizes them until the latent space converges. Figure 8.3 shows the update rule that is used to update the latent positions of the nodes for the BGCD algorithm.

Input: Graphs $\{G_1, \cdots, G_t\}$ and latent space dimension k
Output: latent space $\{Z_1, \cdots, Z_t\}$ and prediction Y_{t+1}
1: Nonnegative initial guess for $\{Z_1, \cdots, Z_t\}$
2: **Repeat**
3: **for** each time τ from 1 to t
4: **for** each u in graph
5: update $Z_\tau(u)$ by Eq. 4
6: normalize $Z_\tau(u)$
7: **until** $\{Z_1, \cdots, Z_t\}$ converges.
8: **return** $Y_{t+1} = \Phi([Z_1, \cdots, Z_t])$ and $\{Z_1, \cdots, Z_t\}$

Figure 8.2: BCGD Algorithm used in Temporal Network Embedding [6]

$$Z_\tau^{(r+1)}(u) = max((1+2\alpha)Z_\tau^{(r)}(u) + \alpha\lambda(Z_{\tau-1}^{(r)}(u) + Z_{\tau+1}^{(r)}(u)) + 2\alpha \sum_{v \in N(u)} G_\tau(u,v)Z_\tau^{(r)}(v) - 2\alpha Z_\tau^{(r)}(u){Z_\tau^{(r)}}^T Z_\tau^{(r)}, 0) \quad (4)$$

where $\alpha = \frac{a_{r+1}+a_r-1}{a_{r+1}L}$, $N(u)$ denotes the set of u's neighbors, L is the Lipschitz constant, $d(u)$ is the degree of node u, and

$$a_r = \begin{cases} 1 & \text{if } r = 0, \\ \frac{1+\sqrt{4a_{r-1}^2+1}}{2} & \text{if } r > 0. \end{cases} \quad (5)$$

Figure 8.3: Update rule for latent position of nodes [6]

9 Alternative Designs

While alternatives to the overall product are more difficult to produce, many of the intermediary steps of the pipeline could be alternatively implemented with other solutions or techniques which would accomplish similar goals.

Data was initially parsed by user, then organized by timeslice. Alternatives to this may include taking only the user content, and fetching relevant users only after the topics have been parsed from the natural language of the content. In this way, the data used in producing network mappings would be primarily derived by its topics, not by the users who contributed at the time.

Other alternative steps in the pipeline include the multitude of strategies that may be used to derive topic-to-topic similarities. Semantic relatedness can be achieved using different methods of interpreting Wikipedia information. Predictions from temporal network embedding also showed that they differ greatly when using a collaborative relatedness evaluation which is driven by the behaviour of the users who also generated the initial content.

10 Material/Component Cost

The project was entirely software and data science driven. As a result, very little material was required. However, the large amounts of data required necessary storage, as well as computational strength which today is primarily achieved through cloud computing. In this way, the material costs of this project fell within what was demanded of the Google Cloud Platform for the execution of the project's code, hosting of the dataset and results, as well as the readily available computation of the SQL databases queried by the model.

The most expensive resource used was hosting the >20Gb dataset as a persistent, publicly queryable SQL database. Over the course of the project, we routinely turned this database on and off to preserve cost. All in all, it still made up roughly 60% of the project's total final cost.

At the end of the project, across 4 months of using Google's platform cost the authors $435.67 CAD. Perhaps more cost efficient platforms or utilizations exist, but should the project be done again, a conservative estimate for the cost of replication would be $500 CAD.

11 Measurement and Testing Procedures

In order to evaluate the model, tests were taken throughout the development and at each stage of the pipeline. It was vital to ensure that at each stage, the require task's solution had been implemented correctly and that the following steps would be built on a solid foundation.

Prior to exploring the process of testing and evaluating in each stage, the following expectations were laid out for the final software solution. It was expected that the software be able to interpret raw Twitter user data including tweets, and identify their content from a natural language format. Secondly, based on the topics of each user's tweets, they should be placed into communities of shared interests. Finally, a prediction model of some kind should be able to produce a better than random prediction for the development of user interests over time as a function of previous interests and shared interests of their communities. This section seeks to outline the testing and evaluation of the techniques used to accomplish the outline objectives.

11.1 Data Parsing and Collection

Three main datasets were used in the development of the software. Tweets sourced the the public Twitter API. This dataset was comprised of over 100 thousand tweets across the last 5000 users which had tweeted chronologically since November 26th, 2018 at 19:04:23 EST. The second dataset was a collection of 20000 Yelp Reviews which had been sourced from the public Yelp API. The major dataset was a collection of tweets

In order to achieve the methods of parsing which were necessary to prepare the natural language for the LDA Model, two python libraries were used. These works include the gensim toolkit, a collection of tools used in topic modelling and natural language processing. The other library necessary was spaCy which performed the lemmatization of the data.

11.2 LDA Model Testing and Development

The second stage of the model was the development of the Topic Latent Dirichlet Allocation Model which was used to parse topics from provided natural language user data. In this way, it was important to ensure that the model was able to properly identify topics, remove content such as links and hashtags that may confuse it, and finally to ensure that the topics it generated were unique and reasonable for human understanding.

While the GenSim python library was capable of producing our model from a provided corpus, the process of training was incredibly computationally intensive. It was important to first ensure that the training was performing as expected on smaller datasets prior to providing the large overarching corpus of tweet data.

One of the initial training datasets included over 20 000 restaurant and hotel reviews from Yelp. This dataset was chosen as the terms and discussions found within this type of data could be relatively well predicted from a human perspective. It is understood that the final produced model will be largely abstract and thus difficult to interpret. Additionally, topic modelling systems provide no guarantees on the interpretability of their output. As a result using smaller well understood datasets, it would be determined as to whether or not the model was performing as expected.

