Analyzing and Predicting Trends in Contemporary Social Discourse through Hashtag Campaigns

by

Shihab Muhtasim
21301610
Raiyan Wasi Siddiky
21301648

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science

Department of Computer Science and Engineering
Brac University
October 2024

Declaration

It is hereby declared that

1. The thesis submitted is my/our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

Student's Full Name & Signature:

Shihab Muhtasim
21301610

Raiyan Wasi Siddiky
21301648

Approval

The thesis/project titled "Analyzing and Predicting Trends in Contemporary Social Discourse through Hashtag Campaigns" submitted by

1. Shihab Muhtasim(21301610)
2. Raiyan Wasi Siddiky(21301648)

Of Summer, 2024 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science on October, 2024.

Examining Committee:

Supervisor:
(Member)

Dr. Farig Yousuf Sadeque
Assistant Professor
Department of Computer Science and Engineering
Brac University

Co-Supervisor:
(Member)

Monirul Haque
Lecturer
Department of Computer Science and Engineering
Brac University

Head of Department:
(Chair)

Sadia Hamid Kazi, PhD
Chairperson and Associate Professor
Department of Computer Science and Engineering
Brac University

Abstract

In the current landscape of social media, hashtags play a significant observable role in boosting the movement of information across a diverse range of fields and people. This research aims to identify and then examine extremely popular hashtags on social media and uncover the characteristics that make these hashtags popular. It attempts to follow the hashtag along its journey over time in gaining and losing popularity and in doing so, extrapolates to analyze the boom of past trends over time and the impact that hashtags have in propagating information. The research further tries to create a structure for a network showcasing the propagation of information and interaction between users as they engage using the hashtag in an attempt to understand why and how these hashtags reach such a diversified range of groups and people. It also attempts to be able to predict and forecast the trend and direction of the hashtag and the interaction it generates after the interaction period. Using a mixture of modern techniques such as network science and graph theory concepts, unsupervised machine learning tools, greedy modularity maximization model, and many other natural language processing (NLP) tools such as LSTM, there is the potential to understand the influence of popularity through hashtags and be able to predict the popularity of hashtags.

Keywords: Hashtags, Information Propagation, Hashtag Characteristics, Network Science, Machine learning, Natural Language Processing(NLP), Trends Analysis, Predictive Modeling, Greedy Modularity Maximization, LSTM

Acknowledgement

We would like to thank our parents for always supporting us. Additionally, We would like to thank our supervisor, Dr. Farig Sadeque, and acknowledge his consistent effort in guiding us every step of the way. We would also like to thank our co-supervisor Mr Monirul Haque for his technical guidance. We are incredibly grateful for everyone's support, mentioned and not mentioned.

Table of Contents

List of Figures

List of Tables

1. Introduction

The world is rapidly moving towards complete digitalization. Every aspect of life is slowly but surely becoming automated and relying on the use of modern tools and technology. One of the most significant changes is the advancements in communication. Humanity now lives in a time where information travels extremely quickly through social media, which has been a focal point of the rapidly changing world. One of the heralds ushering in this new age has been X(formerly known as Twitter), standing out as a central hub for vast dialogues of communication, propagating a diverse variety of different topics including but not limited to news, politics, comedy, opinions, and much more. To clarify, X is a micro-blogging platform that allows users to make short-form messages limited to less than 140 characters called tweets. One of the reasons X has been such a standout has been its use of hashtags, which are words or phrases with the # symbol in front of them. Hashtags are exceptionally efficient tools for grouping information and spreading media to vastly diverse groups of people. Moreover, they have been revolutionary at not only helping cluster relevant and related data but also shaping public opinion. As such, hashtags have quickly become a dominant factor in pushing trends and creating large amounts of social discourse. As a result of all this, it has become imperative to comprehend and understand exactly how hashtags work and analyze fully their capacity to manipulate trends, as this could result in the ability to not only comprehend these unique tools of communication but also gain a deeper understanding of trends themselves and even be able to predict and forecast the direction of public sentiment. While similar research has been done previously, this study intends to go further, incorporating network science, machine learning, and natural language tools to analyze and predict trends through hashtags.

The primary focus of this research is on network science. While machine learning and natural language processing (NLP) techniques have been utilized as tools to facilitate the analysis, the core contribution lies within the domain of network science itself. This study seeks to advance the understanding of complex systems modeled as networks, particularly within the context of social media. By applying network science methodologies, we aim to contribute to the growing body of knowledge in the field, with a specific emphasis on community detection as well as user activity predictions to learn user interaction patterns and analyze social trends.

This research makes use of a detailed dataset that showcases the proliferation of popular hashtags while mainly focusing #Bitcoin on Twitter over a limited time frame. The dataset was thoroughly preprocessed and analyzed to better gain an initial understanding of the many factors that might influence the growth and usage of a hashtag. Multiple factors were looked at such as hashtag co-occurrence, user engagement differences over different combinations of hashtags, user-to-user interactions throughout the Network, hashtag popularity, user activity, and more. Using the processed data, a variety of network graphs were developed to create a visual representation of the features. The graphs were thoroughly analyzed and were used to affirm the existence of communities in the network connected through shared engagement. The graphs were constructed using an algorithm proposed by Clauset et al [1] in 2004 called the greedy modularity maximization

method that detects communities from the graph. Furthermore, using properties suggested by Barabasi [9] such as the degree, clustering coefficient, average path length, degree centrality, etc. the graphs and communities were evaluated and analyzed. Further analysis of different subset communities proved the homogeneity of the network, validating and opening up further applications on these networks and communities. The dataset was then further processed with a focus on grouping users and segmented into two parts, one for the numerical features and one for the tweet content. Each segment was then put into a separate LSTM model to predict the frequency of tweets for each user at the last hour of the period, and the predictions were then combined using other machine learning models to create a final combined prediction using both textual data and numerical feature data.

1.1 Research Statement

This research focuses on analyzing and predicting trends in social media discourse by examining hashtag campaigns through the lens of network science. With social platforms like X (formerly Twitter) playing a central role in public communication, understanding how hashtags influence information spread and community formation is essential. This study applies community detection using Greedy Modularity Maximization, alongside multivariate LSTM models, to investigate hashtag propagation, user interactions, and the predictive dynamics of tweet frequency, providing a deeper understanding of how hashtags shape public sentiment and discourse.

1.2 Research Objectives

Data Extraction and Analysis:

- Data analysis on hashtag campaigns across the social media platform formally known as Twitter.
- Extracting meaningful features from the textual content associated with hashtags.
- Data processing for user activity forecasting in the hashtag domain.

Application of Graphs for Analysis:

- Implement and fine-tune the Greedy Modularity Maximization method to track the propagation of hashtags through tweets and detect communities by construction of graphs.
- Find community leaders after detecting communities and detect the sentiments they are spreading using AlBERT trained on SST-2.
- Analyze graph properties for the communities detected to understand the behavior of the communities.

Model for Prediction:

- Perform the required processing to then implement recurrent neural network models like LSTM capable of predicting the popularity of hashtag campaigns by forecasting the activity of users.
- Implement fusion ensemble models for text-based and numerical data-based model predictions using machine learning regression models like Random Forest, Gradient Boosting, and CatBoost to get better accuracy.

Performance Evaluation:

- Evaluate and compare the performance and effectiveness of the proposed models and methods using appropriate metrics and benchmarks.

2. Background

A thorough reading of many technical research articles related to hashtag propagation on social media was done to understand how hashtags spread as well as ways of predicting and analyzing hashtags. The papers also provided effective data collection options in order to find or create a dataset that fits research needs. The paper selection method was to consider relatable papers with large numbers of citations excluding very similar research. The was also a prominent focus on the recency of publications as well as ensuring modern and up-to-date technological focus. By arranging these chosen papers in chronological order according to the date of publication, a coherent body of knowledge is created that will support future study. Additionally, books and articles on Graph theory, network science, and Natural language processing were read to gain proper knowledge of the technical tools and their workings.

2.1 Literature Review

2.1.1 Network Science

Network science is the study of complex systems modeled as networks, where entities (nodes) are connected by relationships (edges). "Network Science" by Albert-László Barabási [9] is the strong foundation of network science that has brought the topic to where it is today. The book provides a comprehensive and in-depth understanding of network science and its inner workings as well as direct examples of applications of said knowledge in building networks, and graphs and analyzing the networks and graphs. The book also showcases detailed methods for creating communities in networks as well as the usage of a variety of tools, metrics, and techniques used to extract prominent features in said networks.
Network science is not only a theoretical study but is also highly applicable in various practical domains. In real-world scenarios, it can be used to understand social structures. By modeling users and their interactions, we can learn patterns, predict behavior, and analyze the social structure. In the context of social media, in a network of users connected through interactions like retweets network science offers valuable tools for analysis.

Graph component and community

Graphs are created through nodes and edges. When some nodes are interconnected through edges between them, those together create graph components. However, there can be densely connected nodes within a component, which create communities that can be detected by certain algorithms.

Greedy Modularity Maximization Model

After constructing the graph, this algorithm aims to detect communities based on the greedy modularity maximization method. Clauset-Newman-Moore's [1] greedy modularity

maximization is used to find the community partition with the largest modularity. In this model, clusters of users are identified who have interacted with each other by sharing a tweet. Modularity deals with the probability a random edge would fall into a module. It is a measurement that is used to divide a network into distinct communities based on their modularity. Networks with high modularity have dense connections between nodes within a module and sparse connections with other nodes of other modules.

Resolution

The resolution parameter in the Greedy Maximum Modularity Model (GMM) controls how much attention is given to the small or large communities while the algorithm runs. It is associated with the modularity score since higher modularity means more separation of communities whereas with lower modularity values there is less separation resulting in large communities (Chen, Kuzmin, & Szymanski, 2014) [7]. A lower value of resolution parameter that is less than 1, favors large communities detection over small ones resulting in outliers. On the other hand, a higher value greater than one is inclined to help detect the smaller communities better. Setting resolution at 1 is considered standard and can be tweaked to one's needs. However, tweaking too much may often lead to oversplitting. A higher resolution zooms in and provides attention to smaller communities whereas a lower resolution zooms out identifying big communities.

Mean Clustering Coefficient

According to Barabassi [9], the clustering coefficient measures the degree to which neighbors of a given node link to each other. The mean clustering coefficient is the average of the individual clustering coefficients of all the nodes in the community. Hence, when the neighbors of a central node are connected with an edge between them, those connections are measured to find the clustering coefficient. This metric expresses the connectedness of a graph because when the clustering coefficient is 0, there is no edge between neighbors of the central node. However, when it is 1, this means that all the neighbors have an edge between them making it a completely connected graph.

Average Degree

The degree is the sum of the links or edges for any particular node to other nodes, in other words, its connection to and from other nodes. According to Barabasi [9], in an undirected network the total number of links, L, can be expressed as the sum of the node degrees:

$$L = \frac{1}{2}\sum_{i=1}^{N} k_i$$

where N is the number of nodes in the network and k_i is the degree of node i.
The average degree, $\langle k \rangle$, is therefore simply the average of the degrees of all the nodes in the network. It can be calculated as:

$$\langle k \rangle = \frac{1}{N}\sum_{i=1}^{N} k_i$$

Average Path Length

In a physical system, the distance between nodes is crucial to understanding the network itself. In networks, the physical distance is not measured as in geographic maps, instead, the concept of "path length" is used, which represents the number of links (edges) you need to traverse to go from one node to another. Therefore, the shortest path in a network dictates the ease and efficiency of traversal across that network. It is the average number of steps or nodes that need to be crossed for movement from one node in the network to any other node in the network.
We calulcated the average path length using the formula from Barabasi [9]:

$$\langle d \rangle = \frac{1}{N(N-1)} \sum_{i,j=1, i \neq j}^{N} d_{i,j}$$

Where N represents the total number of nodes in the network, and $d_{i,j}$ denotes the shortest path length between node i and node j.

Centrality

Degree centrality, similar to average degree, is a measure of the importance of an individual node in the network which it does by counting the number of direct connections (or edges) it has within the network. Nodes with a higher degree of centrality are more "central" because they are directly connected to a larger number of other nodes. According to Rodrigues (2019) [17], degree centrality is needed where leaders influence other nodes directly. For example, in social media platforms, users with a high degree (most number of edges) are the most important characters. However, Rodrigues also notes that centrality cannot capture the indirect influences.

Modularity

Modularity is a probability measurement determining whether certain nodes are densely connected and can be placed in a module or not. It is calculated using the formula defined by Newman [1]. The modularity Q is defined as:

$$Q = \frac{1}{2m} \sum_{ij} \left[A_{ij} - \gamma \frac{k_i k_j}{2m} \right] \delta(c_i, c_j)$$

Where: - A_{ij} is the element of the adjacency matrix (1 if there is an edge between i and j, and 0 otherwise). - k_i and k_j are the degrees of nodes i and j, respectively. - m is the total number of edges in the network. - γ is the resolution parameter. - $\delta(c_i, c_j)$ is the Kronecker delta which is 1 if nodes i and j are in the same community ($c_i = c_j$), and 0 otherwise.

Density

Graph density is the ratio of the number of edges present in a graph and the maximum number of possible edges in that graph. The density formula for unidirectional graph is:

$$D = \frac{2|E|}{|V|(|V|-1)}$$

while for a directed graph, it is:

$$D = \frac{|E|}{|V|(|V|-1)}.$$

When the density is 1, that indicates a fully connected graph fulfilling all possible combinations, but a density of 0 means there are no edges in the graph. Density helps us understand how interconnected a network is in comparison to its possible connections.

2.1.2 Natural Language Processing

Natural Language Processing (NLP) is the area of research under artificial intelligence, which is about teaching machines to understand human language. To achieve this, NLP utilizes a combination of linguistics, computer science, and machine learning to convert human language into machine-feedable vector form in order to get machines to achieve desirable tasks. The book, Speech and Language Processing by Jurafsky and Martin [27] provides a a great introduction to the field of NLP.

Transformers and BERT

Transformers have been a revolution in natural language processing (NLP) as they are capable of considering the context of words in a text. BERT (Bidirectional Encoder Representations from Transformers) is an even more advanced model that was engineered to look at the words in a sentence in both directions, learning information of their context and capturing the meaning of semantics much better. In this research, a base BERT is applied to convert text contents into the numerical vector within up to 768 dimensions that carries much contextual information on word positional (Rothman, 2020) [22].

SST-2 (Stanford Sentiment Treebank 2)

SST-2 is a dataset introduced by Socher et al. (2013) [6] which dataset consists of movie reviews classified as positive or negative as binary sentiments. This has been used for training several AI models.

Albert (A Lite BERT)

ALBERT is a version of BERT (Bidirectional Encoder Representations from Transformers) trained by Lan et al. (2020) [21] on the SST-2 dataset after removing several parameters from BERT, it still performed acceptably well with an accuracy of 97.1%.

Word Lemmatizer

Lemmatization in Natural Language Processing (NLP) is the process of retrieving a word to its base form, which is known as the lemma. This makes the tokenization of the data much easier as it combines various versions of the same word into a single token. This research utilizes the NLTK (Natural Language Toolkit) library [28] in Python, particularly towards the data analysis of the text data (tweets).

LSTM and Time Series Analysis

Long Short-Term Memory (LSTM) implemented in this research is a recurrent neural network (RNN) which works well on sequential problems such as time series analysis and forecasting due to its memory keeping capabilities. Below is a simple and straightforward architecture in order to present a single LSTM unit. (Calzone, 2020)) [19].

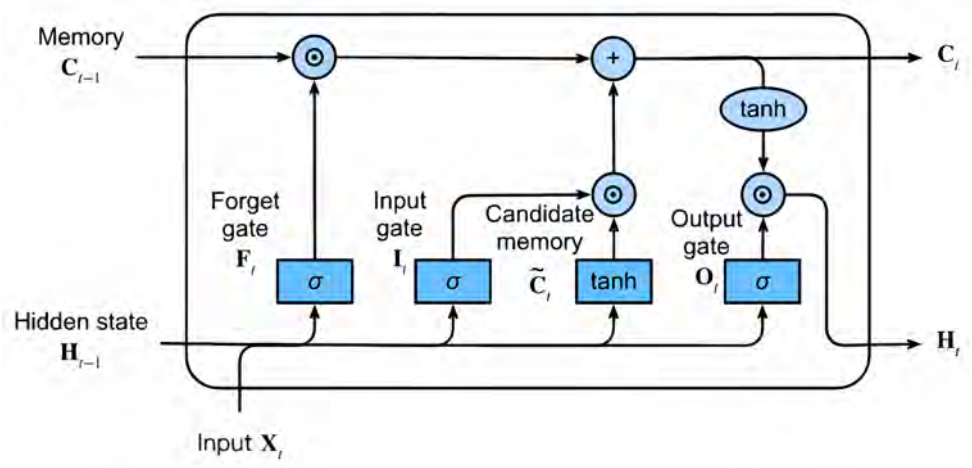

Figure 2.1: Inner LSTM unit architecture

The figure above shows a single LSTM unit that has several subunits called gates which are the main components of the LSTM unit. They help to retain important information in memory over the long term period.
The forget gate is responsible for removing the previous unit's dull information from memory using the sigmoid activation function which produces the values between zero and one. The equation for the calculations in the forget gate is as follows:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

where f_t is the forget gate output, W_f is the weight matrix, h_{t-1} is the previous hidden state, x_t is the current input, and b_f is the bias term.
The input gate decides which new information should be added to the cell state. This is also calculated using the sigmoid function:

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

where i_t represents the input gate output and b_i is the corresponding bias term.

The cell state of the LSTM helps to store information that are relevant and this is the component that ensures the LSTM has the specific information when needed. The candidate cell state is obtained through the hyperbolic tangent activation function which gives the output values of the inputs in the range of -1 to 1. The equation is as follows:

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

Here, $\tilde{C}_t$ is the candidate cell state and b_C is its bias term.
The cell state C_t is then updated by combining the previous cell state C_{t-1} with the new candidate values.

$$C_t = f_t \cdot C_{t-1} + i_t \cdot \tilde{C}_t$$

The equation showcases how the input gate and forget gates are used to update the cell state of LSTM unit.
Finally, the output gate gives the output of the LSTM unit. It determines what the next hidden state h_t should be based on the cell state:

$$h_t = o_t \cdot \tanh(C_t)$$

where $o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$ is the output gate.
Hence, LSTM can memorize information over long sequences and reduce the effect of vanishing gradient problem common to standard RNNs (Géron, 2019; Calzone, 2020) [16] [19].

Adam (Adaptive Moment Estimation)

Adam is a highly established optimizer that changes the learning rate to properly update weights during the training process in deep learning models. It is industry standard to use Adam for efficient computation [16].