Once the model had been trained on a corpus of Yelp reviews, each topic and its relevant terms were visualized. While this qualitative approach worked well on a small dataset, it is not scalable and at the mercy of an individual human interpretation. This is far from ideal and in order to optimize the final topic model more formal methods of evaluation were also implemented. The most widely accepted form of LDA model evaluation is topic coherence.

The major variable when producing and training different iterations of the model comes from adjusting the number of topics it will discover. Over 20 000 yelp reviews you may only find 50 topics; it’s a fairly concise and specific dataset. On twitter however, users are encouraged to discuss everything. In order to accurately map the tweets of hundreds of thousands of users, you may need a couple thousand topics. Adjusting the number of topics for each model training iteration and optimizing for coherence, is a method to determine the number of topics that should be used when evaluating a dataset. Our

software solution used the coherence methods already present in the GenSim toolkit, but the calculation of topic coherence is as follows.

Four steps exist in our coherence modelling pipeline; segmentations, probability estimation, confirmation measurement and aggregation. Segmentation involves the measurement of a word subset W, as it is support by another subset. The result of segmentation of a given word set W is a set of pairs of subsets of the wordset W. Traditionally proposed coherence methods compare pairs of single words, while segmentation her creates subset pairs such that every single word in each topic wordset is paired with another set and single words in each other set. This is a fairly exhaustive process.

Once this is completed, probability estimation may begin. This stage of the pipeline utilizes a boolean sliding window methods to estimate the probability of each word existing in each document against the number of total documents. The window moves over each document one token/word at a time. Each step defines a new document by copying the content of the current window. From there, each virtual document is computed under a boolean document method to determine word probabilities. Boolean document estimates the probability of a single word as the number of documents in which it occurs divided by the number of total documents.

Confirmation measure makes up the third step. This takes each single pair of words and their probabilities to compute the degree at which the topic wordset supports the single word. Finally, all confirmations of each subset pair is aggregated into a single coherence score.

11.3 Network Graph Testing

After the LDA Model, the next step in the pipeline was creating temporal network graphs by relating topics, users and timeslices. This testing was fairly simple. It was understood that the level of abstraction achieved at this stage pushed the level of human understanding, let alone the ability to visualize the results. As such, smaller dataset were once again used to create sample network graphs of the content. It was here that we could evaluate whether the model was accurately grouping like topics, assigning the correct users to each topic based on their content, and finally, grouping and connecting users appropriately based on their submissions to twitter. Some samples of these generated graphs are

included in the section below.

11.4 Full Performance Evaluation

The final stage of the software involved using the time-sliced network graphs to perform temporal network embedding and predictions. To evaluate this stage when executed on the correct data, was to evaluate the success and performance of the entire model. Prior to those evaluations, it was necessary to ensure that the software used was operating on and interpreting the provided data correctly.

This segment of the software's pipeline was executed on a compiled set of code. We ran under the assumption that an error-free execution of the program would suggest that the data was interpreted correctly and that the Temporal network Embedding and Prediction software was able to perform its desired task.

In order to gauge the accuracy of the predictions made by the software, an AUROC model was adopted. AUROC, or Area Under Receiver Operating Curve is a performance metric used when evaluating the performance of a classification system or model. It measures the system's performance based on the rate of true positives predicted against the rate of false positives predicted. In essence this curve measures the model's ability to predict 1s as 1s and 0s as 0s, against how often it predicts a 1s as 0s, and 0s as 1s. A linear, or 0.5 threshold of the curve would suggest that the model is as accurate as a coin toss. The closer to 1, and the greater the area under the curve indicates tangible, replicable accuracy of the model. The AUROC results of the trials and improvements of the model have been provided in the following section.

12 Measurement Results

The results from our project can be split up into the several components that we have created during our pipeline to the end product. Chapter 12.1 discusses the results of the LDA model that we created first.

12.1 LDA Model

The following showcases the steps involved in testing and evaluating the Latent Dirichlet Allocation Topic Modelling system, as well as the process under which a model is trained and developed.

The first figure demonstrates the Word2Vec model and how it would isolate like terms in an abstract space. That abstraction has been mapped in 2-space as to visualize it meaningfully.

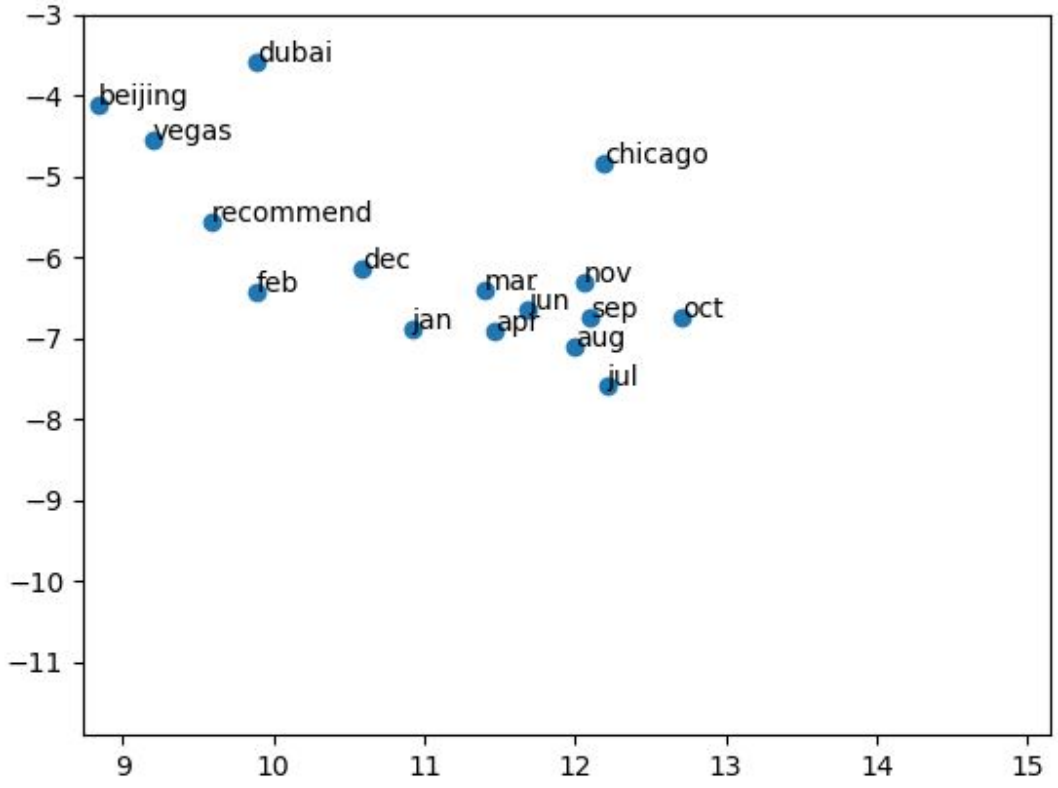


Figure 12.1: Word2Vec model showing cluster of months and cluster of cities.

It is clear that the model can correctly identify the likeness of month three letter codes and has isolated them from other terms within the corpus based on their use within the documents. At this point, it appears rudimentary functionality of term understanding has been achieved. This does not permit us to proceed to the generation of the LDA Model however, and a second proof of concept test was executed with more complex data that would closer resemble the final twitter dataset.

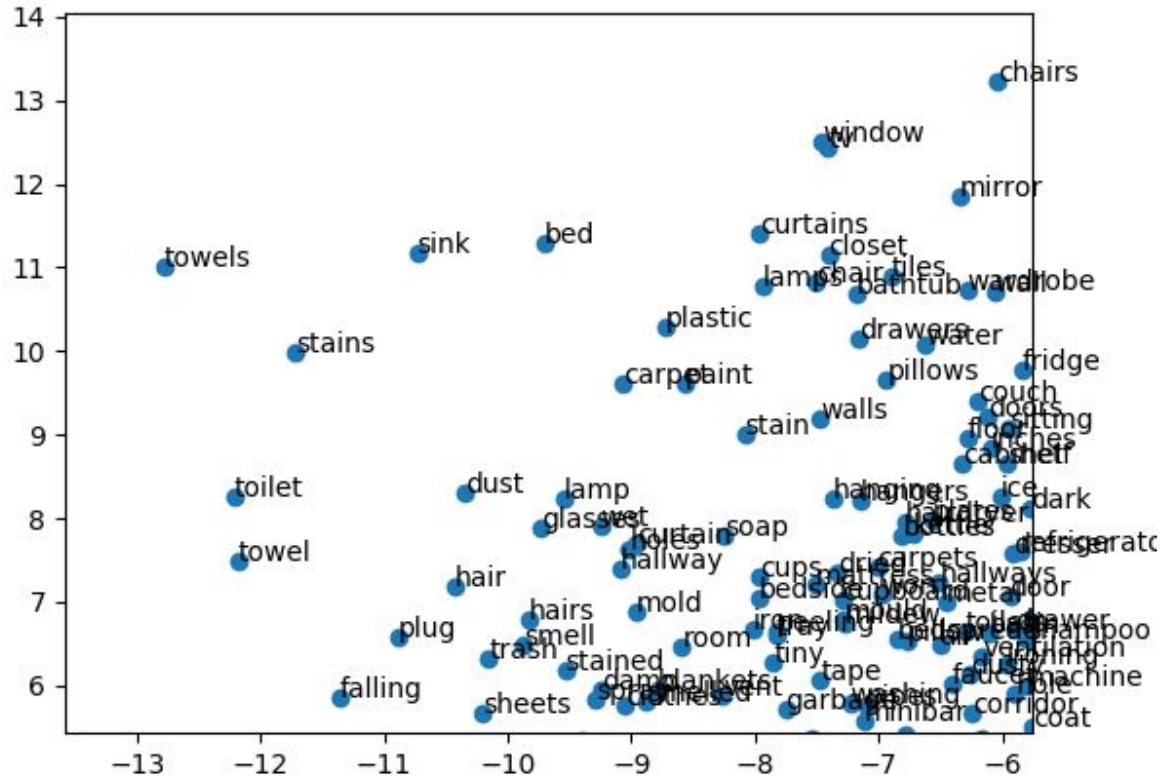


Figure 12.2: Word2Vec model built from Yelp Reviews.

The above figure demonstrates the ability of the model to map likeness of words to a spatial mapping in 2-space. While meaningful clustering of words hasn't occurred yet, it is clear that the model is capable of interpreting definition and purpose of terms within documents based on their use in natural language compared to the use and proximity of other terms within said document.

From here, it was possible to use this Word2Vec model as a basis from an LDA Topic modelling system. We reused the Yelp data as a means to both evaluate the LDA model's performance, but to provide some semblance of progress on the data analysis that was being performed.