2.1.3 Machine Learning

Machine learning is a branch of Artificial intelligence that deals with training models to learn by identifying patterns and making the models capable of making decisions to get desirable output.

Gradient Boosting Regressor

Gradient Boosting Regressor is a machine learning algorithm that uses weak learners like decision trees to solve regression tasks. This research uses the Gradient Boosting Algorithm for ensemble machine learning due to its capability to learn complex patterns from small amounts of data(Géron, 2019) [16].

Random Forest Regressor

Random Forest Regressor is another algorithm that builds multiple decision trees and averages their results for regression tasks. Unlike gradient boosting, Random Forest builds each tree independently by using random subsets of the training data (bagging) (Géron, 2019) [16].

Cat Boost Regressor

CatBoost Regressor is a version of the gradient boosting algorithm that can handle categorical features without extensive preprocessing and even without one hot encoding. It is briefly mentioned in (Géron, 2019) [16] and this research uses it as one of the models for ensemble learning.

One Hot Encoding & Binary Encoding

One-hot encoding is a technique used to convert categorical data into a format that machine learning models can understand. This is done by first separating categories into individual columns and then, for each instance, only one category is given the value of 1 to represent the instance having this categorical feature and all others are 0 (Géron, 2019) [16]. Similarly, binary encoding deals with only two category of values and encodes with the values 0 or 1.

Standard Scaler

The StandardScaler is a preprocessing technique that normalizes data to achieve a mean of 0 and a standard deviation of 1. This standardization is commonly applied to reduce the range of data, helping to prevent machine learning models from being biased by larger values during training models (Géron, 2019) [16].

$$X_{\text{scaled}} = \frac{X - \mu}{\sigma}$$

where X is the original data, μ is the mean and σ is the standard deviation.

Mean Squared Error (MSE)

Mean Squared Error (MSE), is a widely used metric for predicting regression tasks. It takes all the predicted values and the actual value and finds the mean of the differences, then squares this mean. (Géron, 2019) [16].

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

where n is the number of data points, y_i respresents the actual values, and $\hat{y}_i$ represents the predicted values.
MSE is also commonly used as the parameter for machine learning models to learn from during training.

Root Mean Squared Error (RMSE)

RMSE is the square root of MSE. This can be an upgrade over MSE in cases where larger errors can seem more problematic (Géron, 2019) [16].

Mean Absolute Error (MAE)

Mean Absolute Error (MAE) calculates the average of the absolute differences between predicted values and actual values. Taking the absolute mean make MAE less susceptible to outliers compared to MSE (Géron, 2019) [16].

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

Correlation (R^2)

R-squared (R^2) is a statistical measure that explains within a dataset, what percent train data variation is being properly captured to predict the test data. The values of R^2 will go between 0 and 1, where higher values signify a better fit of the model (Géron, 2019) [16].

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

where n is the number of data points, y_i represents the actual values, $\bar{y}$ is the mean and $\hat{y}_i$ represents the predicted values.

2.2 Related Works

A significant amount of research has been conducted over the years around social networks to analyze user interaction through hashtags. Much research has utilized Network science, ML, and NLP tools to explore various aspects of social media user behaviors. The popular research in this emerging field of Network science includes predicting hashtag popularity, sentiment analysis, hashtag recommendation models, and event or campaign promotion tracking while analyzing the role of hashtags in these propagation and movements.

Kywe et al. [2] proposes a novel hashtag recommendation method based on collaborative filtering to suggest relevant hashtags for a new tweet taking into consideration the user's preferences and tweet content. They started by analyzing a Twitter dataset generated by more than 150,000 Singapore users from October 2011 to December 2011. The dataset comprises 44 million original tweets, with a focus on understanding hashtag usage patterns. 65 thousand users wrote original tweets whereas the rest were retweets and other data. They use information such as user profiles, hashtags, and tweet content to derive insights into how hashtags are used in this specific user community. Even though 39% of the users used hashtags, less than 8% of the original tweets contained hashtags suggesting that people, despite knowing how to use hashtags, don't use them very frequently. Observing the dataset, they found that the life expectancy of hashtags is short as they are replaced with new trending ones each day. They propose a model that recommends hashtags based on a user's personality by gathering similar user tweet pairs which is a very simple yet interesting intuition. To do this, they followed three subtasks. Firstly, for selecting Hashtags from Similar Users, they perform TF-IDF to represent users, and using cosine similarity they get similarity between the target and another user from which based on scores similar users are chosen. After that, hashtags are selected from similar users based on frequency ranking. Similarly, for Selecting Hashtags from Similar Tweets they find similar tweets using the same method and find hashtags from those. Lastly, The candidate hashtags to be recommended for the target user and tweet can be obtained by the union of hashtags from top similar users and top similar tweets. Lucene, an information retrieval library, is employed to derive similarity scores and retrieve hashtags of top similar users and tweets efficiently. The recommendation method is based on collaborative filtering, combining user preferences and tweet content. They measure the performance of our method using the hit rate. Moreover, they have used the Python programming language and various libraries such as NumPy, SciPy, and sci-kit-learn to implement their proposed method. The proposed personalized hashtag recommendation method achieved its best performance with a hit rate of 31.56% for the top five recommended hashtags and 37.19%

for the top ten recommended hashtags. Their experiments showed that considering user preferences along with tweet content resulted in better hashtag recommendations compared to using tweet content alone. They also noted that user preferences from very few similar users significantly improved recommendation accuracy.

Muntean et al. [3] attempted to identify hashtag meanings using only computers without human assistance by associating the context of tweets and terms in tweets to hashtags used in tweets., which has huge applications in classification and clustering for social media algorithms to better target users. Various datasets were collected through the Twitter Streaming API for a period of three days and then 10 percent of this was randomly taken, which resulted in 280,000 distinct hashtags from approximately 900,000 daily tweets per dataset. Data preprocessing was done using the MapReduce paradigm from the Hadoop framework, where instances are mapped to key-value pairs and then the values are reduced. Only English tweets were kept and retweets were removed after cleaning (removing URLs, mentions, etc.). Hashtags were also lowercase and words with similar meanings were considered the same hashtag. Tokenization was done to create a dictionary and only hashtags with a document frequency over or equal to 10 were kept. To vectorize the hashtags, the vector space model VSM was used. The Lucene Analyzer class was used to remove stop words and words with certain patterns. Lowercasing and stemming were done using the Porter Stem algorithm. TF-IDF was used to give weights to the terms and the 2-norm form was used to normalize the vectors. K-means clustering was used while varying the number of clusters k (The greater the clusters the more specific, lower clusters mean more generalized). K-means calculations were done using the Jaccard distance. All of this was done using the Mahout library over a Hadoop single node for efficiency. The best number of clusters k was also found using canopy clustering. Results indicated the potential to cluster semantically related hashtags, as well as relate top terms of clusters with top hashtags.

Tsur et al. [4] used regression models with L1 regularization and stochastic Gradient Descent to predict the acceptance of hashtags on Twitter and to learn the optimal model parameters. They aimed to address whether content alone can accurately predict a hashtag's acceptance and explore how the structure of social networks and hashtags work together to make information spread effectively. The researchers used data from Twitter focusing on tweets containing hashtags. They collected over 400 million tweets from June to December 2009 using Twitter's API, ensuring a 5%-15% daily sample of public tweets. The data included hashtags, which are strings preceded by the '#' character, indicating the context or core idea of tweets. The researchers filtered out non-Latin characters to maintain an English corpus, though some non-English tweets remained. The dataset was subject to variations in sample size every week which introduced bias in data size. Hence, the researchers addressed this problem using normalization. They removed hashtags that occurred less than 100 times considering those to be typos or introduced at the end of the experiment date range. The researchers used a regression model to predict the acceptance of hashtags on Twitter. They employed a transformed target function by taking the log of the hashtag count to capture the magnitude of acceptance and to get a smoother representation of the temporal patterns in hashtag usage. Additionally, they implemented a linear regression model with L1 regularization to learn the optimal parameters for predicting the acceptance of hashtags. Regularization was used to reduce sparsity and the risk of overfitting. Stochastic Gradient Descent was employed to avoid

large matrix inversions and learn parameter values. The Nelder-Mead method was used to find the optimal values of hyperparameters in the regression model. The study employed a 10-fold cross-validation approach to assess the performance of the models. Support Vector Machine for Regression was used as well. The system utilized features categorized into four types: Hashtag content, Global tweet features, Graph topology features, and Global temporal features. These features included aspects like hashtag characteristics, tweet content, network structure, and temporal patterns. Support Vector Machine for Regression (SVR) was implemented as well which did outperform SGD. However, SGD was chosen as it is more efficient on high-dimensional vectors. The hybrid model containing the four types of features outperformed all partial models (HTall, TWcontent, Graph, Temporal). Again, Each basic model, such as content-based and global models, showed better performance than the baseline model. The baseline model showed no correlation, while the hybrid model demonstrated a correlation of 0.669, indicating that the linear regression model served as a good approximation to the actual model. The research demonstrated the importance of considering both content and graph topology in predicting the spread of ideas in online communities. The hybrid model, incorporating various feature types, proved to be effective in capturing the dynamics of hashtag acceptance.

Ma et al. [5] attempts to predict the popularity of hashtags that are used on a daily basis on Twitter, and therefore predict the popularity of new topics emerging on Twitter. They do this by trying to evaluate the features that make the hashtag popular, in this case, they look at two main features, the hashtag itself (content) and contextual features surrounding the hashtag (the community and user base using the hashtag, etc.). They have used data collected from around 2 million Singaporean users from which they got roughly around 31 million tweets from January 1, 2011, to August 31, 2011. They evaluated the tweets around 7 content features (sentiment, word/number, content, etc.) and 11 contextual features. (user count, tweet count, retweets, graph edges, etc.). They have also used a global user graph with 214,000 users (to incorporate social interactions between groups of users). So basically they evaluate the popularity of a hashtag at time t+1 and more once it emerges at time t. They classify each tweet into five categories - not popular, marginally popular, popular, very popular, and extremely popular. They formulate the problem as a classification task, and as such they use 5 classification algorithms - (Naive Bayes, K nearest neighbors(Euclidean Distance), Decision Trees, Support Vector Machines(linear kernel), and Logistic regression). They perform 10-fold cross-validation and then evaluate their results based on a number of evaluation metrics, eventually settling for the micro-F measure. Results demonstrated that contextual features are more effective than content features. Additionally, it was also found that analyzing the popularity of a hashtag over 2 days does not improve prediction metrics significantly over 1 day, which was theorized due to not being able to fully utilize the 2-day data.

Lim et al. [8] proposes a new model for aspect-based opinion mining (extracting reviews/opinions on products from social media) and sentiment analysis called the Twitter Opinion Topic Model (TOTM) to overcome the shortcomings of the standard LDA models in the automation and analysis of Twitter data such as lacking explicit scores, containing informal language, and often including emoticons and strong sentiment words. They have used 9 million tweets that give some form of opinion on electronic products as reviews. They removed non-English words using langid.py. They use Twitter NLP for POS tagging. They also perform normalization and other techniques such as decapitalization, and

removal of stop words, common words, and infrequent words. For sentiment evaluation, labels for sentiments are required which are missing from the original corpus, so they train the model for sentiment using a pre-labeled dataset called the Sentiment140 (Sent140) which contains 1.6 million tweets. They also evaluate this data using another dataset called SemEval which only has 6322 tweets but is better for evaluation due to being more accurate as the dataset uses annotation. They use a 90-10 split for training their model. They use LDA and ILDA as baseline models to compare and contrast with their new proposed TOTM model. A major advantage of TOTM over existing models is its ability to apply lexicons directly to specific words in specific situations as opposed to giving them a more general lexicon which helps to evaluate sentiment a lot more effectively. Additionally, they have also used a new method for integrating sentiment prior information into the topic model using a public sentiment lexicon, which is updated dynamically with the data. Analysis of results shows that there is value in mining opinions as it gives a larger and more encompassing opinion on electronic products as opposed to direct reviews despite all the additional noise. However, certain difficulties in the interpretation of tweets such as sarcasm and spam were acknowledged.

"Network Science" by Albert-László Barabási [9] is the strong foundation of network science that has brought the topic to where it is today. The book provides a comprehensive and in-depth understanding of network science and its inner workings as well as direct examples of applications of said knowledge in building networks, and graphs and analyzing the networks and graphs.

Gong et al. [10] in his research paper used CNNs with attention in order to solve the challenge of hashtag recommendation specifically automatically suggesting relevant hashtags for user-generated microblog content. The researchers utilized a microblog dataset containing 110,000 microblogs collected from platforms like Twitter. The content of the microblogs serves as the basis for training and testing the proposed model. The vocabulary in the dataset includes 106,323 words and 37,224 hashtags. Each microblog, annotated by users, includes an average of 20.45 words and 1.20 hashtags. The dataset was split into a training set (100,000 microblogs) and a test set (10,000 microblogs), with a 10% random subset used as the development set. Additionally, They trained their model on 10 million words from Sina Weibo. The research leverages Convolutional Neural Networks (CNNs) and an attention mechanism. CNNs leverage pre-trained word embeddings while the attention mechanism selectively focuses on trigger words within a small window. This combination enhances the model's ability to understand and recommend relevant hashtags. The networks take the input of the microblogs of different lengths and each dimension of the output layer is to get the probabilities of recommended hashtags. The model consists of two main components: a global attention channel and a local attention channel. Both channels process the input into feature vectors capturing relevant information. The global channel utilizes CNNs to process the entire microblog, employing convolutional layers with filters of varying window sizes. Multiple filters are used to get multiple features. Additionally, a max overtime pooling operation is executed over the feature map gained from applying a nonlinear function on all combinations of input's words to extract the most important feature from the map. The final output of the global channel represents the embeddings of the input. The local attention channel, on the other hand, generates a few trigger words by employing an attention mechanism to selectively consider words within a small window. Then scores of the words are calculated and ones greater than the

threshold are taken as triggered words. These then enter the folding layer which abstracts the features of these words resulting in the output of the local channel as embeddings of the triggered words. This attention mechanism allows the model to give more weight to contextually relevant words, simulating the importance of trigger words in the microblog content. The paper concludes by showcasing the effectiveness of the proposed CNN with an attention-based model with a precision of 0.443, Recall of 0.362, and F1 score of 0.398. The experimental results on real-world microblogging data demonstrate the effectiveness of this model and even CNN and attention alone, outperforming state-of-the-art methods. The paper highlights the importance of considering trigger words and the benefits of using a combination of global and local information.

Li et al. [11] explored the application of attention-based Long Short-Term Memory (LSTM) to improve how hashtags are suggested for social media posts, making it easier for users to categorize and find content to overcome existing methods with lackings. They used a massive dataset from Twitter, including over 185 million tweets from the latter part of 2009. Within this massive dataset, 16,744,189 tweets were identified with annotated hashtags. Among these, 500,000 tweets were randomly selected for training, and 50,000 each for development and testing. The dataset contains a variety of hashtags (27,720) and a large vocabulary (337,245 words). To encode the microblog posts for their models, the researchers used word embeddings. These embeddings were pre-trained on the original Twitter data from 2011, utilizing the word2vec toolkit. The proposed model, named Topical Attention-Based LSTM (TAB-LSTM), introduces an innovative attention mechanism that incorporates topic modeling into the LSTM architecture. The first part of the model is an LSTM-based sequence encoder where the microblog inputs each word's word embeddings are passed to get sequential output. LSTM uses its input, output, and forget gate to generate output h for each layer. The output layer has a sequence of hidden vectors collected from the LSTM outputs which contains information about the input microblog. In the second part, Latent Dirichlet Allocation (LDA) is employed to capture the topic structures of microblogs. It makes assumptions about topic distribution considering each microblog input has a hidden topic label from the distribution. In the last part of the model, the topical attention layer takes the hidden states from LSTM and the external topic vector from topic modeling and outputs a vector of the weighted sum of each hidden state. This final vector is added to a linear layer with an output of the number of hashtags. Then a softmax layer calculates the probability of possible hashtags. The model is trained in a supervised manner, optimizing the cross-entropy error of hashtag classification. They set the maximum sentence length to 40 words and the dimension of all hidden states of the LSTMs to 500. The training of these models involved using a stochastic gradient descent (SGD) algorithm with the Adam optimization method. Researchers found that their model TAB-LSTM works well. It is much better at suggesting hashtags compared to other methods they tested, like LDA and SVM. TAB-LSTM improves the results by more than 7%, showing that paying attention to specific topics in posts is a useful way to suggest hashtags. TAB-LSTB outperformed the state-of-the-art models with a precision of 0.503, a recall of 0.435, and an F1 score of 0.467. In essence, their approach of using a smart neural network with attention to topics significantly outperforms other ways of recommending hashtags, making it easier for people to organize and find content on social media.

Wang et al. [12] used hashtags and their characteristics to visualize the movement of

information during a networked social movement. They tried to do this using both new technology (virality framework and network modeling) as well as using comparative analysis to see the evolution of hashtag usage over a movement. They used data from over two days during the "Occupy Wall Street Movement" (OWS) from Twitter. OWS data was used as it reached a large audience very quickly through networking. To collect the required Twitter data, the GNIP PowerTrack service was used(third-party Twitter data reseller) instead of the Twitter API as it has access to more real-time tweets. They targeted the analysis of the Virality framework's bottom-up mechanism, which is how individuals spread information through social networks. Two valued, non-directional viral hashtag co-occurrence networks were constructed: one with the top 5 percent of hashtags from November 2; and one with the top 5 percent of hashtags from November 18. The paper uses network analysis to identify popular hashtags and hashtag co-occurrence patterns. The researchers then use the virality network and network modeling to uncover the characteristics of hashtags that have an influence on co-occurrence patterns on Twitter. They use degree centrality to measure prominence (degree centrality is just the number of edges; here edges is how many times a hashtag has been used with others). Exponential Random Graph Modeling(ERGM) was used to predict what type of hashtags were more likely to co-occur. The parameters fed to the ERGM were hashtag co-occurrence, frequency, and (each of 10 specific) hashtag types - (geolocation, movement, government, civil society, identity claim, public figure, economy, time, media outlet, event). Data analysis was done with the ERGM package in R. Gephi was used for network visualizations and descriptive statistics. (specifically the ForceAtlas 2 layout algorithm). Finally, they analyzed two proposed questions to check for the frequency of types of hashtags during movement and the likelihood of virality of hashtag type. Hashtag usage(in this case for the movement by movement participants) is used strategically to reach different social circles(influential actors, media outlets, etc.)

Varol et al. [13] focused on tackling the challenge of early detection of promoted campaigns on social media, particularly focusing on detecting when a particular trend or topic on social media is intentionally promoted, distinguishing it from organic, non-promotional content. Twitter provides an interface listing trending topics with promoted trends labeled at the top. The researchers obtained Twitter data by web-crawling tweets with trending hashtags at 10-minute intervals from January to April 2013 in the United States to collect both organic and promoted hashtag data for 927 hashtags. This dataset expands to 487 features in five different classes such as network structure and information diffusion patterns, content and language, sentiment, timing, and user meta-data. This dataset includes information about when tweets are posted, the content of the tweets, who is posting them, and how people are responding, comprising both promoted and organic trends, serving as ground truth. They considered retweeting, mentioning, and hashtagging co-occurrence networks to fetch the data. They retrieved tweets two days before to two days after each trend's peak to study the characteristics of each trend before, during, and after the trending point. The organic and promoted data lengths are kept imbalanced to observe realistic conditions. The feature selection process involved selecting the most predictive features for classification using a greedy forward feature selection method. The authors used K-Nearest Neighbor with Dynamic Time Warping (KNN-DTW) which is capable of dealing with multidimensional time series classification for discriminating between organic and promoted trends on social media. During the learning process, the algorithm was provided with training and testing sets generated by 10-fold cross-validation, and time series for each feature were processed

in parallel using DTW. The authors used a time series coarsening technique known as piecewise aggregation which led to a significant boost in efficiency through the reduction of processing costs and maintenance of marginal degradation in classification accuracy. In the evaluation step, the K-Nearest Neighbor (KNN) algorithm is used to assign a class score to a test trend. Each training trend is compared with the test trend using DTW, producing distances for each feature. The K nearest neighbors with the smallest DTW distances are selected, and the fraction of promoted trends among these neighbors is computed for each feature. The average across features yields the class score, indicating the probability that the test trend is a promoted campaign. The Area Under the Curve (AUC) is computed to assess model performance, and results are averaged across folds in cross-validation. The KNN-DTW classifier demonstrated exceptional performance in online campaign detection using multidimensional time-series data from social media. Outperforming baseline models, including SAX-VSM and traditional KNN, KNN-DTW exhibited high accuracy, as indicated by consistently high Area Under the Curve (AUC) scores.

Reyes-Menendez et al. [14] tried to identify Twitter user sentiment on a multitude of factors such as social, economic, environmental, and cultural factors related to the sustainable care of both environment and public health using sentiment analysis. 5873 tweets that used the hashtag #WorldEnvironmentDay and were published in English on the 5th of June 2018 were collected. Initially, there were 9467 tweets which were reduced after cleaning by removing retweets and other relevant things. The tweets were collected from the Twitter API, and additionally, gender data was also recorded from the API (number of male and female tweets) as well as information on the countries that the tweets originated from. They used the MonkeyLearn algorithm in Python for sentiment analysis where training was done using 732 samples divided into positive, neutral, and negative samples. The entire data was then worked on using NVivo Pro 12 software for categorization while aiming for a success rate of over 0.650%. The algorithm used for sentiment analysis was SVM. The study resulted in determining the key environmental and health factors that Twitter users are concerned about such as poverty, education, energy, etc.

Turker et al. [15] tried to use a complex approach to create a multilayer network analysis of hashtags in Twitter, where the first layer examines the co-occurrence and the second layer examines the semantics. Data was collected from the Twitter Streaming API from 1 October 2015 to 20 January 2016 to get 9,320,567 tweets. Important to note is that the data used was Turkish tweets. Of these, only 1,402,618 had hashtags in them with only 104,492 being unique. Due to being Turkish letters, two main preprocessing steps were taken(All tweets were lowercase, and Turkish letters were converted to their non-Turkish counterparts.) showcasing how a different language required different steps compared to the English language. They also removed URLs, emoticons, punctuations, user names, etc. They used the open-source php-based 140 dev framework to collect Twitter data. They used two main layers for the network, a co-occurrence layer where multiple hashtag tweets were allowed and a semantics layer where only one hashtag per tweet was allowed. Additionally, it also used a third connecting or intersection layer between the two main layers. For the weights or edges, they use TF-IDF alongside cosine similarity to give weights to the edges of the network. Finally, they used Gephi for network analysis. As this is an analytic paper, it further goes into deeper territory explaining the characteristics of the network. The multilayer network for hashtags displays the same universal characteristics

as other real networks, showcasing power law consistency.

Zhang [18] aims to address three main research questions related to understanding hashtags on Instagram: temporal-spatial patterns, semantic displacement, and the potential for using hashtags to infer social relations. They propose a bipartite graph embedding model to summarize a user's hashtag profile and predict friendship based on this. They collected three large datasets from Instagram, comprising over 7 million hashtags shared over a span of 5 years. They used Foursquare's API to obtain the location categories among New York, Los Angeles, and London of posts and Instagram's API to map the locations to corresponding IDs. Using Instagram's public API, he obtained 51,527 Instagram IDs with over 39.5 million Instagram posts of which 7 million consisted of hashtags. They processed the data by categorizing hashtags into clusters based on temporal patterns, exploring spatial patterns using Foursquare location data, and studying semantic displacement through the skip-gram model with negative sampling. They also used natural language processing techniques to extract the semantics of the hashtags. They employed several models to conduct their analysis on Instagram hashtags. Firstly, K-means clustering was used to identify different temporal patterns among hashtags. Secondly, to investigate spatial patterns, four square data was used to categorize locations into more than 300 types from which the top 10 location categories in each dataset were selected for analysis. Thirdly, to analyze if hashtags exhibit semantic displacement, the researchers used the Skip-gram model with negative sampling to represent hashtag semantics and then they used the cosine distance of a hashtag's two vectors from the span of 5 years of that hashtag to measure a single semantic displacement. Lastly, to infer friendship from hashtags, a bipartite graph embedding model was used to learn hashtag profiles. The continuous bag-of-words (CBOW) model, a shallow neural network, was applied for profile learning. The optimization objective involved maximizing the likelihood of nodes' neighborhoods based on random walks. Negative sampling was utilized for efficient model training. The proposed bipartite graph embedding model achieved effective friendship prediction with an AUC above 0.8 in all three datasets, outperforming several baseline models by 20%. They found that hashtags exhibit diverse temporal patterns and semantic displacement.

Cruickshank et al. [20] introduces a novel multi-view clustering technique with multiple data types to deal with the fast-evolving discussion topics over the course of the pandemic. The data is collected from Twitter's streaming API and includes tweets collected from February 1, 2020, to April 30, 2020 during the pandemic, comprising over 300 million tweets with 45,000 unique hashtags used per day. Because of such a large dataset, the data went through a lot of preprocessing steps. Firstly, it was divided into days. Next, they removed any hashtags with too few uses(<3) and then for all remaining tweets removed everything except the critical content(hashtag and other symbols and punctuations, URLs, mentions, etc.). The clustering is done using a technique called multi-view modularity clustering(MVMC) after converting all the data into graphs by using a network modularity technique as the clustering function. The graphs are created with similarity in mind, as such the raw data is first converted to TF-IDF scores and then a cosine similarity is calculated to measure the similarity between hashtags. As there are multiple views of the data, a graph for each view needs to be created, and KNN(K-nearest neighbors) was used for making the graphs. For calculating modularity they have used two algorithms- one for handling the MVMC process itself and one for calculating the edge propensities. The authors get hugely diverse and detailed results of Twitter discourse over a long time period.

They get multiple clusters detailing different topics that Twitter users were concerned with over different time periods during the pandemic. Overall, MVMC proves to be a very effective algorithm for dealing with extremely large datasets over periods of time.

Antonakaki et al. [23] seeks to present the patterns of social graphs, sentiment analysis, and the impact of malicious activities on Twitter using graph sampling, natural language processing, and machine learning. The primary data source is Twitter itself. The researchers used Twitter data consisting of "tweets," each containing up to 280 characters. The paper mentions six prominent features of Twitter's data model, which are hashtags, trends, retweets, mentions, replies, and URLs. These features play a crucial role in understanding user behavior, content propagation, and interactions within the Twitter network. Twitter had an open approach, allowing extensive data collection. However, due to concerns about misuse, stricter rate limits were enforced in 2012. Twitter introduced paid data access plans with more relaxed limits, categorized as 'Premium' and 'Enterprise' APIs, providing access to data from the last 30 days or even as early as 2006. Twitter's API returns data in JSON format, prompting a preference for NoSQL databases like MongoDB. They used the Random Walk technique to download a subset of Twitter's social graph containing 13.2 million users and 8.3 billion edges and used the igraph library to measure the time required to assess a variety of graph properties. The paper suggests that sentiment analysis is one of the promising methods for content analysis in social media which involves NLP. Moreover, machine learning is one of the computational techniques used in areas such as graph analysis, sentiment analysis, and threat detection. In order to perform a general classification workflow for Twitter data with machine learning methods they presented 6 steps starting with data collection using Twitter API and later labeling those based on the classification problem type. After this, the data is split into test, validation, and training portions from which then different features are extracted. In step 5, external knowledge is added to the data. For example, in sentiment analysis add sentiment lexicons and vocabularies. In the final step, the data is trained in a machine learning model such as Decision Trees, Naive Bayes, Random Forest, etc. Finally, accuracy is measured using different matrices to ensure accuracy, sensitivity, and efficiency. Overall, the paper aims to provide a clear conceptual model of Twitter and act as a guide to expand further the topics presented, rather than presenting new research results.

Peterson et al. [24] attempted to gain insight into Twitter communities' opinions on COVID-19 using data science and NLP libraries, aiming to support health communication by clearly identifying relevant hashtags containing useful information. A total of 28.5 million tweets were retrieved, of which 6.9 million had hashtags. Data retrieval was performed using a keyword-based search with tweet IDs. Python was the primary language used for analysis, and the panda's library provided statistics for qualitative analysis. Preprocessing was conducted using spaCy. The analysis revealed multiple themes (Covid-19, social media, healthcare, politics, etc.). Furthermore, detailed co-occurrence graphs were created to display hashtag co-occurrence and identify distinct relationships between themes.

Collados et al. [25] introduced a cutting-edge Natural Language Processing tool called TweetNLP to support a diverse set of NLP tasks on social media data. TweetNLP utilizes datasets sourced from Twitter for training and evaluation. The core datasets include SemEval tasks on Affect in Tweets (2018), Emoji Prediction (2018), Irony Detection (2018), Hate Speech Detection (2019), Offensive Language Identification (2019), Stance Detection

(2016), Topic Classification (2022), and Named Entity Recognition (NER). These datasets cover a range of social media language challenges, allowing the models to learn and adapt to diverse contexts and tasks. The Affect in Tweets (2018) dataset simplifies emotions into anger, joy, sadness, and optimism. Emoji Prediction (2018) involves predicting the final emoji in a tweet, while Irony Detection (2018) is a binary classification for a tweet's irony. Hate Speech Detection (2019) targets hateful content, and Offensive Language Identification (2019) identifies offensive language. Stance Detection (2016) evaluates tweet stances toward a topic. Topic Classification (2022) assigns topics to tweets, and Named Entity Recognition (NER) annotates entities in tweets with seven types. This diverse range of tasks ensures that TweetNLP is well-equipped to handle the multifaceted nature of Twitter communication. The technology employed in TweetNLP includes fine-tuned transformer-based language models, which are based on RoBERTa and XLM-R architecture and pre-trained on Twitter-specific corpora. Following this architecture, TimeLMs (pre-trained on 132M tweets) were used for English and XLM-T (198 M tweets) as a multilingual model. The models are trained using fastText for word embeddings and contrasting embeddings with an InfoNCE loss for tweet embeddings. Both XLM-T and TimeLMs models were trained using the fastText package. By using fastText, TweetNLP gains the ability to analyze and interpret the meaning of words in tweets, enhancing the accuracy of various language-related tasks. TweetNLP provides an integrated Python library that serves as a modern toolkit for supporting social media analysis. In conclusion, TweetNLP's default models showed better performance than general-purpose models. The default TimeLMs-21 model achieved strong results across multiple tasks, emphasizing its suitability for social media NLP challenges. Furthermore, the TweetNLP (XLM-T) model outperformed fastText and XLM-R in all different languages in Sentiment analysis results with the Macro-F1 metric. TweetNLP not only provides valuable insights into multilingual sentiment analysis, word embeddings, and tweet retrieval tasks but also emphasizes transparency by acknowledging potential biases in social media data.

To the best of our knowledge, no research has been conducted to analyze social trends using community detection, user network analysis of user network, and leader sentiment as well as to predict the popularity of trends by training complex models to learn user activity.

3. Dataset

The dataset contains tweets and retweets collected from Twitter over three consecutive days starting from the 15th of September to the 17th of September 2022 on discussions about cryptocurrency. This dataset consists of 337702 tweets and retweets. For each row, there are 19 columns which are 'Tweet Id', 'Tweet URL', 'Tweet Posted Time', 'Tweet Content', 'Tweet Type', 'Client', 'Retweets Received', 'Likes Received', 'Tweet Location', 'Tweet Language', 'User Id', 'Name', 'Username', 'User Bio', 'Verified or Non-Verified', 'Profile URL', 'User Followers', 'User Following' and 'User Account Creation Date'.

3.1 Data Preprocessing

The dataset had many columns and data that were irrelevant to the research and formats that needed to be changed. Hence, in order to prepare an analysis of the data, data preprocessing methods were applied to bring the data to an analyzable and legible format.

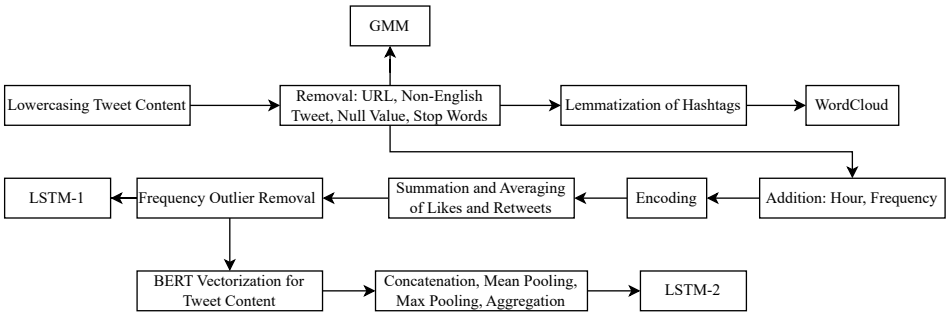

Figure 3.1: Flow Chart of preprocessing

3.1.1 Processing for Analysis & Interaction Graph

Since the dataset was extremely large, it was divided into 6 parts and required concatenating the parts to create a complete dataset. Afterward, all Tweet Content was lowercase to avoid repetitions or duplicates. Then all URLs were removed from the dataset in the 'Tweet Content' column as it was concluded that they would be irrelevant to establishing connections between hashtags. Moreover, all tweets in any other language aside from the English language were removed to maintain consistency and ease of use. Additionally, the date and time were extracted from the 'Tweet Posted Time' column and configured it into a DateTime format, and kept in a new column 'DateTime'. All tweets with null values in the 'User Followers', 'User Following', 'Retweets Received', and 'Likes Received' columns were also removed as it was judged to be an inconsequential amount of data, removal of which only helped in maintaining consistency and it was further ensured that the values in said columns were in integer format to maintain data cohesion during operations. The NLTK word lemmatizer was also used to lemmatize hashtags, ensuring plural and slightly different versions of the same hashtag are considered the same and only the first instance of hashtags in individual tweets are kept so as to ensure that a single tweet can only count

as one instance of hashtag usage even if the hashtag appeared multiple times in said tweet. Additionally, for certain analyses such as creating a word cloud to extract impactful words related to the event, stop words were removed using NLTK stop words. At the last stage of preprocessing, some columns such as 'User ID', 'Tweet URL', 'Tweet Posted Time', 'Tweet Language', 'Tweet Type', and 'Tweet Location' were removed as there were a minuscule number of tweets containing location data meaning they would not be very useful to the overall research, 'Name' as the twitter usernames were enough to classify users, 'Profile URL' and 'User Creation Date' because these were only needed for the preprocessing and won't be helpful in the later parts of the research. After the preprocessing, the dataset now has 253516 rows dropping 84,186 tweet rows and 11 columns down from 19. Below in the figure [3.1], the columns and one row of data is shown after preprocessing.

Tweet Id	Tweet Content	Client	Retweets	Likes	Username	User Bio	Verified	Followers	Following	Datetime
"1570926 713571.."	"ethereum officially merged moved towards pos..."	"Twitter for Android"	0	0	plusendless	"#Bitcoin value = ∞ supply limited to 21 million..."	Non-Verified	128	432	2022-09-17 00:04:49

Table 3.1: Hashtag data frame after preprocessing

3.1.2 Processing for Activity Prediction

In order to prepare the data for the prediction model LSTM's input, several additional processing steps were taken. Firstly, we analyzed the range of hours of data we had as we needed to know the hour range for prediction which was 39 hours total from 2022-09-15 (09:12:23) to 2022-09-17 (00:04:53). Hence, we created a column named Hour that starts its count from 0 to 38 where instances were assigned with its corresponding hour acknowledging when the tweets were tweeted. Encoded columns such as the 'Verified or Non-Verified' column using binary encoding method and the 'Tweet Type' column using the One-Hot-Encoding method for 'ReTweet', 'Reply', and 'Tweet. Furthermore, we calculated the frequency of tweets for each unique user per hour and added that in a new column named "Frequency". This column represents the number of tweets by each person per hour. For multivariate time series analysis, we kept the last instance for each user and removed the previous ones as we felt the last instance of features in an hour was an adequate summation of a user's features in that hour. However, before removing the previous entries, essential information such as Likes and retweets were summed up and added to the last instances. Apart from that, the average of both Likes and Retweets per hour for each user was calculated and added as a new 2 column to the dataset. Other columns such as followers, user followings, verified or not verified held enough information in the last instance so those were kept as they were.