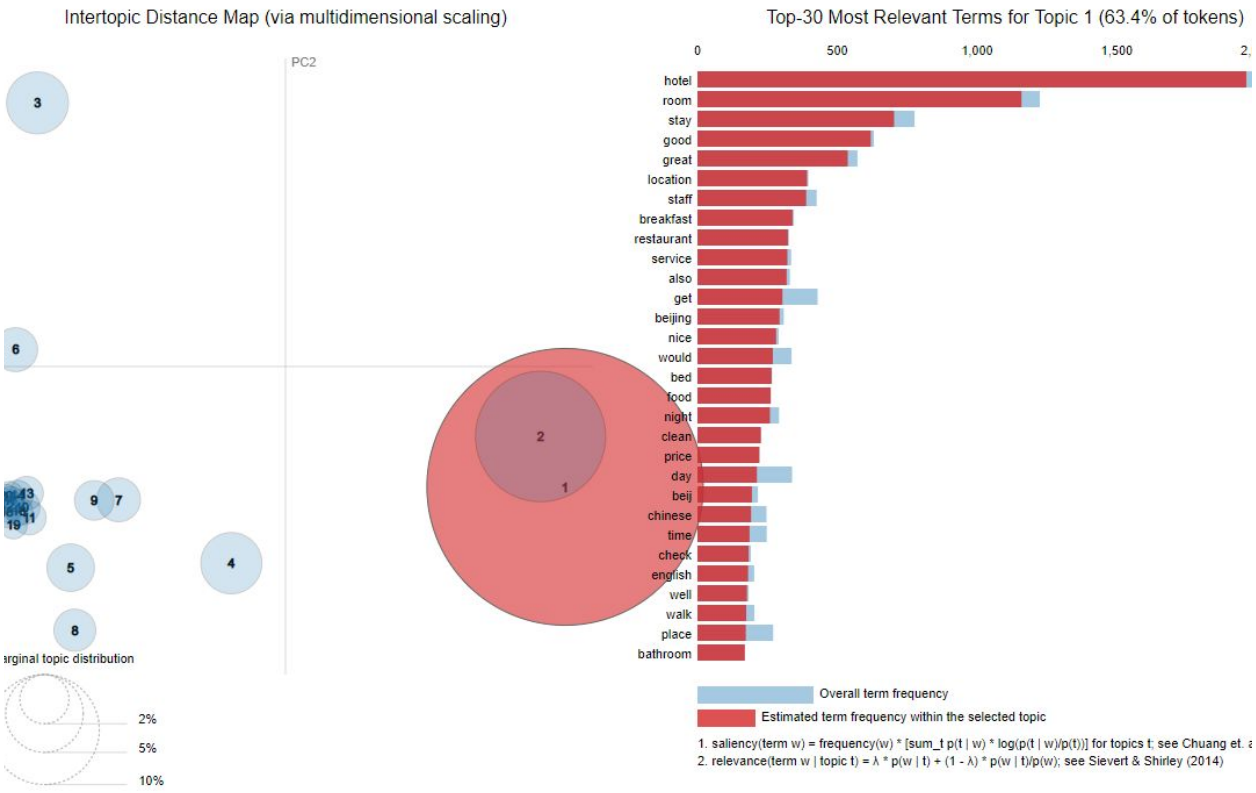


Figure 12.3: LDA Model generated from Yelp reviews with 20 topics.

Additional tests were performed against multiple iterations of the model. The following figure illustrates a case where the model was told to assume a number of topics which were too large. This lead to many topics being identified as incredibly similar to one another. This is one form of overfitting that is possible from an LDA model. It can easily be resolved by reducing the topic count, but that is not easily as verifiable when the corpus is large. The data used in this overfitting example were Trump tweets over a period of 8 months.

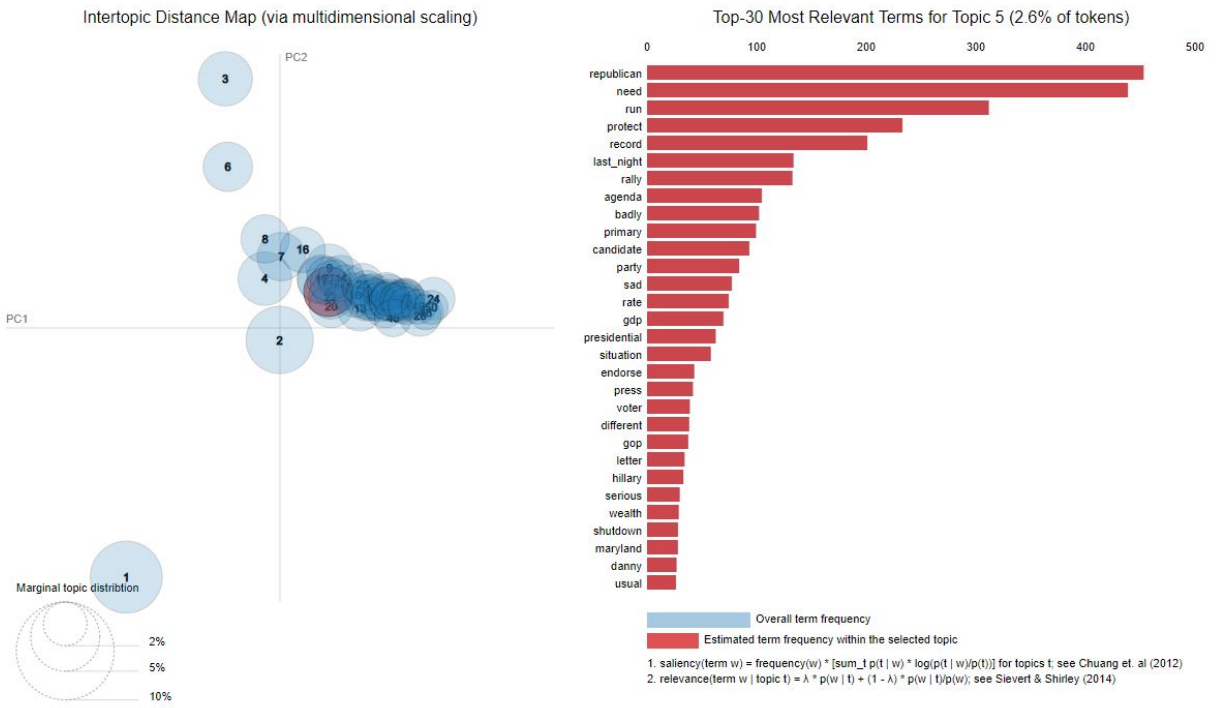


Figure 12.4: LDA Model generated from Trump tweets.

Toeing the line between proper visualizations for evaluation and absorbing a meaningful amount of data is difficult. On one hand, using small sample sizes can simplify the process of certifying the authenticity of results, however the level of abstraction that arises with large data sets can prove to be challenging to evaluate visually.