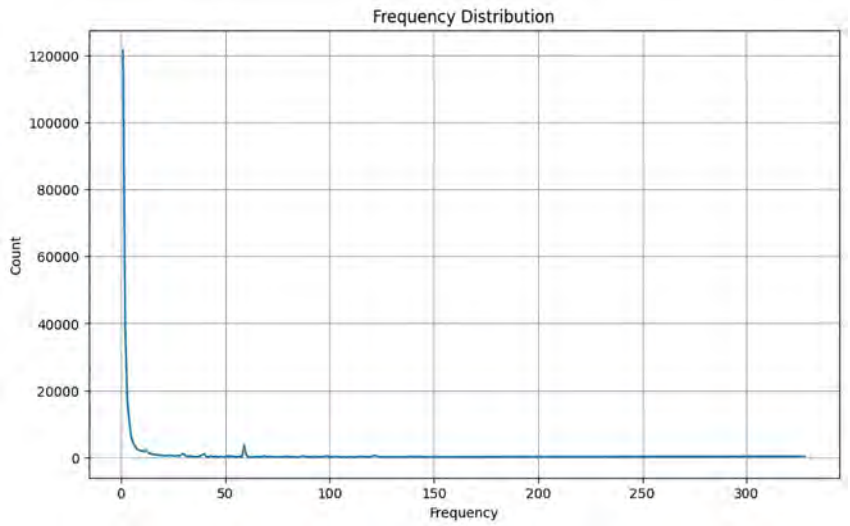

Figure 3.2: Frequency Distribution

Afterward, we analyzed the frequency count with the plot [3.2] to identify the outliers. We chose to remove instances with a frequency above 100 in an hour as we found that the amount of data with a frequency over 100 only constituted around 4 percent of values and either way it seemed unlikely for a real person to tweet that many times in a single hour. Hence, we decided to remove those outliers. However, for the time series forecasting using tweet content, we kept all of the tweets in each hour and then used BERT to vectorize the tweets. To handle multiple tweets in a single hour, a variety of different approaches were taken. We tried concatenating the tweets for each hour before vectorization for one model, and for the other models, we vectorized the tweets first and then performed either element-wise mean pooling, max pooling, or aggregation, all of which were evaluated to find the best approach. Lastly, removed redundant columns for this task and kept the ones given in the table [3.2].

User Id	**bert_embedding**	**Retweets Received**	**Likes Received**	**Verified Encoded**	**User Followers**	**User Following**	**Hour**	**Tweet Type Encoded**	**Average Likes**	**Average Retweets**	**Frequency**
”100015 7817528 049664”	[4.39e-01 ..5.60e-01]	0	0.0	0	954	3872	0	1	0.0	0.0	0.0

Table 3.2: Numerical data for LSTM after preprocessing

3.2 Analysis

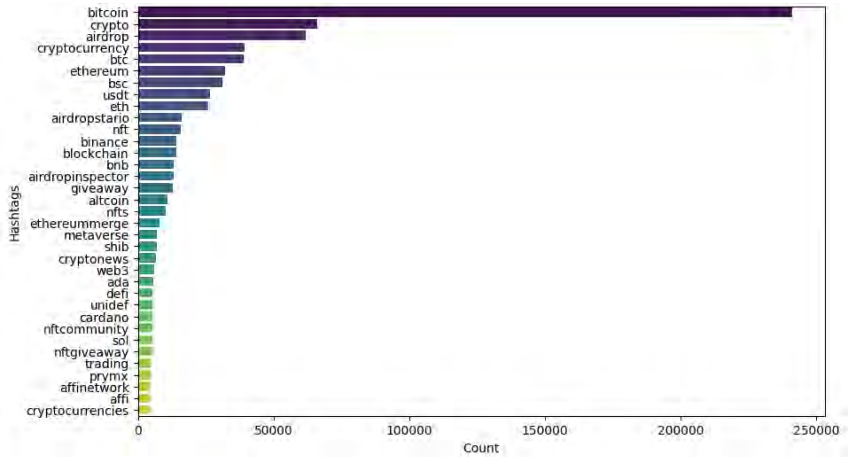

Figure 3.3: Top 35 Hashtags

The figure [3.3] represents the top 35 hashtags in our dataset and the number of occurrences of each hashtag which helps determine the popularity of each hashtag. As seen in the figure, #bitcoin has occurred more than 200 thousand times which is significantly higher than any other hashtag. Being the most used hashtag, it can be hypothesized that this hashtag has a higher propagation rate. Another cryptocurrency #airdrop has between 50 to 100 thousand tweets which is also very popular. Other coins such as #ethereum and #usdt have occurred between 25 to 50 thousand times as these are also moderately in demand and talked about. Other hashtags used are significantly unpopular in our dataset

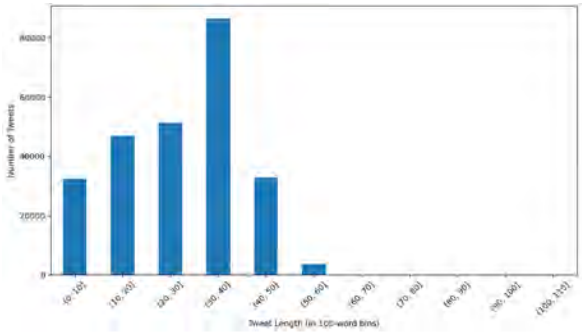

Figure 3.4: Tweet length Distribution

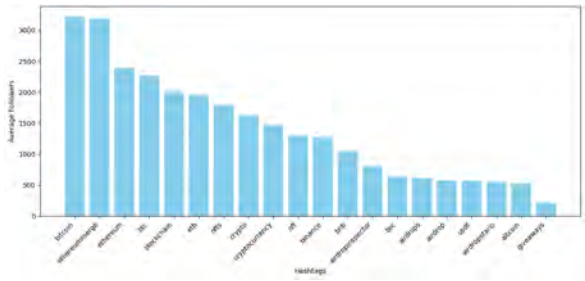

Figure 3.5: Average followers of Users per hashtag

As seen in figure [3.4], the dataset consists of tweets of a maximum of 50-60 words. However, the 30-40 word range consists of the highest number of tweets exceeding 80000 total tweets. This figure indicates that an average user usually writes 30-40 words per tweet. This analysis gives us an understanding of one of the characteristics of general users which is a handful for understanding user behaviours.

The figure [3.5] contains follower counts of users who have used the top 20 most popular hashtags in our dataset. To elaborate, it represents the average number of followers for

users who have used the hashtags in the x-axis. For instance, if 5 people used #bitcoin and their total follower count combined is 30, the average follower count will be 30/5=6. This graph will help us understand if highly followed users can influence the propagation of hashtags. As seen from the figure, users who have used #bitcoin have the highest number of followers in total which is above 3 thousand. This is decently higher than other hashtags such as bnb, airdrops, usdt, eth where the follower count is below 1500. This analysis provides an understanding of another parameter which is user followers total, that may drive the popularity of hashtags.

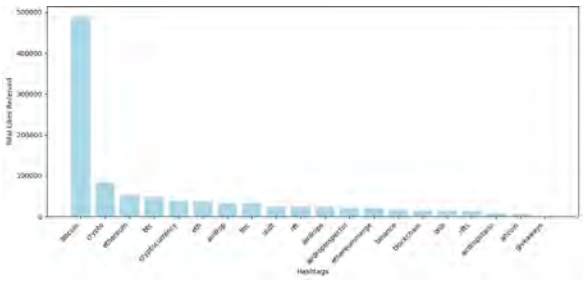

Figure 3.6: Total Likes Received for the top 20 Hashtags

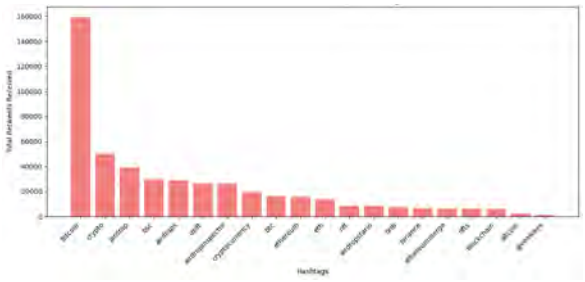

Figure 3.7: Total Retweets Received for the top 20 Hashtags

The figure [3.6] shows the total number of likes received for tweets containing the top 20 most-used hashtags. As seen in the figure, tweets containing #bitcoin have the highest number of likes in total which is above 4500000 that indicates that people enjoy to engage in these topics related to bitcoin directly, and like to express their emotions towards these tweets. However, different cryptocurrencies other than bitcoin such as ethereum, btc, bnb, usdt have not gained as much popularity or engagement from people as those tweets have total likes below 100000. Hence, people are more drawn to Bitcoin and propagate this more than other cryptocurrencies.

The figure [3.7] is a significant plot for understanding and determining the interconnectedness and propagation of hashtags. This graph plots tweets containing the top 20 most used hashtags against their retweet count. As can be seen, tweets containing #bitcoin have almost 160000 retweets showing that this hashtag has propagated the most because people were more likely to be familiar with and engage in topics related to bitcoin. Additionally, tweets containing“#crypto” and “#airdrop” have a significantly lower but decent amount of retweets between 30-50 thousand. These hashtags have influenced the propagation of tweets related to Bitcoin as well. This number of retweets denotes the interconnectedness between the users who posted the main tweet and the users who retweeted those, where the retweet count acts as an edge between the two users. This is a very important parameter to further determine the propagation of hashtags.

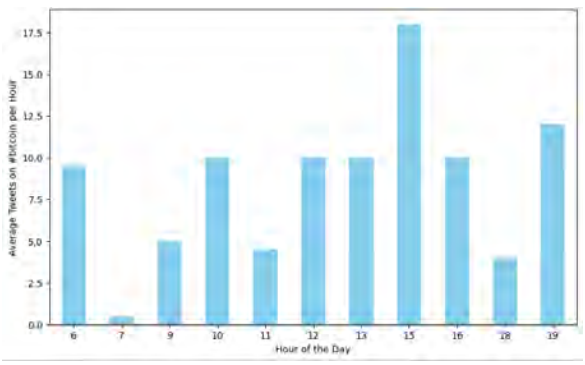

Figure 3.8: Average Tweets on #Bitcoin per hour by user kosehknn

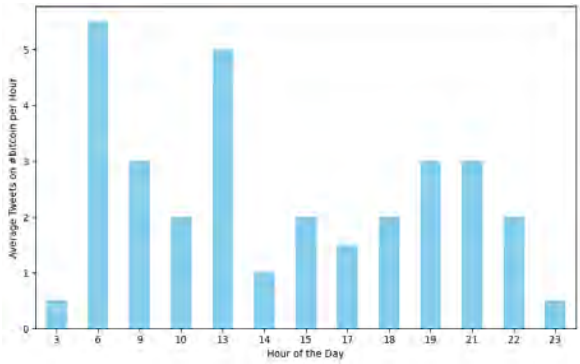

Figure 3.9: Average Tweets on #Bitcoin per hour by user MonroeVickyy

Figure [3.8] represents the 24-hour tweet activity of the user "kosehknn" who is one of the most active members in the dataset in case of using "#bitcoin". He had posted 18 tweets at 15 o'clock which was his highest tweet in an hour on Bitcoin. On his activity timeline, he was idle for 13 hours, and in the remaining 11 hours, on average, he posted between 5 to 10 tweets from 9 am to 1 pm each hour. These activities of users help us understand random user behavior and the time gaps between their activities.

Another user "MonroeVicky" was active for 13 hours of the day however he has tweeted a maximum of 5-6 times in an hour, shown in figure [3.9]. He was mostly active from 1 pm to 11 pm where he tweeted 2-3 times on average each hour except at 16th and 20th hour of the day. He shows a more consistent behavior on Twitter.

Figure 3.10: WordCloud

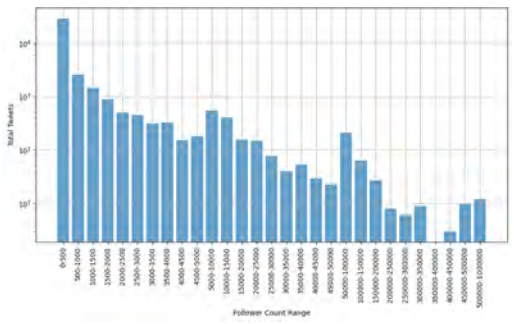

Figure 3.11: Total Tweets By Follower Count Range

The word cloud [3.10] shows the most commonly used words in our tweet contents excluding the stopwords, symbols, emojis, urls and unprintable characters as well as the hashtags. The word cloud dedicatedly represents the relevant non-hashtag words that were used many times alongside the hashtags related to cryptocurrency. Words like price, reward, link, and telegram are used to express cryptocurrency trading and to communicate with others to convince them to trade as well because there exist many telegram channels that provide price prediction of coins. Influencing others to pursue trading is a boosting factor to the propagation of the hashtags related to cryptocurrency as many find these financially profitable topics interesting. Additionally, there are many names of cryptocurrency used such as airdrop, ship, btc, and ethereum which portrays that these words and coins were

used many times with the hashtags so these are significant in the propagation.

The figure [3.11] presents the total number of tweets by users with different follower ranges which illustrate how people with different popularities propagate topics related to crypto. To show a better comparative analysis, the y-axis is represented in a logarithmic scale demonstrating that in tweets, people within the range of 0-500 have the highest number of tweets. Here, users with 0-500 followers have the highest count of total tweets which is above 10^4(175 thousand) whereas people with 5 to 10 thousand followers have an exponentially lower amount of total tweets which is actually below 1000. Here, the plot shows that most common people with low followers are interested and more likely to engage in conversations related to crypto and have a major role in the propagation of hashtags related to this currency. Moreover, as the number of user followers increases, the tweet count decreases as well which is an interesting analysis that represents that due to less number of people being there with high followers, the number of tweets can be low signifying that the general population is more active on Twitter in the discussions on crypto. It can be further deduced that users with higher numbers of followers showcase celebrity or influencer behavior in that they engage in conversations a lot less and only make occasional tweets and very few retweets.

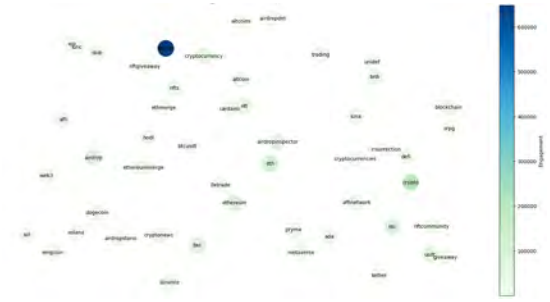

Figure 3.12: Top 50 Unigram Hashtags by User Engagement

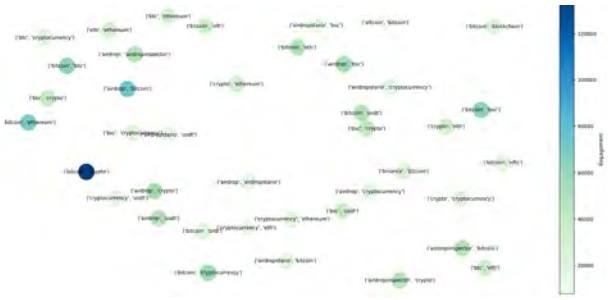

Figure 3.13: Top 40 Bigram Combinations by User Engagement

The above figure [3.12] highlights the user engagement that hashtags create through calculating the sum of the occurrences of likes and retweets of tweets containing the specific hashtag. The data is visualized for the top 50 hashtags where nodes are hashtags and the system is color-coded such that higher engagement hashtags are deeper in color. The table [3.3] showcases the top 5 such hashtags and numerical and average values for engagement. As seen in both the graph and the table, #bitcoin creates significantly more engagement than the other hashtags, with #crypto and #eth being engaging enough to showcase a deeper color but still nowhere near bitcoin. However, looking at the average values which are the total engagement each hashtag is generating divided by the number of times that hashtag is used, we can see that #eth is actually generating more engagement on average each time it is used compared to the other hashtags.

Hashtag	Total Retweets	Total Likes	Total Engagement	Avg Retweets	Avg Likes	Avg Engagement
#bitcoin	159470	489298	648768	0.669704	2.054838	2.724542
#crypto	68954	116515	185469	1.048284	1.771337	2.819621
#eth	26605	77515	104120	1.028451	2.996444	4.024895
#ethereum	18761	60251	79012	0.584673	1.877680	2.462354
#airdrop	39063	34356	73419	0.826713	0.727096	1.553808

Table 3.3: Engagement statistics for top hashtags

Similarly, analysis is done on bigram combinations of hashtags in the above figure [3.13] which highlights the user engagement that bigram combinations of hashtags created by calculating the sum of the occurrences of likes and retweets of tweets containing the specific combination. The data is visualized for the top 40 hashtags where nodes are bi-gram combinations and the system is color-coded such that higher engagement combinations are deeper in color. The table [3.4] showcases the top 5 such combinations and numerical values for engagement. Yet again, it is noticeable that even in combinations, tweets containing #bitcoin in combination with another # still generate more engagement, but now other combinations are also showcasing darker colors meaning combinations of other hashtags are also starting to create more engagement, leading to the hypothesis that combinations of hashtags do start to generate more engagement. Even though overall engagement is down, this can be chalked up due to combinations of hashtags appearing less frequently than individual hashtags appearances. However, looking at the average engagement, it seems that combinations of 2 hashtags not only fail to generate more engagement per use, it also seems to lower the engagement a little, which perhaps show that combinations of 2 hashtags don't really affect engagement all that much.

Combination	Retweets	Likes	Total Engagement	Avg Retweets	Avg Likes	Avg Engagement
#bitcoin & #crypto	49605	83562	133167	0.769022	1.295455	2.064477
#airdrop & #bitcoin	39037	34303	73340	0.957047	0.840987	1.798034
#bitcoin & #ethereum	15500	52901	68401	0.485376	1.656573	2.141949
#bitcoin & #bsc	29646	33540	63186	0.954844	1.080263	2.035107
#bitcoin & #btc	15368	47166	62534	0.413975	1.270533	1.684508

Table 3.4: Engagement statistics for hashtag combinations

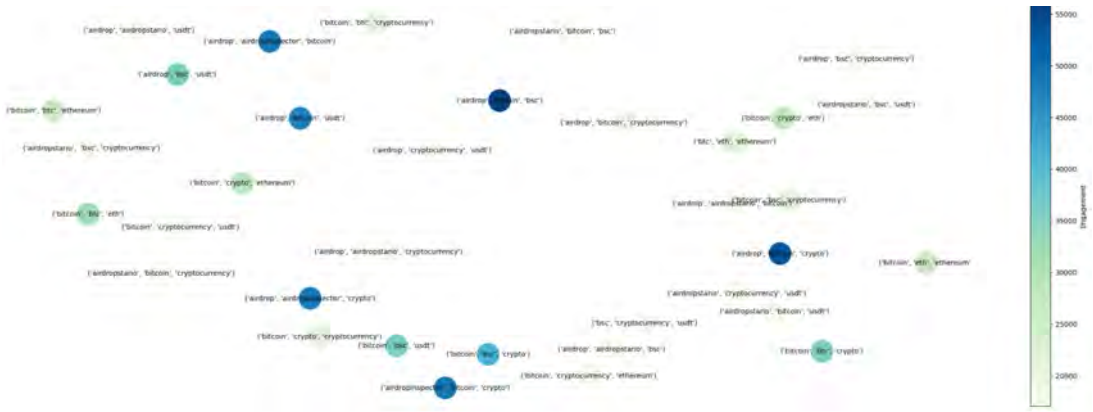

Figure 3.14: Top 35 Trigram Combinations by User Engagement

Finally, the analysis on trigram combinations of hashtags in the above figure [3.14] highlights the user engagement that trigram combinations of hashtags create through calculating the sum of the occurrences of likes and retweets of tweets containing the specific combination. The data is visualized for the top 35 hashtags where nodes are trigram combinations and the system is color-coded such that higher engagement combinations are deeper in color. The table [3.5] showcases the top 5 such combinations and numerical values for engagement. Here, finally we start to see that in trigram combinations, bitcoin is combinations are rivaled very closely by other combinations not including bitcoin, further encouraging our hypothesis on combinations creating more engagement. This is especially proved looking at the average engagement values from the table, which on average boasts significantly

more user interaction per usage compared to single hashtags or bigram combination of hashtags.

Combination	Retweets	Likes	Total Engagement	Avg Retweets	Avg Likes	Avg Engagement
#airdrop & #bitcoin & #bsc	27269	28663	55932	1.035270	1.088193	2.123462
#airdrop & #bitcoin & #crypto	29707	23610	53317	1.817387	1.444390	3.261777
#airdrop & #airdropinspector & #bitcoin	26291	22207	48498	2.011707	1.699212	3.710919
#airdropinspector & #bitcoin & #crypto	26289	22203	48492	2.012324	1.699556	3.711880
#airdrop & #airdropinspector & #crypto	26289	22203	48492	2.011554	1.698906	3.710460

Table 3.5: Engagement statistics for hashtag combinations with three hashtags

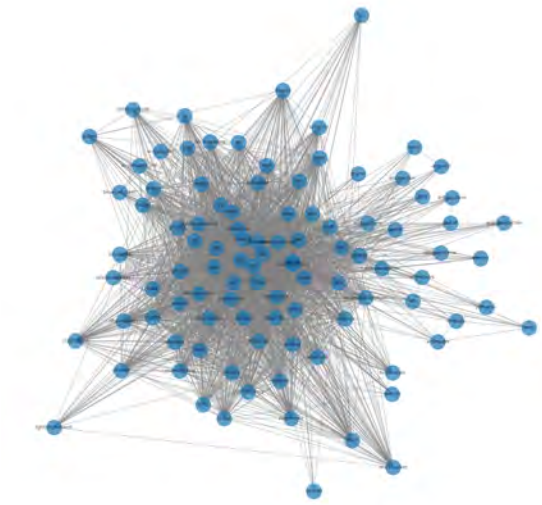

Figure 3.15: Hashtag co-occurrence of Top 100 hashtags

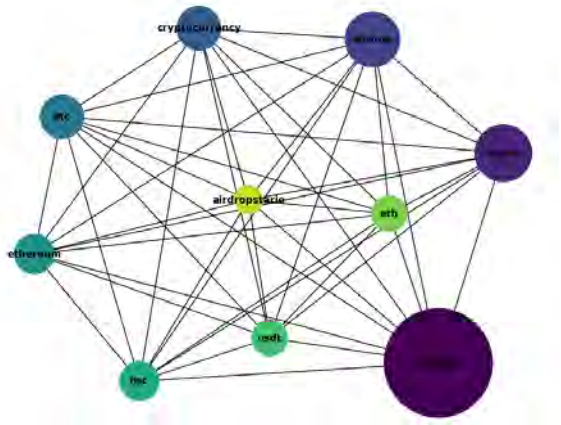

Figure 3.16: Hashtag co-occurrence of Top 10 Hashtags

Table 3.6: Co-occurrence Table

Index	Word 1	Word 2	Co-occurrences
0	bitcoin	crypto	66169
1	bitcoin	airdrop	61924
2	bitcoin	cryptocurrency	39475
3	bitcoin	btc	39073
4	bitcoin	ethereum	32299

Furthermore, an initial graph [3.15] has been constructed to showcase the hashtag co-occurrence where each node is a hashtag, edges reflect the co-occurrence of hashtags, and edge weights the count of co-occurrence to allow clustering and centralization of the prominent hashtags. Only the top 100 hashtags were shown to allow for clarity. One of the interesting properties that can be contrived from the graph is that there are no isolated nodes showcasing how all of the top 100 hashtags have been used in combination to another hashtag at some point. Additionally, we can also see the hashtags that have fewer edges and are used less within our data be pushed to the outskirts while more

prominent hashtags are in the center amidst many gray lines blending together showcasing a very high co-occurrence.
Furthermore, a simpler graph of a significantly smaller sub-sample consisting of only the top 10 hashtags and their co-occurrences was made for visual clarity shown in the figure [3.16]. In this graph, each node only has singular edges to the other nodes signifying co-occurrence and the node sizes indicate the occurrence of the hashtags themselves. The table [3.6] shows the numerical values of co-occurrences.

Finally, for predictions, we created a matrix to find the correlation between the features in the dataset to find out how strongly or weakly certain features are relative to one another to see if any features hold more weight in predicting another feature.

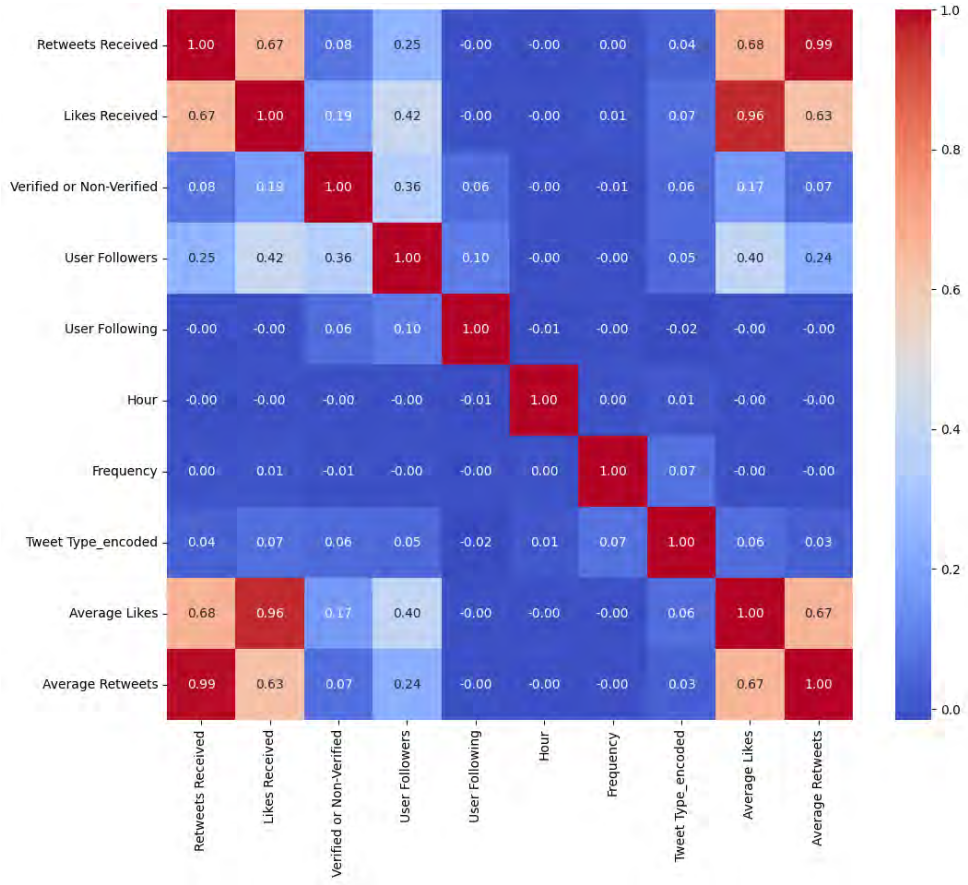

Figure 3.17: Correlation Matrix of Features

We can see from the correlation matrix [3.17] that frequency has a moderate positive correlation with Tweet type whether it is tweet retweet or reply. The co-occurrence between Feature Frequency and Likes Received is weak and positive (0.01). This indicates a minimal relationship between the two variables. Again, it has a negative correlation with verified or non-verified implying that there is a correlation between them which could mean verified users tweet less or more.

4. Methodology

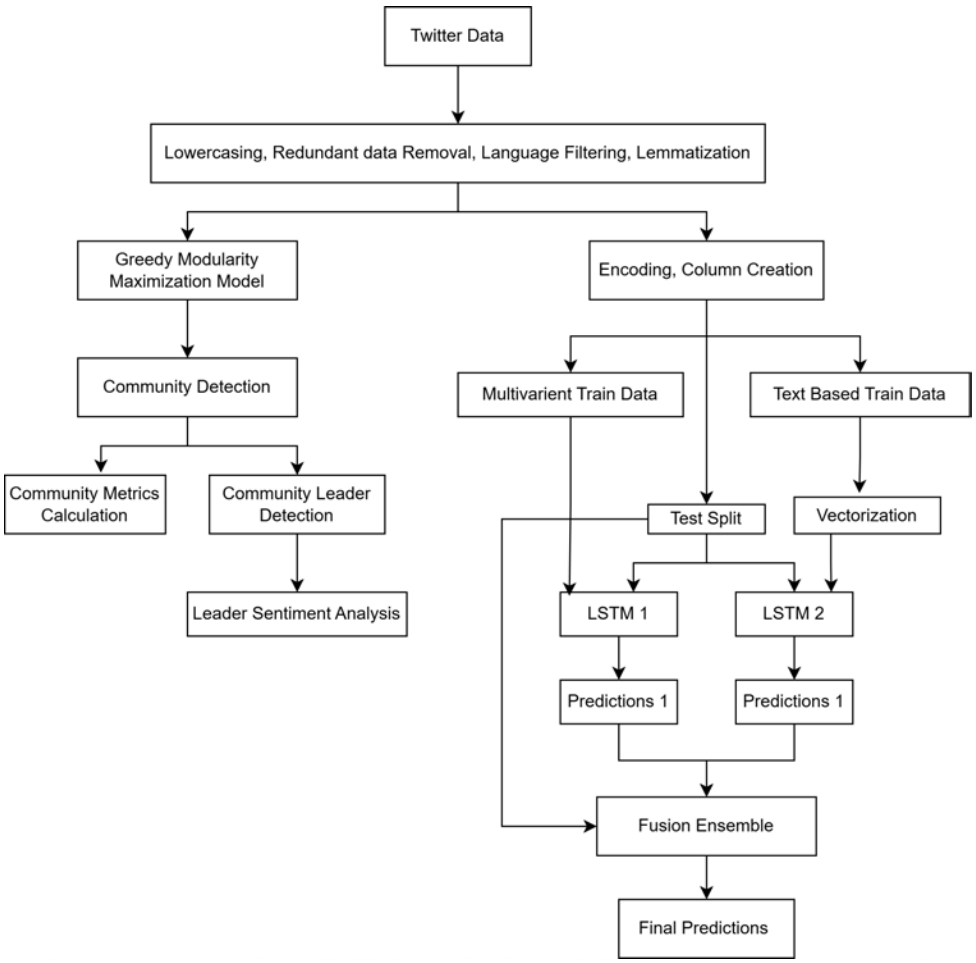

Figure 4.1: Flowchart of the models

4.1 Network Analysis through Graph Construction

Graph-based models for network analysis of the whole dataset were constructed based on connections between different entities. A user-user interaction graph was made on which algorithms were run to detect communities or distinct groups of users who engaged with each other through communication.
Graphs are data structures represented as a bunch of nodes and the connections between them are shown as edges. Each node in our graph represents a unique user of Twitter and the unidirectional edges represent a mutual relationship between two users established by a retweet of an original tweet. The Greedy modularity maximization implementation as well as other network-related implementations were used using the NetworkX libaries [26].

4.1.1 User-to-User Interaction Graph

We created user-to-user interaction graphs for our dataset where each node in the graph represents a unique user of X(Twitter) and the unidirectional edges represent a mutual relationship between two users established by a retweet of an original tweet that contains #bitcoin. We created this interaction graph using the library from Networkx in Python. We created different subsets and generated interaction graphs of those subsets to later detect communities from those graphs. The graphs can be found in the results and findings section.

4.1.2 Community Detection

Community detection from the constructed graph is a fundamental objective of this research. The research aims to detect distinct communities from the graph by identifying densely connected nodes that can form groups and distinguish between different groups. When nodes are closely connected and have more participation in certain areas, those highly connected behaviors can be measured using the greedy modularity maximization model. Using this modularity metric, different communities are detected based on their high modularity.

Greedy Maximum Modularity Model

There have been many algorithms proposed to detect communities between networks. One of the most effective algorithms is the greedy maximum modularity algorithm proposed by Clauset et al. [1] in 2004. The reason this algorithm was chosen for this research is because it runs faster than many other algorithms such as Newman and Girvan [1] in $O(md \log n)$ where d is the depth of the dendrogram. This is a clustering algorithm that iterates through the whole network while merging different combinations of different nodes to detect communities by maximizing the modularity parameter. Initially, all the nodes in the network are assigned to their own community. Hence, if there are n number of nodes, there will be a total of n number of communities. Afterward, for each pair of different nodes, it assumes those pairs to be communities. For each assumed community, it calculates the modularity gain by subtracting the previous modularity from the newly calculated modularity with the assumed community. If the modularity gain is positive, the merged community is identified as a community; otherwise, it is discarded, and a new combination is calculated again. After a community is detected, other nodes are added to the community and the modularity gain is calculated again. This way, the model iterates through the whole network and adds nodes to communities to which by adding the modularity is increased. These steps are repeated until there is no more modularity gain for any other combinations. Lastly, after the iteration of the whole network, different communities are identified and colored as those are distinct groups.

Fine-Tuned Greedy Modularity Maximization Model

Addressing the issue with the outliers, we tweaked the resolution parameter of our GMM model to give more attention to the small communities. Using the trial and error method, we tweaked the parameter several times to finally find the best homogeneous community-detected networks with proper attention to all the possible communities. Our final resolution parameter is set to 2 which favours the smaller communities over large ones

keeping the balance of over-splitting of communities at a high success rate. This resolution makes the GMM model zooms in on the small communities increasing its resolution from the standard and giving more attention as needed. Many outliers that were visible in the graphs were removed and the graphs were left with very few outliers.

4.1.3 Leader Sentiment Detection

After detecting communities using the GMM model, we wanted to find the leaders of each community to see what the leaders were influencing. In our graph, components were created based on retweets of the tweet a person created. In a community, the people who created the tweets are the leaders. We separated the nodes/ tweets with the highest degrees as the leaders since the degree indicated how many other nodes this particular node is connected to. After leader detection, we created a separate dataset for leaders and their tweets. The total number of leaders detected was 1503. Furthermore, we implemented several pre-trained sentiment analysis models to analyze the sentiments of the tweets of the leaders of communities. Finally, we used Sentiment Analysis on the SST-2 Binary classification dataset that was pre-trained on a transformer called ALBERT: A Lite BERT for Self-supervised Learning of Language Representations with an accuracy of 97.1% [21]. This accuracy is higher than several other sentiment analysis models that we used, making it a reliable tool for detecting sentiment in text.

4.2 LSTM: Predicting the Next Hour's Frequency

LSTM (Long Short-Term Memory) is a type of recurrent neural network (RNN) that can capture a long sequence of information and learn different patterns over a period of time which is especially good for sequential data tasks such as time series forecasting. Since our data comprises of two completely different types of inputs, one for the numerical features and one for the textual content, we used two different LSTM models to handle the two different types of input data and combined them.

4.2.1 Splitting the data

For each model, we now had to prepare two separate types of data. As such we made two initial copies of our data, one containing tweet vectors, and one containing the multivariate features, and scaled our training features separately. We, however, ensured that both models were predicting the same target variable with the same scaling. To ensure this, from the original data, we set aside a certain portion as test data, and from the remaining data, we made two copies and performed the necessary processing to prepare them for the two LSTM models, trained each model on their own training data, and then finally combined them in an ensemble to predict the test data we had set aside previously. This also ensured that the test data was not seen by either model during training. Any additional splits mentioned below were done on individual train data copies for evaluating the different approaches, or a split on the test data set to train the ensemble models. Lastly, all splitting was done based on users, not on the hours, thus maintaining the sequentiality of time so all models correctly look sequentially from Hour 0 to Hour 37 and finally predict the frequency of Hour 38, just for different users.

4.2.2 Multivariate LSTM Forecasting

We trained an LSTM model to predict the frequency of a user's last hour based on his previous hours' activities. This helps us understand how frequently a user is interested in engaging in conversation on a particular event such as Bitcoin in our case. Since we have a total of 39 hours of data from the 0th hour to the 38th hour, we have predicted the frequency of the last hour(38th hour) for each user based on their previous 38 hours(0 to 37).

Input Data

One of the most essential tasks for preparing an LSTM model is proper input data creation. Since, we wanted our model to learn the patterns of individual users and based on their individual behavior, wanted to predict their activity of the last hour, we uniquely identified the users and stored their data per hour. As mentioned in our dataset processing, to simplify the complication of multiple tweets in an hour which could result in a change in user features, we only kept the features after the last tweet of each hour. The data was then stored in a 3-dimensional array where the exterior array contained an array for each user, and each user array contained 39 arrays for each hour(0 to 38), where each hour array contained the features for that hour. For hours where there were 0 tweets, the hour array was padded up to the feature size. The 3D array was then split into the input and target(test). The input shape resulted in arrays in (users, hours, and features) where the first dimension is for all the unique users in our dataset, the second dimension is to hold all 39 hours of data for each of the users and the third dimension carries the features for each user in each hour. Furthermore, the test data is the frequencies of the last hour for the individual users only. Hence, its dimension is (users, 1, the frequency feature) which carries only the frequency of each user for the last hour. Afterward, the frequency for the last hour(our target variable) and the features were scaled separately using the standard scaler. Moreover, the data was split based on the users after preparing the dataset. Hence, 70% of the users were used for training and 10% for validation tests while training. Again, 20% of users were used for testing purposes.