12.2 Temporal Network Graph Generation

When producing network graphs, it was impossible to visualize them at full scale. Smaller sample datasets were used to certify that the functionality was behaving as intended, even when the full dataset's complexity would prevent such an assurance.

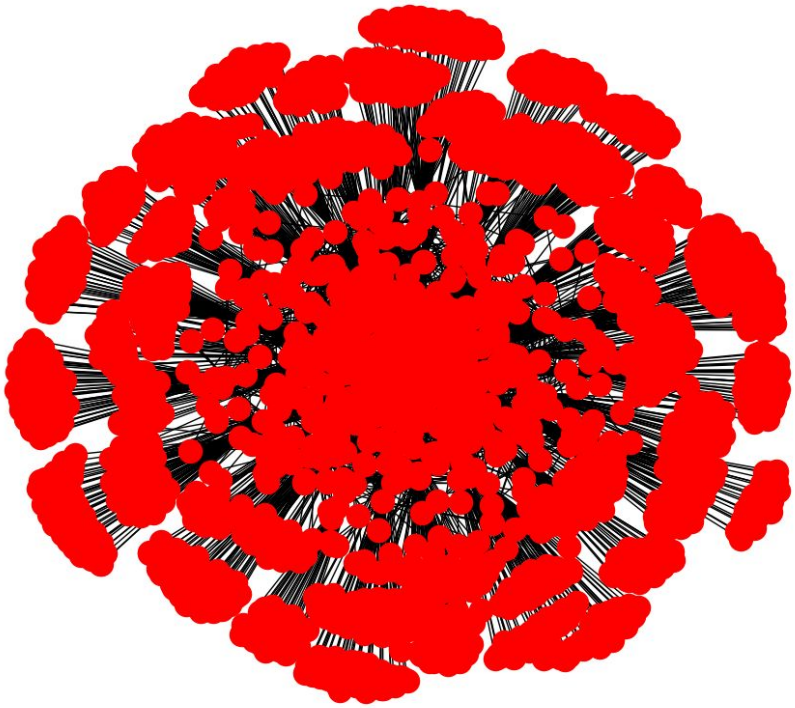


Figure 12.5: Sample user-to-topic network graph.

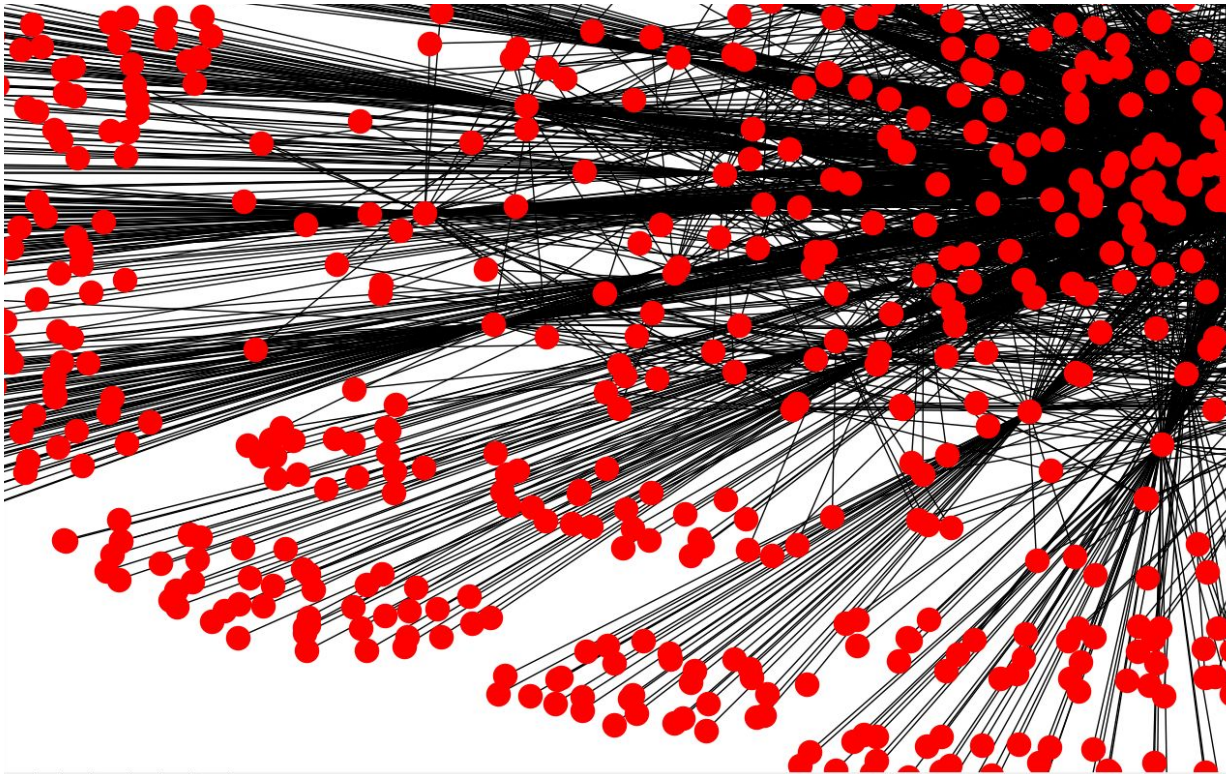
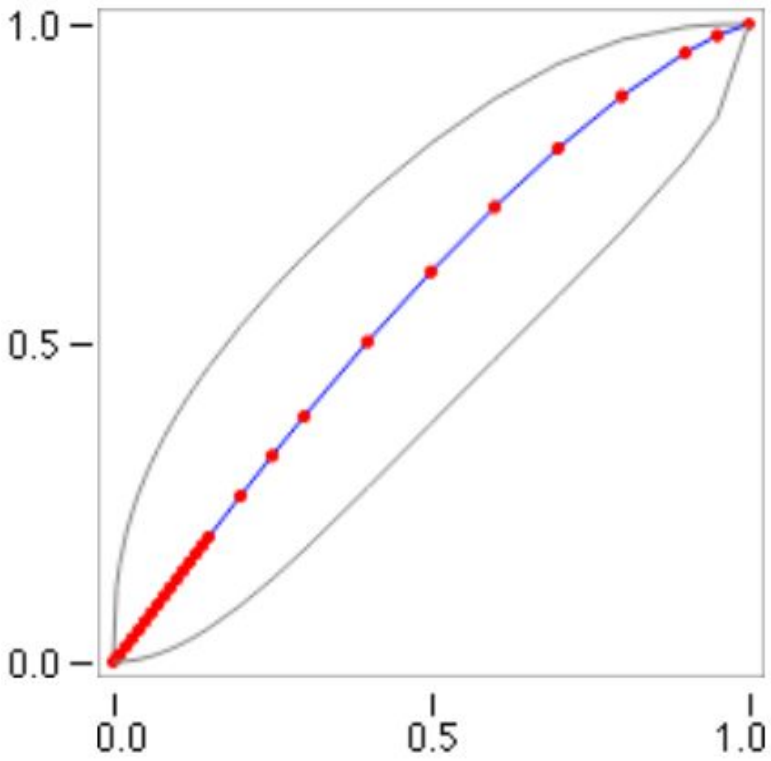