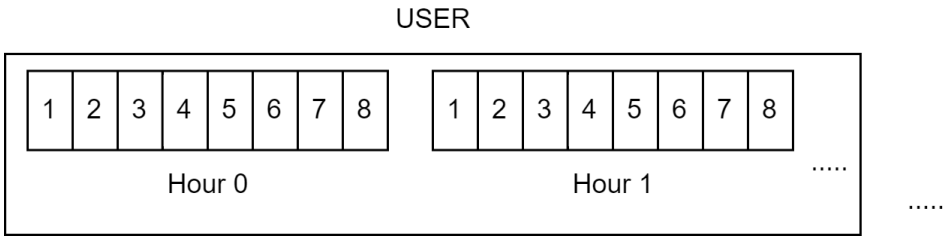

Figure 4.2: Multivariate Input Shape

Model

We prepared our model for multivariate time series forecasting of the frequency of users using several layers. Firstly, there are 38 LSTM units in the first layer of LSTM with full return sequence characteristics of the output since the input shape is of size 38 for hours 0 to 37 so each hour is processed in one LSTM unit. Additionally, our input vector shapes for each unit are of the feature size which is 6 or 8 in our case for different approaches. In recurrent models like LSTMs, it's common to treat layers as part of a funneling process,

where higher-dimensional input data is gradually compressed. Hence, we maintained the number of hidden state outputs to 16 as we didn't want to expand too much and risk overfitting nor did we want to lose our model's ability to handle the complexity. We decided to use 16-sized hidden layer output by trial and error of different hyperparameter tweaking. In this first layer, L2 Regularization is used with a parameter value of 0.01 adding a gentle penalty to make sure that weights don't become too large that our model overfits easily. This first layer returned 38 time steps and against each time step the 16 outputs of the hidden layers of LSTM created a (38, 16) output shape. In order to tackle the over-fitting problem, a dropout layer 0f 0.02 is added which dropped 20% neurons after the first layers. Afterward, another layer of LSTM is added with the same structure except this time only the last hour's output was of size 16 for the last hidden output of the last LSTM unit. Again, another dropout is added followed by a dense layer of output with unit 1 which is our final frequency prediction for the last hour.

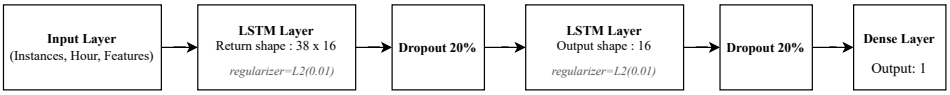

Figure 4.3: Abstract LSTM architecture for Multivariate model

The model is trained using the loss function MSE which tries to reduce the to minimize this loss during training, leading to better accuracy in predictions. Additionally, the optimization technique called Adam is used to adjust the learning rate dynamically to improve performance. Moreover, early stopping is implemented by monitoring the validation loss with a patience of 10 which ensures the model won't overfit while training since it'll observe the validation loss for 10 epochs, and if it doesn't improve, then training will stop early.

Forecasting Approaches

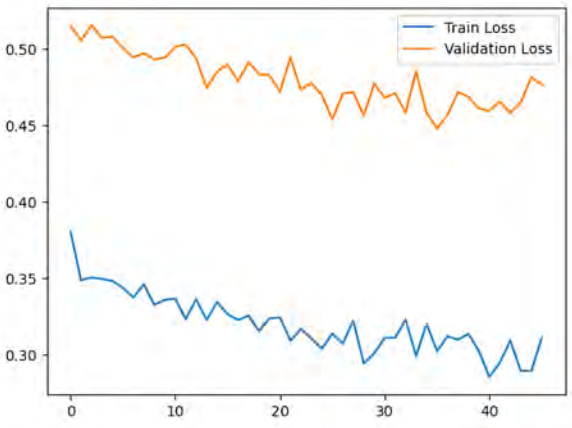

Figure 4.4: Train vs validation loss for non-imputed 6 Features forecasting

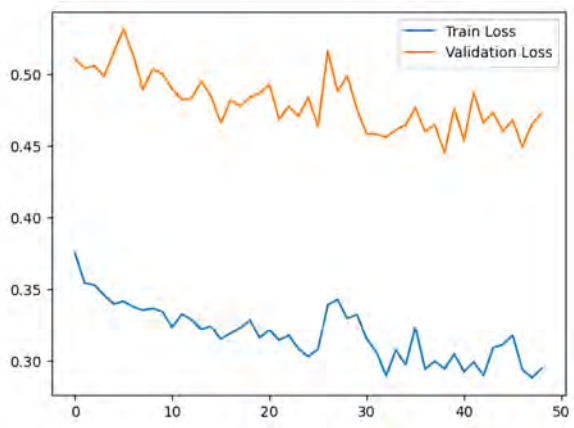

Figure 4.5: Train vs validation loss for imputed 6 Features forecasting

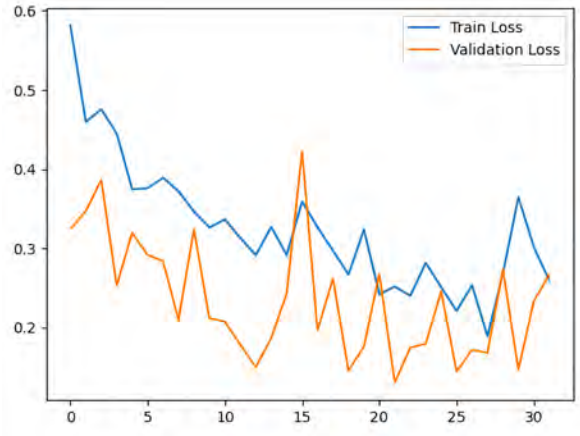

Figure 4.6: Train vs validation loss for imputed 8 Features forecasting

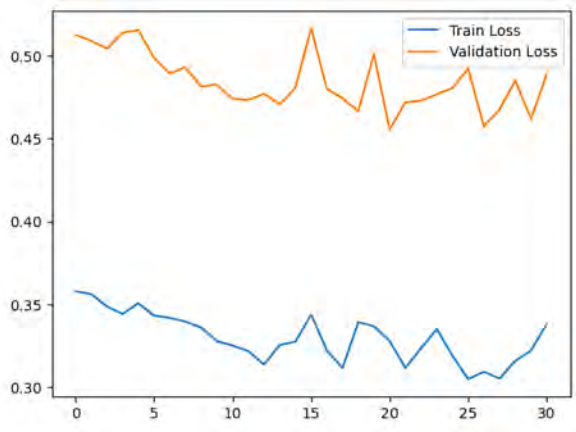

Figure 4.7: Train vs validation loss for non-imputed 8 Features forecasting

6 Feature non-imputation method

Firstly, we prepared the input with 6 features excluding the average Likes and average Retweets. The features included: 'Sum of Retweets Received', 'Sum of Likes Received', 'Verified or Non-Verified', 'User Followers', 'User Following', and 'Frequency'. However, The model was not performing to our expectations. In figure [4.4] it can be seen that the validation loss is higher than the training loss, meaning our model is over-fitting too much to our training data and can not perform well with unseen data. After some contemplation, we concluded that these results arise due to the nature of our data itself, as we can see in figure [3.2], most of the frequencies of our data lie in the smaller range and very little of the data has higher frequencies, which is actually very natural as most people are unlikely to post multiple times in an hour. As our validation set is very small (10 percent) compared to our training set, it is likely that very more instances with high frequencies end up in our validation data, so it might become difficult for our model to predict the validation data. Additionally, since our model was struggling to learn the pattern of our data, it needed over 45 epochs which is a lot for this data. Hence, since the validation data did not have enough pattern compared to the train data which is why it resulted in under-fitting.

6 Feature imputation method

This led us to impute the dataset to make it continuous to increase our model performance and add complexity to our data. We imputed the data by implementing a simple mathematical calculation. Firstly, we iterated over all 39 hours of data for each user, and for each hour where there was no tweet indicating a padded vector of 0s, indicating that the user did not make any tweet that hour, we calculated the average of its previous and next hour and assigned it with that value. As we are iterating from the start, if the starting values are all 0s, they shall not be imputed, so to ensure that the starting values are also imputed in the case of 0s, we also ran a back-iteration and did the same calculation. This way, a larger value was reduced to very little of its weight, and our data continued keeping the important information about the user's activity period intact.
It can be seen from the figure [4.5], that after imputing the data, our model is learning slightly better than before with validation loss lower than the previous approach. However, it was still overfitting and couldn't learn the patterns properly as it needed the same number of epochs. Additionally, the loss reduction was not significant as well.

8 Feature imputation method

In order to get better performance, we increased the features by adding 2 more features which are the average likes and average retweets for each user per hour. This added a boost to our model in terms of understanding patterns and it performed better than the previous models during training since it needs fewer epochs under the same configuration of early stopping indicating that the model is learning patterns effectively without memorizing. It can be seen in Figure [4.6] that our validation loss and train loss are very close now and the model performs better in understanding the patterns of the data as it performs well predicting the unseen data. However, the validation loss is now lower than the training loss for the most part indicating under-fitting, but increased data complexity results in the model needing fewer epochs to learn the variations after imputation and feature addition. More importantly, the imputation is manipulating our original data to get a lower error which is theoretically wrong. We concluded that real-time data won't be imputed which can result in our model predicting wrong frequencies in fractions. Hence, we decided to remove the imputation technique to stay more true to a real-world scenario.

8 Feature non-imputation method

After removing the imputation and keeping 8 features we prepared our final multivariate time series frequency prediction model. This model not only solves our theoretical doubts but also has a desired validation loss and train loss. It can be seen [4.7] that the validation loss is greater than the training loss which was our expected result suggesting that the model is not under-fitting to the training data, which is a critical factor when ensuring long-term stability and performance. Additionally, the difference between validation and train loss is lower in this model. Like an ideal case, both train and validation loss decreased over time with validation loss higher than train loss. This means that our dataset is properly set and the model is learning from the dataset finding patterns and reducing errors as it iterates through the training process. More importantly, it needs a lot less number of epochs indicating that the model is capable of learning the patterns quickly and efficiently. The validation loss being higher than the training loss, yet decreasing suggests that the model is still capable of generalizing well to unseen data.

4.2.3 LSTM Forecasting using Tweet Content

We trained another LSTM model yet again to predict the frequency of the last hour(38th hour) of a user based on his previous hours(0 to 37), but this time training the model on the textual data of each user. We tried to find if only text content in vector form had any effect on the frequency of tweets and if it was a valid means of predicting the frequency of data.

Input Data

Similar to the multivariate model, the initial goal was to get our input to the format of (Users, Hours, Vector dimensions as features) where the first dimension is for all the unique users in our dataset, the second dimension is to hold all 39 hours of data for each of the users and the third dimension this time carries the features for each user and each of the hours. This time, however, the main complication arises from the possibility of there being multiple tweets in each hour, all of which must be kept to train the model on the entirety of the textual content. We decided on four main methods to tackle this

problem, one was to concatenate the raw tweet content for each hour and then vectorize the data. The other methods all performed vectorization first and then branched out to either perform element-wise addition, mean pooling, or max pooling. For vectorization, we used BERT while setting the maximum input sequence length of 512 into BERT and producing the maximum output vector of size 768. The steps following this were the same as the multivariate model, the data was grouped for each user for each hour and stored in a 3-dimensional array where the exterior array contained an array for each user, and each user array contained 39 arrays for each hour(0 to 38), where each hour array contains the BERT vector for the tweet content of that hour. Yet again, for hours where there were 0 tweets, the specific hour's array was padded up to the max BERT vector dimension size. The 3D array was then split into the input and target(test). The input shape resulted in arrays in (users, hours, and features) where the first dimension is for all the unique users in our dataset, the second dimension is to hold all 39 hours of data for each of the users and the third dimension carries the BERT tweet vectors for each user in each hour. Similarly, the test data is the frequencies of the last hour for the individual users only. Afterward, the frequency for the last hour(our target variable) and the BERT vectors were scaled separately using the standard scaler. Moreover, the data was split based on the users after preparing the dataset. Hence, 70% of the users were used for training and 10% for validation tests while training. Again, 20% users were used for testing purposes.

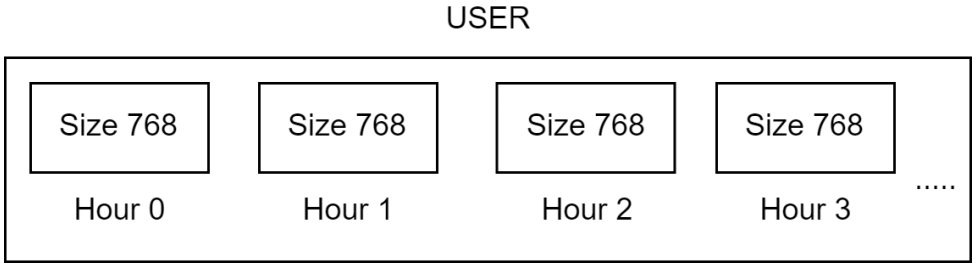

Figure 4.8: Tweet Content Input Shape

Model

The model for text content time series forecasting of the frequency of users uses several layers. The first layer of the LSTM takes in an input of size (instances, 38, 768), therefore there are 38-time steps, meaning we are training the model on the first 38 hours(0 to 37) or considering the previous 38-time steps. Since our input vectors are 768 dimension vectors for each hour, to perform funneling and squeeze or compress the data down, we have selected 128 as the size of hidden layer outputs for each unit in the first layer which will compress the vectors down to 128 size vectors. From this layer, each unit of the LSTM returns an output to the next layer with size (38 x 128). The layer after this is a dropout layer that drops 30 percent of the data to prevent overfitting. Following the funneling convention, the units in this layer were set to 64 to further compress the data to hold information. Following this again, the next layer is a dropout layer to drop 30 percent of the information. The next layer is a dense layer of size 32 to funnel the data even more followed by one last dense layer of output shape 1 to get our prediction.

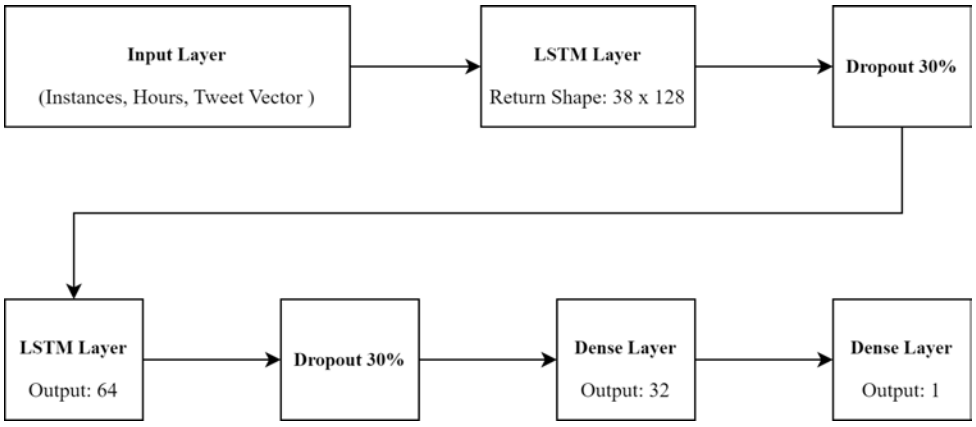

Figure 4.9: Abstract LSTM architecture for Tweet Content model

This model is also trained using the loss function MSE which tries to reduce the to minimize this loss during training, leading to better accuracy in predictions. Here also, Adam is used to adjusting the learning rate dynamically to improve performance before eventually settling on a learning rate of 0.0001, which is lower than the standard but helps the model fit to the data more gradually, leading to more stable results. Early stopping is also implemented by monitoring the validation loss, but this time with a patience of 20.

Text-based Forecasting Approaches

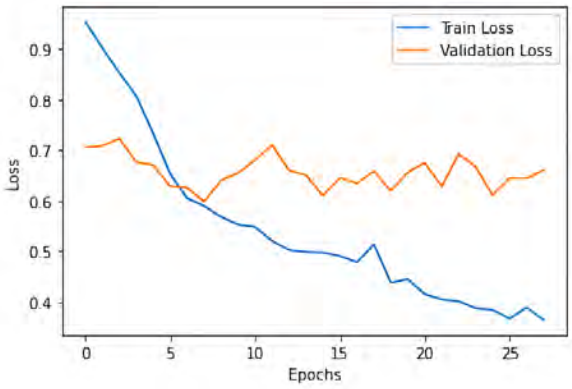

Figure 4.10: Training and Validation Loss for Mean Pooling

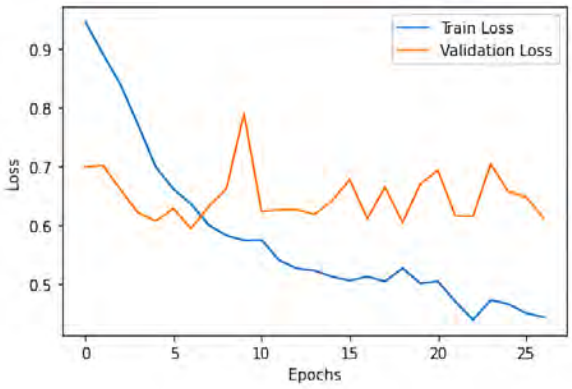

Figure 4.11: Training and Validation Loss for Max Pooling

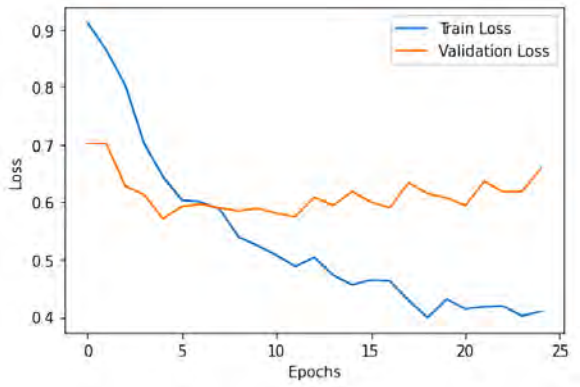

Figure 4.12: Training and Validation Loss for Concatenation

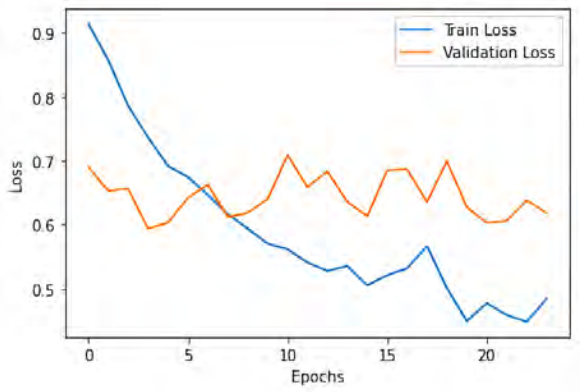

Figure 4.13: Training and Validation Loss for Addition

Tweet Concatenation

The first method we attempted was concatenating the raw tweet content for each hour before vectorization. It took 25 epochs for our model to be trained as seen in the graph [4.12]. However, we realized that some information may be lost as the maximum input length in Bert is 512, and when we concatenate a large number of tweets, the word count exceeds this number causing some information to be lost. Hence, we moved on to trying different approaches.

Mean Pooling, Max Pooling, and Element Wise Addition

The next methods we attempted were done to circumvent the maximum input length for BERT vectorization, meaning we vectorized the tweets first and then performed some sort of calculation on them to aggregate the tweets in an hour. The first method we tried was mean pooling which performs element-wise averaging for each vector [4.10]. We then tried max pooling which keeps the maximum element for each index in all vectors for that hour [4.11]. The final method we attempted was aggregating the vectors through element-wise addition [4.13]. As can be seen from the graphs, they all boast very similar results, all requiring around 25 epochs, ending with a training loss of around 0.4 and a validation loss of around 0.65. As such further analysis using metrics was required to evaluate which model would truly give the best results.

4.2.4 Fusion Ensemble Model

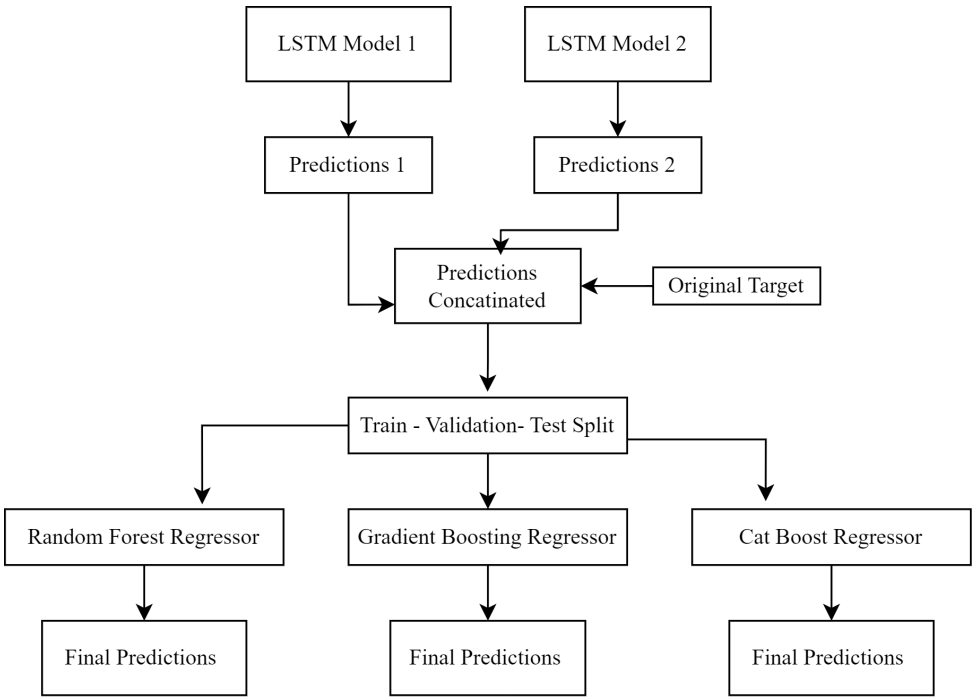

Figure 4.14: Workflow of the fusion ensemble models

After getting predictions from our multivariate frequency prediction model and frequency prediction model using tweet content, we used regression models to ensemble those outputs and generate the combined results of these two models. Firstly, we created a 70% test split, 10% validation, and 20% test split of the two input features with the target variable which is the original frequency from data. Afterward, we implemented several models to train on the prediction results. Firstly, we used the Random Forest Regressor, Gradient Boosting Regressor, and finally the Cat Boost Regressor to generate the final output.

5. Results and Findings

5.1 Graph Network Analysis

In this section, we analyze the graph network constructed from user interactions, focusing on retweet connections within the dataset. Our goal is to explore the relationships between users and identify potential community structures through network metrics and community detection techniques. This analysis provides insights into the overall network dynamics and user behavior.

5.1.1 User-to-User Interaction Graph

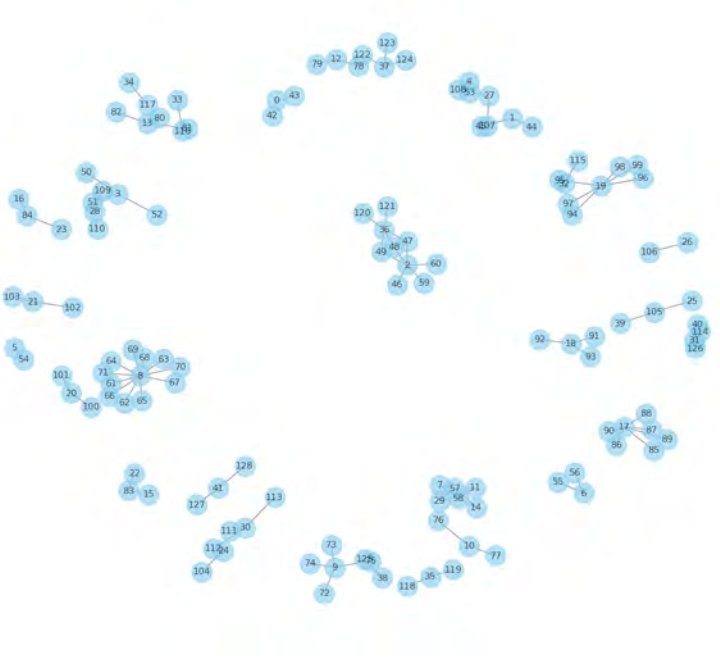

Figure 5.1: User Interaction Network(Subset 1)

In the figure [5.1], a graph is constructed based on only 0.005% of the dataset that represents the connections between different users of tweets. The nodes in this graph are labeled with numbers in the serial these were added to the graph. The edges represent the connection that represents that a user has retweeted an original tweet containing #bitcoin. The retweets are determined by checking for mentions of the user who posted the tweet in the tweet content of each tweet. When a retweet is found, an edge is established between the original tweet user node and the retweet user node. For instance, from node 83, there are two edges towards the 22nd and 15th user nodes. This identifies that user 83 has an original tweet that is retweeted by both users 15 and 22. Similarly, all the other nodes that are connected to a distinct node represent users who have retweeted a tweet by the

user they are all connected to.

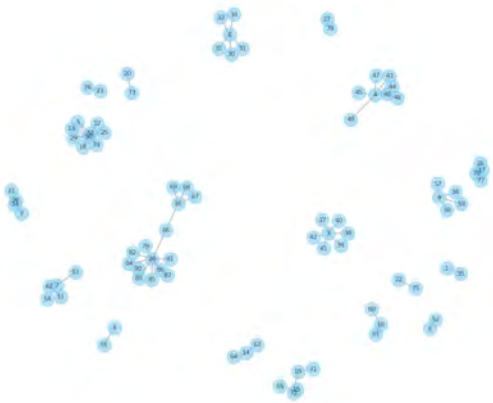

Figure 5.2: User Interaction Network (Subset 2)

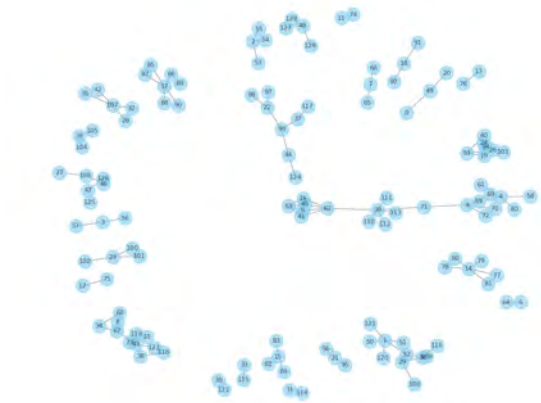

Figure 5.3: User Interaction Network (Subset 3)

Further tests were run on random sub-samples to ensure homogeneity. As it can be seen from each figure there are different nodes connected to a central user node which are more likely to be community leaders after community detection. Each figure shows separate clusters representing clusters of connected graphs completing this whole disconnected network. Here, different components can be later used to further identify communities within these separate components.

5.1.2 Community Detection

After creating User-user interaction graphs, we implemented the greedy modularity maximization method and optimized it to its best ability to detect communities in our network.

Greedy Modularity Maximization

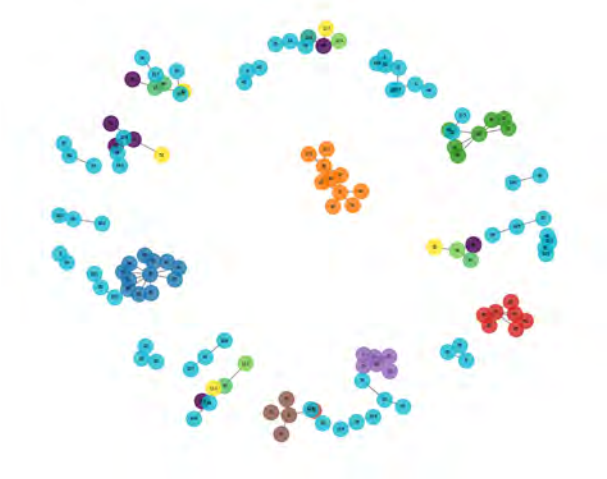

Figure 5.4: User Interaction Network(Subset 1) with community detection

The figure [5.1] showcases a graph with nodes and edges that represent the interactions between users of Twitter based on retweets of a small subset of our dataset. However, just from this much it is hard to visually pinpoint groups or communities. Hence, in figure [5.4] Newman's maximum modularity detection approach is used to detect communities within that graph constructed using 0.005% of the whole dataset. The nodes are labeled with the numbers used in the previous graph. However, the nodes with no edges or zero degrees are removed as those can't form any community. Additionally, after the community detection, the network is divided into distinct communities, each represented by a different color. For instance, the nodes numbered 7, 29, 58, 57, 11 and 14 are colored purple and are represented in a different cluster showing that these nodes create a community. Similarly, many other nodes have created clusters with different colors each representing different communities.

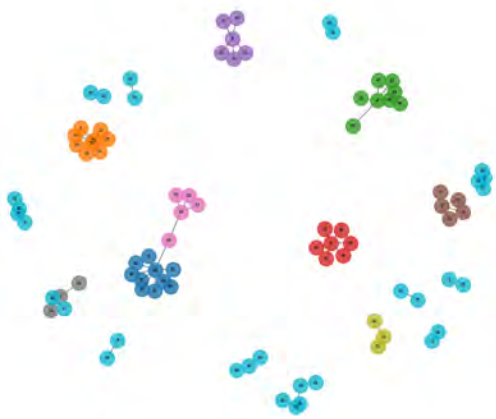

Figure 5.5: User Interaction Network (Subset 2) with community detection

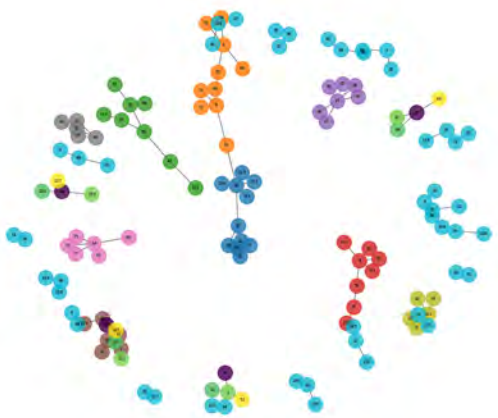

Figure 5.6: User Interaction Network (Subset 3) with community detection

As seen in figure [5.6], the blue and orange coloured groups are detected as different clusters due to their merge modularity increase performance even though these are connected by an edge. This can also be seen in figure [5.5] with the blue and pink cluster which portrays that different combinations of nodes connected by edges are taken into account while calculating the modularity gain and are declared as community by the performance of those clusters merges.
Greedy maximum modularity community detection is an optimal state-of-the-art model to iteratively detect distinct communities in a large network. Even though it is a simple method to implement it works as a very powerful tool to concise a very large amount of data by dividing it into different components. However, it may not always find the optimal result as it may get stuck on local optima and no more communities can be merged in those cases as there will not be a modularity increase. This may lead to sub-optimal communities as the algorithm is greedy and doesn't look out for global optimal solutions. Therefore, the blue-colored clusters are present in all three figures which occurred when the algorithm reached local optima and the modularity gain stopped increasing even after taking the existing nodes in combination with the other nodes or each other. Hence, these nodes are presented as outliers or those that don't belong to any community. When set of nodes are merged into different clusters, but the modularity gain is negative those nodes are not added to a community. If the exploration comes to a point where for no combination of merging nodes the modularity increases, the greedy maximum modularity

model stops and colors the outliers are blue that represents that those are not of any community due to weak connections. Since these graphs run on different subsets of our dataset and show the same characteristics, it can be declared that these are homogeneous networks and will behave similarly in the whole dataset.

Fine Tuned Greedy Modularity Maximization

Figure 5.7: User Interaction network (1)

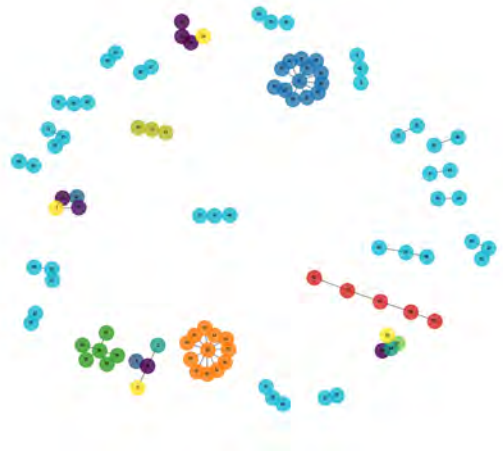

Figure 5.8: Community detection (1)

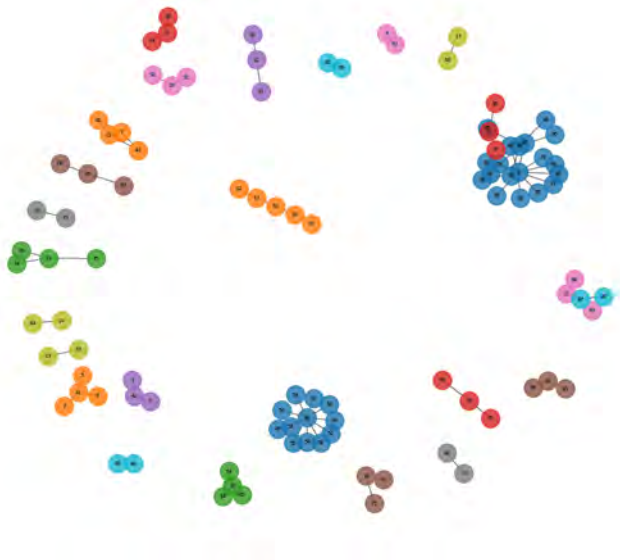

Figure 5.9: Fine-tuned community detection (1)

In figure [5.7], a user-to-user interaction graph is created that is evaluated using GMM with standard resolution in figure [5.8] which fails to properly identify the communities for several nodes such as nodes numbered 3,42,6 that are connected as a component. The numbering in the nodes represents the order in which these were inserted into the graph which is completely for the identification process. Now, after applying the resolution at 2, in Figure [5.9] it can be seen that the outliers 3, 6 and 42 have now been assigned the color

"purple" as it has been identified as a different community. Similarly, many other outliers colored sky blue have now been assigned to a community after increasing the resolution.

Figure 5.10: User Interaction network (2)

Figure 5.11: Community detection (2)

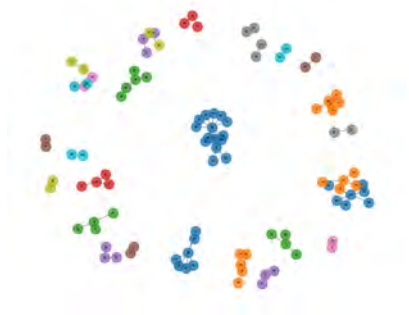

Figure 5.12: Fine-tuned community detection (2)

Figure 5.13: User Interaction network (3)

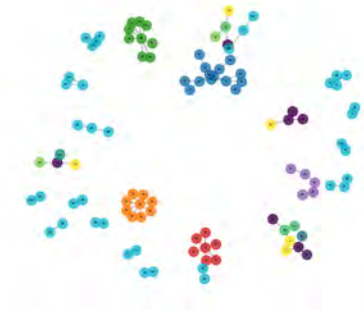

Figure 5.14: Community detection (3)

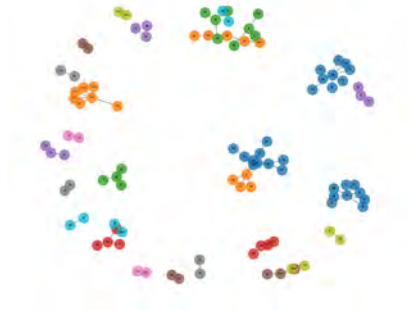

Figure 5.15: Fine-tuned community detection (3)

Figure 5.16: User Interaction network (4)

Figure 5.17: Community detection (4)

Figure 5.18: Fine-tuned community detection (4)

Figure 5.19: User Interaction network (5)

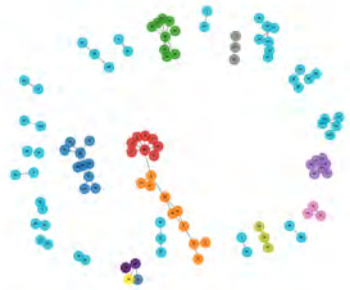

Figure 5.20: Community detection (5)

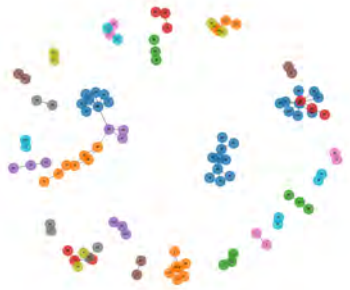

Figure 5.21: Fine-tuned community detection (5)

The successful community detection for the whole data cannot be interpreted from the enormous network that will be generated. Hence, several subsets have been created to implement GMM to prove the homogeneity of the graph that these all behave the same way and so will the graph generated for the whole network. In the figures above, it can be seen that the graphs with resolution 2 are behaving similarly in the task of community detection establishing that hyperparameter tweaking has created a better model implementation for our social media network.