Figure 12.6: Zoomed in user-to-topic graph showing groups of users connected to the same topic.

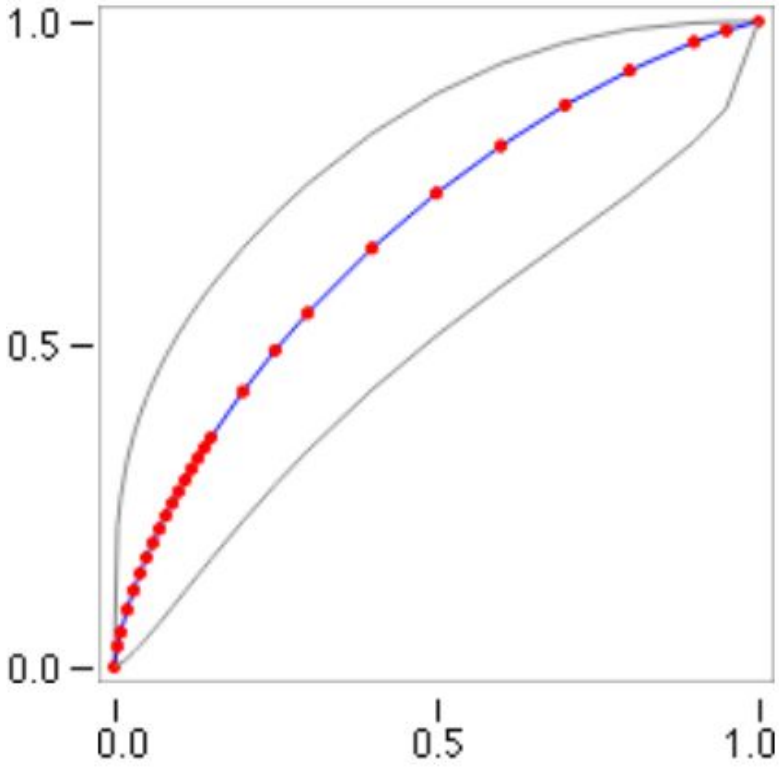
Upon producing the figure above, it was clear that the model had grouped users into "communities" based on the LDA Model's assigned topic interests. With this simpler model, each user was only assigned a single topic. Topic nodes are located in the middle, and contributing user communities in the surrounding area.

12.3 Final Model and Prediction Results

At this stage, the pipeline from user data in preparation for temporal network embedding and prediction had completed. It was here that we began to run full iterations of the software and evaluate each once it had completed it's prediction process. The following figures illustrate a variety of AUROC curve results as they relate to iterations of testing.



Figures 12.6: ROC Curve resulting in an AUROC value of 0.507.



Figures 12.7: ROC Curve resulting in an AUROC value of 0.714.

The above figure is the result of the final prediction model using 50 potential topics, over 5000 uniques users, and week long time slices across 20 weeks. The 0.672 result was one of the highest results that was able to be achieved using a 50 topic LDA Model. Not yet featured are trials that include a 20 topic model across >20000 unique users and 52 weeks. On the larger scale, the temporal network prediction fell apart, achieving only 0.507. At that rate, it was consistently worse than a randomized prediction, or coin flipping model. The highest achieved result from AUROC was 0.714. Table 12.1 shows the performance of our initial and final model against already existing prediction models.

Table 12.1: AUROC values showing performance of different models compared to our model

Model	Adamar /Adar	Common Neighbor	Jaccard coefficient	Katz β =0.0005	Katz β =0.005	Katz β =0.05	SimRank λ =0.8	**Temporal Network Embedding**
F	0.500	0.500	0.500	0.524	0.524	0.528	0.510	
S	0.791	0.790	0.774	0.790	0.790	0.788	0.500	
SF	0.791	0.790	0.762	0.757	0.753	0.720	0.520	
C	0.712	0.710	0.700	0.714	0.715	0.728	0.500	
CF	0.773	0.771	0.758	0.752	0.738	0.716	0.517	
CS	0.762	0.761	0.748	0.763	0.763	0.767	0.500	
CSF	0.762	0.761	0.738	0.736	0.732	0.707	0.520	
Initial								**0.507**
Final								**0.714**

13 Post Analysis

In brevity, it was clear that the overall software solution was able to perform its intended functionality at or above the required expectations. It met its objectives and successfully identified user communities of shared interests as well as predicted to a significant degree of accuracy potential future user interests.

Some concerns do linger about the performance of the model and it's robustness in the face of new or completely organic data. While the initial trials were at or below a random model, tweak and adjustments managed to improve the model's performance to meet the criteria. However, this model was trained and evaluated against the same dataset in each instance. As a result, it is possible that the changes in the model's input variables resulted in a model that was overfitted to the provided data, and not a model at which it could produce consistent results against any data.

While we were working on our results, we found different ways we could change the metrics in order to improve our results. One of the ways we could improve the results of our LDA model was by optimizing our LDA coherence score. This was done by adjusting our number of topics that we chose for our LDA model by using a trial and error based process. One way that we managed to greatly improve our models prediction results was by adjusting the way that we calculating user to user predictions. We found that user followership wasn't the best metric of seeing whether two users would be connected by a relationship. Connecting users based on their contributions to the same topics was a more optimal way that we used in order to create relationships between users rather than the explicit connections between them.

When comparing our results of 0.714 to the highest achieved results of 0.792, there's a gap where a lot more improvement can still be made for our model. If we had more time to adjust some parameters such as topic numbers in order to we maybe have gotten a higher AUROC value. For even larger improvements, we may have had to change some of the ways we calculated relationships such as user relationships and try a different methodology in order to higher final results.

14 Conclusions

The course of development for this solution was the incremental building on previous successes and tasks. In conjunction they created a final solution that managed to outperform some previously existing models as well as significantly outperformed a “coin-toss” random prediction model. In brevity, the creation of software to solve the problem statement was achieved successfully.

Over the course of development some smaller accomplishments were necessary. This began with successful creation and training of a machine learning based topic modelling system. This was our Latent Dirichlet Allocation Model. Trained on over 20 Gb worth of twitter user data, it was able to separate tweets into a probability rates of their belonging to particular topics. It also managed to assign keywords and concepts to each topics. Iterations of this model managed to produce 20, 50, 100, and 1000 topics across the natural language input data.

Network graphs were generated by querying both a SQL database and the LDAModel to pair users to their suspected interested, based on their contributed content. Users were connected via direct messages, shared topic interests, and followerships. Ultimately the best performing models were based solely on LDAModel identified shared topic interests. Followership based mapping resulted in the worst predictions. This may even suggest that followership are a better gauge of topical interest than of user connection in the real world. This is true when celebrities and influencers within preconceived communities are used as indicators of interest as opposed to user-to-user relationships. Future work should consider using this type of interpretation of followerships to see if they can improve predictions even further. Perhaps even reaching a point where new followerships could be suggested based on a users contributions to the platform.

Several major issues were encountered and overcome during the development of this solution and software.

The initial iterations of the complete solution were unsuccessful. The first actually underperformed against a random prediction model. With tuning and the adjustment of multiple variables across the entire pipeline managed to increase the performance of the system significantly. An

AUROC result of 0.714 was the highest achieved by our solution, a successful model when it came to producing prediction about future user to topic connections.

While that result is satisfactory towards the objectives of the project, future work should be able to continue to tweak the model to produce an even higher result. Do be warned however, over fitting to the training data and continuous evaluation against the same dataset will not create a model that works during forward testing of new data. It will simply produce an optimized robot for the specific data provided. Future work should aim to improve the model's results, but should remain skeptical about creating a model that works exceptionally on the training sets, but incredibly poorly on real world, future data.

15 Appendices

15.1 Source Code

```
##Import Packages
import re
import numpy as np
import pandas as pd
from pprint import pprint
import networkx as nx

import json

####La7S<AB?5_H=ObFmIo{


# Gensim
import gensim
import gensim.corpora as corpora
from gensim.utils import simple_preprocess
from gensim.models import CoherenceModel
from gensim.test.utils import datapath

# Let's go get our friend WikiRelate:
#from folder.wikirelate as wikirelate

# spacy for lemmatization
import spacy

# Plotting tools
import pyLDAvis
import pyLDAvis.gensim  # don't skip this


# MySQL Import
import pymysql

# Enable logging for gensim - optional
import logging
logging.basicConfig(format='%(asctime)s : %(levelname)s : %(message)s', level=logging.ERROR)

import warnings
warnings.filterwarnings("ignore",category=DeprecationWarning)
```

```
## 5. Prepare Stopwords
from nltk.corpus import stopwords
stop_words = stopwords.words('english')
stop_words.extend(['from', 'subject', 're', 'edu', 'use', 'timestamp', 'com', 'http', 'www'])


## 6. Import Newsgroups Data
# Import Dataset
connection = pymysql.connect(host='127.0.0.1', user='root', password='5TRcaSeTr4L',
db='Twitter')


##
#
#df = pd.read_csv('abcnews-reduced.csv', dtype=str, usecols=[1])

Tweets_df = pd.read_sql('SELECT * FROM Tweets', con=connection)


UserIdList = Tweets_df.iloc[:,3].tolist()
UserIdString = str(UserIdList)
UserIdString = UserIdString[1:-1]



Users_df = pd.read_sql(('SELECT * FROM Users WHERE Id IN ('+UserIdString+')'),
con=connection)



data = Tweets_df.iloc[:,1].tolist()


##Tokenize document in GenSim modifiable tokens.
def sent_to_words(sentences):
    for sentence in sentences:
        yield(gensim.utils.simple_preprocess(str(sentence), deacc=True))  # deacc=True
removes punctuations

data_words = list(sent_to_words(data))


    # Creating Bigram and Trigram Models
    # Build the bigram and trigram models
bigram = gensim.models.Phrases(data_words, min_count=5, threshold=100) # higher threshold
fewer phrases.
trigram = gensim.models.Phrases(bigram[data_words], threshold=100)
```

```
    # Faster way to get a sentence clubbed as a trigram/bigram
bigram_mod = gensim.models.phrases.Phraser(bigram)
trigram_mod = gensim.models.phrases.Phraser(trigram)

    #Remove Stopwords, Make Bigrams and Lemmatize
    # Define functions for stopwords, bigrams, trigrams and lemmatization
def remove_stopwords(texts):
    return [[word for word in simple_preprocess(str(doc)) if word not in stop_words] for doc
in texts]

def make_bigrams(texts):
    return [bigram_mod[doc] for doc in texts]

def make_trigrams(texts):
    return [trigram_mod[bigram_mod[doc]] for doc in texts]

def lemmatization(texts, allowed_postags=['NOUN', 'ADJ', 'VERB', 'ADV']):
    """https://spacy.io/api/annotation"""
    texts_out = []
    for sent in texts:
        doc = nlp(" ".join(sent))
        texts_out.append([token.lemma_ for token in doc if token.pos_ in allowed_postags])
    return texts_out

    # Remove Stop Words
data_words_nostops = remove_stopwords(data_words)

    # Form Bigrams
data_words_bigrams = make_bigrams(data_words_nostops)

    # Initialize spacy 'en' model, keeping only tagger component (for efficiency)
    # python3 -m spacy download en
nlp = spacy.load('en', disable=['parser', 'ner'])

    # Do lemmatization keeping only noun, adj, vb, adv
data_lemmatized = lemmatization(data_words_bigrams, allowed_postags=[u'NOUN', u'ADJ',
u'VERB', u'ADV'])

    ## 11. Create the Dictionary and Corpus needed for Topic Modeling
    # Create Dictionary
id2word = corpora.Dictionary(data_lemmatized)

    # Create Corpus
texts = data_lemmatized

    # Create corpus bag of words
corpus = [id2word.doc2bow(text) for text in texts]
```

```
    # Human readable format of corpus (term-frequency)
[[(id2word[id], freq) for id, freq in cp] for cp in corpus[:1]]

    ## 12. Building Topic Model (LDA)
    # Build LDA model
load_file = datapath("TweetsLdaModelDatapath")
lda_model = gensim.models.ldamodel.LdaModel.load(load_file)


    # Print the Keyword in the 10 topics
pprint(lda_model.get_topics())
doc_lda = lda_model[corpus]

ndarray = lda_model.get_topics()

nameList = lda_model.get_topic_terms(2)

user_topic_edges = [(0,0),(0,1),(1,1)]

#For each lemmatized tweet:
ind = 0;
for doc in data_lemmatized:
    q_vec = id2word.doc2bow(doc)

    #Fetch the topic of each tweet, in order to pair it with a UserId.
    topic_vec = lda_model.get_document_topics(q_vec)

    topic_prob = 0;
    topic_id = -1;
    for pair in topic_vec:
        prob = pair[1]
        id = pair[0]

        if prob > topic_prob:
            topic_prob = prob
            topic_id = id

    user_id = UserIdList[ind]

    user_topic_edges.append((topic_id, user_id))

    ind = ind + 1;

G = nx.Graph()
G.add_nodes_from(UserIdList)
G.add_edges_from(user_topic_edges)
```

```
nx.draw(G)


nx.write_edgelist(G, "test1.edgelist", data=True)
graphJSONData = nx.readwrite.node_link_data(G)


vis = pyLDAvis.gensim.prepare(lda_model, corpus, id2word)
pyLDAvis.save_html(vis, 'TestRunExport.html')

# We made it here, we can go anywhere
# "Don't put song lyrics in the comments" -Ron, probably.

# wikirelate was here

topicModel = Word2Vec.load_word2vec_format('conceptEmbeddings.bin.gz', binary=True)


t = 0;
for topic in np.nditer(ndarray):

    tsg.add_node(t)

    try:
        topicTermsA = lda_model.show_topic(t)
        topicTermsB = lda_model.show_topic(t + 1)

        distance = topicModel.wmdistance(topicTermsA, topicsTermsB)

        tempEdge = (t, t + 1, {'weight':distance})

        topic_topic_edges.append(tempEdge)
    except:
        #Pathetic.
        print("End of Topics!")

tsg.add_weighted_edges_from(topic_topic_edges)
neworkGraphJSONData = nx.readwrite.node_link_data(G)
json.dump(networkGraphJSONData, 'topicGraph.json')
```

16 References

[1] Zarrinkalam F., Fani H., Bagheri E., Kahani M. (2016) Inferring Implicit Topical Interests on Twitter. In: Ferro N. et al. (eds) Advances in Information Retrieval. ECIR 2016. Lecture Notes in Computer Science, vol 9626. Springer, Cham

[2] Zarrinkalam, Fattane & Fani, Hossein & Bagheri, Ebrahim & Kahani, Mohsen & Du, Weichang. (2015). Semantics-Enabled User Interest Detection from Twitter. 469-476. 10.1109/WI-IAT.2015.182.

[3] Zarrinkalam F., Fani H., Bagheri E., Kahani M. (2017) Predicting Users' Future Interests on Twitter. In: Jose J. et al. (eds) Advances in Information Retrieval. ECIR 2017. Lecture Notes in Computer Science, vol 10193. Springer, Cham

[4] Trikha A.K., Zarrinkalam F., Bagheri E. (2018) Topic-Association Mining for User Interest Detection. In: Pasi G., Piwowarski B., Azzopardi L., Hanbury A. (eds) Advances in Information Retrieval. ECIR 2018. Lecture Notes in Computer Science, vol 10772. Springer, Cham

[5] Kusner, Matt & Sun, Y & Kolkin, N.I. & Weinberger, Kilian. (2015). From word embeddings to document distances. Proceedings of the 32nd International Conference on Machine Learning (ICML 2015). 957-966.

[6] Zhu, L., Guo, D., Yin, J., Steeg, G. V., & Galstyan, A. (2017). Scalable temporal latent space inference for link prediction in dynamic social networks (extended abstract). 2017 IEEE 33rd International Conference on Data Engineering (ICDE). doi:10.1109/icde.2017.35