5.1.3 Leader Sentiment Detection Results

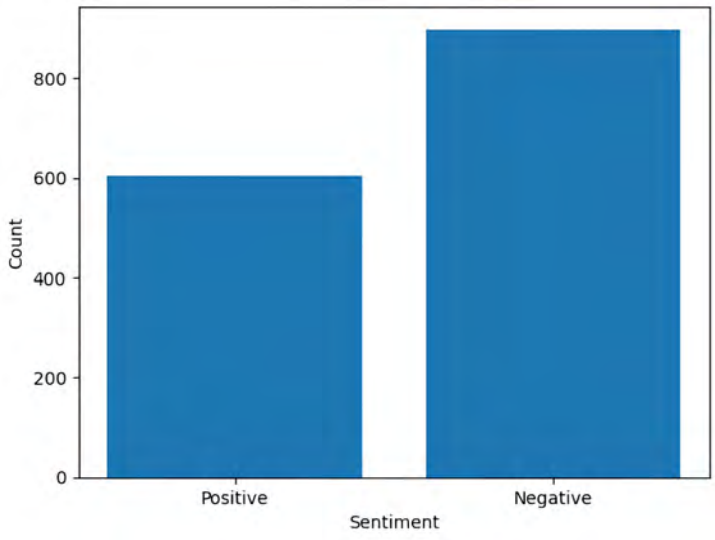

Figure 5.22: Albert results on sentiment analysis of Community Leaders

After implementing the model Albert trained on the SST-2 dataset, as shown in figure [5.22] out of the 1503 leaders 600 of them sent positive tweets whereas the rest of the 900 leaders spread negative sentiment, hence, this result sums up that positive sentiments were less used, and negative sentiments held a rather higher reach in terms of spreading hashtags in social networks. This helped us understand the kind of news these leaders were spreading whether these are positive or negative. The motive behind this analysis was to understand what sentiment increases the reach and activity of users on a certain event. For instance, since the leaders mostly had negative sentiments, this can mean that negative tweets

are more likely to be followed by users and they are more likely to engage in those. Understanding the sentiment expressed in the leaders' posts, we could gain insights into what tone drives user engagement and the spread of a hashtag. This analysis helped us to identify patterns that can be used for future campaign promotion strategies. Further research could explore the underlying psychological and social factors driving this result, which could help in more effective social media marketing and content strategies.

5.1.4 Component & Community Metrics Analysis

To conclude the graphs, several metrics are calculated to prove different hypotheses of the graphs and analyze the networks after community detection. Inspired by Albert Barabasi [9], our measurements include mean centrality, average degree, average path length, clustering coefficient, and degree centrality for each of the communities that were detected after implementing the greedy modularity maximization algorithm by Newman. These properties provide an in-depth analysis of the behavior of the graph and prove different characteristics of the users in the network through communities and components.

Table 5.1: Top 5 biggest Components

Component	**Size**	**Edges**	**Density**	**Avg. Path Length**	**Avg. Degree**	**Mean Clustering Coefficient**
1	11414	23895	0.000367	4.481839	4.186963	0.018676
2	9467	11213	0.000250	3.901057	2.368860	0.001891
3	7586	14969	0.000520	3.009789	3.946480	0.000440
4	7222	11430	0.000438	4.025407	3.165328	0.000030
5	5259	5428	0.000393	2.659996	2.064271	0.000010

Table 5.2: Top 5 Biggest Communities

Community	**Size**	**Edges**	**Density**	**Average Path Length**	**Average Degree**	**Mean Clustering Coefficient**
1	6715	12990	0.000576	2.608315	3.868950	0.000248
2	5510	10752	0.000708	4.083798	3.902722	0.002332
3	4014	4030	0.000500	2.073640	2.007972	0.001914
4	3158	6441	0.001292	5.081623	4.079164	0.040676
5	2557	2556	0.000782	2.009362	1.999218	0.000000

Nodes and Edges

Graphs are constructed using nodes and edges. The more nodes and edges the more dense the graph is. Using the complete dataset to formulate a complete graph and analyzing the node count and edge count for both components and the communities detected, it

can be seen from [5.1] that the biggest component of the network has 11,414 nodes and 23,894 edges connecting those nodes. Additionally, the top 5 biggest communities of the network have thousands of user nodes who have created communities involving a high number of connected edges. When the modularity maximization algorithm detects the communities, it merges sets of nodes based on the density of edges between those nodes which separates a certain number of nodes from others. Hence, the number of edges plays a significant role in community detection. From the figure [5.2]it can seen that for the biggest component, the number of edges is almost double the number of nodes. However, as the number of nodes decreases, the number of edges reduces its density and is decreased to a number that is closer to the number of nodes. Similarly, for 5259 nodes the number of edges is 5428 in the fifth biggest community [5.1]. This metric explains the behavior of the graph by explaining the clustering coefficient or interconnectedness of each community and component. Hence, the larger the community or component, the more likely it is that the neighbor nodes have edges between them which is why there are more edges than the number of nodes. The smaller the communities or components the fewer chances of neighbors having edges between them which is why the edge number keeps decreasing toward the node count. At one point, when the edge count is almost equal to the node count which is the case for most of the dataset, the community will have no neighbors with edges connecting them, rather all neighbors are connected to one central node.

Mean Clustering Coefficient

The analysis showed that most of the connected components have a mean clustering coefficient of 0. This displays a very interesting characteristic hypothesis showing that inside social networks of X(Twitter), it is extremely unlikely that users inside the same social circle retweet a tweet by the group of users who retweeted the original tweet, rather all retweets are essentially done from the original tweet. All 3 of the smaller subsample User interaction figures, [5.1], [5.2] and [5.3], showcase this behavior as it can be seen that the neighbors have no edges between themselves in their components, and the results from the complete dataset display this same property, further pushing our hypothesis. However, interestingly, it can be seen that the biggest components of our dataset have a very low clustering coefficient. For instance, as seen in Table [5.1], the mean clustering coefficient for the community with 11414 nodes which is the biggest component in the dataset, is 0.01867. This means that, on average, only about 1.867% of the connections between the neighbors of a node in this component are actually present which is relatively low. The other components in the table show similar results. As explained earlier, node edge counts appear more than the node count which represents the extra edges that connect the neighbor nodes leading to a clustering coefficient value other than zero.
Similarly, it can be seen from the table [5.2] that after community detection the big communities have a clustering coefficient close to 0 representing that the interconnectedness between the users who have retweeted a tweet in a community is very low and highly unlikely which justifies the behavior of the user interaction graphs with community detection. Moreover, the mean clustering coefficient for the mid and small-sized communities was seen to be 0 as well. Lastly, this proves that hashtags are not mostly propagated or shared among the users who are already engaged, rather these users who are engaged in the hashtags in a community influence the spread of the hashtags further to new users but not within themselves in most cases except in big communities where we say some exceptions.

Average Degree

The average degree provides a further measurement of the connectivity of the network. In Table [5.1], the average degree for different components within our network is presented. In the context of this analysis, the average degree can be interpreted to understand the overall interaction level within the network. A high average degree might suggest that users (nodes) have many social connections (edges), which could imply a high level of social engagement or interaction and vice versa. Therefore, a higher average degree generally indicates faster dissemination of information across the network. Components with higher average degrees are more likely to be more cohesive and connected as well as more dense whereas a lower average degree indicates a looser connection and higher sparsity in the community. The average degree does seem to showcase a somewhat proportional relationship with the size and more prominently the total links or edges of the network. It can be seen that the 2nd component, even though is larger in size, has a lower average degree than the 3rd and 4th components, however, the results remain consistent in proportionality when looking at the number of edges of the networks as component 2 despite being larger, has fewer edges than the 3rd and 4th communities. On the contrary, in the case of communities, It can be observed from [5.2] that although other big communities remain consistent with average degree and edge count, the 4th biggest community has a higher average degree than communities with higher edge count. This leads to a hypothesis that this community has a more clustered structure, where a few nodes have many connections, leading to a higher average degree whereas other communities have a more evenly distributed edge structure of the nodes. Other than this, the other communities show similar patterns when compared to the components. The mid-sized communities and small ones also have an average degree close to the big communities. For instance, a community with size 2 and edge count 2 has an average degree of 1.33. This can be explained by the fact that the average degree depends on the community and its connection structure. Since the edge count and node count are closer to each other this resulted in an average degree that suggests that within these nodes, a decent number of connections is present which when compared to the nodes isn't too high. Hence, these findings indicate that the average degree is a key indicator of connectivity within communities, revealing that even small networks can exhibit significant interaction patterns. This suggests that the structure of connections plays a crucial role in understanding social dynamics and information dissemination within the network.

Average Path Length

It is essential to study the average path length to understand the information propagation speed of a network. A low average path length indicates that nodes in the network are more efficiently and closely connected, with relatively short paths between them, which suggests faster information propagation in the network. On the other hand, a high average path length signifies that the network may be more decentralized or fragmented, resulting in slower communication and the spread of newer hashtags. From the table [5.1], it can be seen that the larger the size of the components, the higher the average path length aside from the sole exception of component 4. This suggests that more nodes result in a greater number of nodes being more disconnected and towards the outskirts of the network, resulting in a higher average path required to traverse the network. It is also notable that despite the greater increase in edges or links which should result in more efficient traversal, it seems edges are less prominent in determining the average path compared to the increase

in the number of nodes. Further insight can be obtained by looking into the average degree as a greater degree, meaning more connections to and from all nodes in the network should result in lower average paths, but yet again this hypothesis is not provable with the data present due to contradictory data as the largest component- component 1 with the highest average degree also has the highest average path length and the smallest component-component 5 with the lowest average degree also showcases the lowest average path length. From this, we can hypothesize that larger networks, despite the greater number of nodes and edges as well as greater average degrees, are more likely to have a greater number of outskirt nodes with minimal connections that bring up the overall average path length, and our table showcases this hypothesis with the sole exception of component 4. On the contrary, our initial hypothesis can be challenged by the results obtained from the table [5.2] where the biggest community has a comparably small average path length despite having the largest number of nodes and edges. Again, this community supports our second hypothesis that the average degree reduces average path length. However, the second and 4th communities show opposite characteristics. hence, this suggests that the structure of the network and the placement among nodes and their interconnectivity are important for understanding the connectedness and information propagation.

Density

Density is calculated as the ratio of the number of existing edges to the maximum possible edges. From the table [5.1], it can be seen that the density of the biggest component is 0.000367 which suggests that for every potential edge in the network, only about 0.0367% of those edges are actually present. This low-density value implies that while there are many nodes (11,414) and edges (23,895), the nodes are not very interconnected. Again, the other components show similar characteristics. Furthermore, if we observe the big communities, from the table [5.2] we can see all relatively low densities suggesting that the nodes are not as dense as it would've been possible suggesting sparse network connections. However, we can understand that due to our way of creating the graph, it is much less expected that the majority of nodes will be interconnected among them very densely since it is very rare in practical scenarios that all the users be densely connected through certain leaders. However, small and mid-sized communities show significantly higher density. For instance, a community with node count 3 and edge count 2 has 67% density due to the way of calculation and fewer possible connections yet the bigger communities uphold the real characteristics of our network in terms of density.

Centrality

Table 5.3: Community Top Users and Degree Centrality

Component	Top User	Degree Centrality
1	puritylekutit	0.044423
1	von_programmer	0.039779
1	kaka_aero	0.039078
...	...	...
2	clean28468721	0.203676
2	you97953679	0.182654
2	hello23643181	0.175681
...	...	...

The table [5.3] showcases the top 3 most prominent or rather the most "central" nodes or users within their respective components for 2 components. This is done by calculating the degree centrality of all nodes in the components and arranging them in descending order. The degree centrality is simply the degree of a node, except it is normalized to make nodal connections more comparable and easier to analyze. Relating to a graph, the most prominent users in their respective components have the highest degree of centrality or are the most central nodes connected to a majority of other nodes or rather they have the highest number of connections with other users and are creating the most engagement and interaction. These results highlight the critical role that degree centrality plays in identifying key influencers within the components. Users with higher centrality not only possess more connections but also significantly contribute to the flow of information and engagement across the network. This underscores the importance of focusing on these central nodes for understanding social dynamics and enhancing communication strategies within these communities.

5.2 Results of Activity Prediction

5.2.1 Multivariate LSTM Forecasting

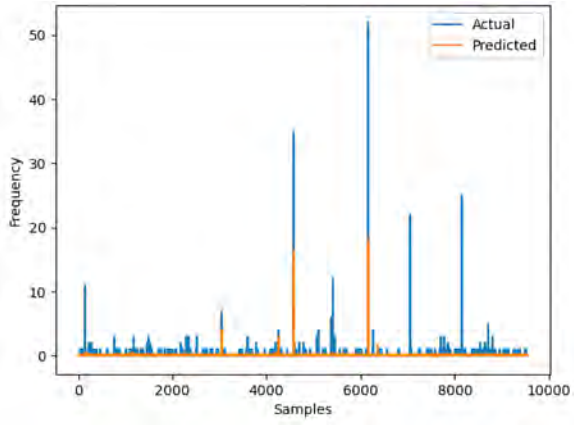

Figure 5.23: 6 features non-imputation predictions

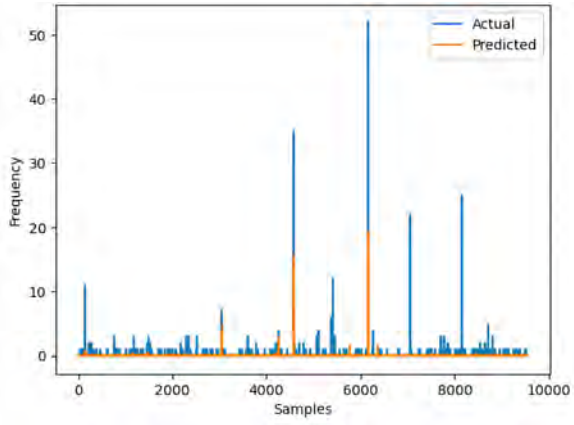

Figure 5.24: 6 features imputation predictions

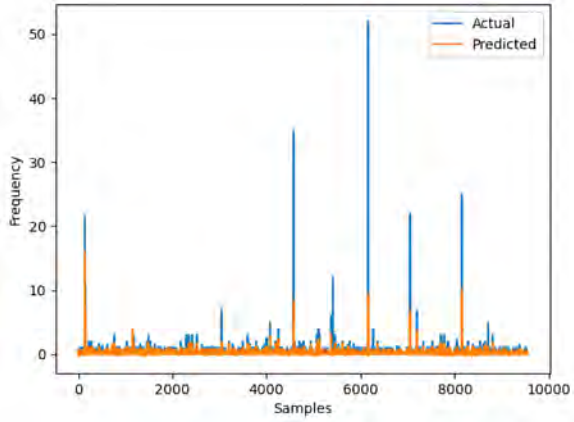

Figure 5.25: 8 features imputation predictions

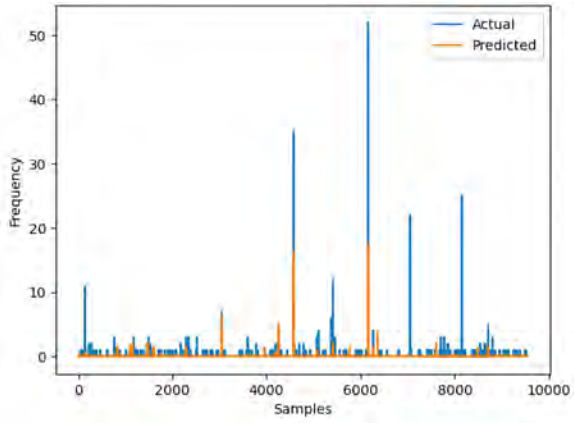

Figure 5.26: 8 features non-imputation predictions

Approach	MSE	RMSE	MAE	R-squared
6 Feature Non-Imputed	0.3453	0.5876	0.1098	0.4296
6 Feature Imputed	0.3383	0.5816	0.0717	0.4412
8 Feature Imputed	0.3554	0.5961	0.0639	0.4879
8 Feature Non-Imputed	0.3893	0.6239	0.0887	0.3570

Table 5.4: Model Performance Metrics Comparison

Comparison of the models

The graphs for plotting the predictions are tested on the same subset of users to make sure our comparison is fair for all approaches. As can be seen from the table [5.4] the imputed models have the overall best performance in terms of lowering the error metrics. These

models have increased R-squared values allowing the model to explain the variance in data since imputation added extra variance to be learned by the model. However, Imputation can artificially improve model performance but may distort the original data patterns. By choosing not to impute missing values, we are opting for a realistic data processing model that can handle real-world scenarios where missing values cannot always be filled accurately.
The table [5.4] shows that the 8-feature non-imputed data processed model has better performance with lower MAE than the 6-feature non-imputed model. Our hypothesis is that due to the added extra 2 features our model is able to learn the data patterns and find the correlations better. This can also be observed from the train and validation loss graphs from the previous section where the 8-feature model is being able to learn more quickly with fewer epochs compared to the 6-feature model. Hence, we chose to finalize the 8-feature non-imputed model observing several reasons. Firstly, the 8-Feature Non-Imputed model includes additional features that could capture more patterns of user behavior which allows the model to handle a wider range of inputs with the added complexity. Over time, with more data and fine-tuning, this could lead to improved generalization in real-world applications. Secondly, the MAE of this model is much better suggesting that the overall magnitude of the actual errors are lower with this model despite having larger squared errors. Hence, the 8-feature model makes smaller errors in predictions, making it more accurate in terms of overall prediction precision. Again, most of our dataset target values are 0 which leads to the models learning a more realistic approach to predicting the target resulting in overall high squared error due to values that are different. For example, in the figure [5.26] it can be seen that very rarely some users have a frequency above 20 which the model struggles to predict possibly due to the training process without extreme outliers, resulting in a larger error. However, the other predictions can be observed from the figure that those that explain most of our dataset have been more accurately identified by our model. In contrast, from figure [5.23]it can be seen that this model struggles to even predict the values with lower frequencies which explains most of our dataset resulting in a lower MAE. Although the 8-feature model has a higher RMSE and R-squared, these metrics are not drastically worse, and the improvement in MAE suggests a favorable trade-off. Moreover, it can be seen that the R-square of this model is slightly lower than the 6-feature model. However, given that the model incorporates more features, it is likely to improve with further training or optimization. Additionally, with more features, it captures more information about the data. This improves generalization and creates the opportunity to fine-tune or optimize the model for even better performance. Hence, we chose the 8-feature model due to its overall strong theoretical foundations and practical results that make it a viable option.

Final Multivariate Model Performance Evaluation

In the 8-feature mode, we can see from table [5.4] that it has an MSE of 0.389 which suggests some acceptable prediction errors in our case since [5.26] explains the reasons for these large errors that occurred due to only a small number of data points. Unlike MSE, MAE doesn't square the errors, so it doesn't penalize larger errors as heavily which is why we saw a decent number of MAE of 0.0887 which suggests that on average, the model's predictions are off by around only 0.0887 units. Furthermore, RMSE tends to be higher than MAE when the model makes a few large errors because those errors are magnified by the squaring operation in MSE. These larger errors affect RMSE more than they affect MAE. In our model, the RMSE (0.623931) is notably higher than the MAE (0.08874),

which suggests that the model made some predictions with larger errors as seen in [5.26], increasing the RMSE disproportionately compared to MAE. Moreover, the R-squared is 0.35704 for this model, suggesting that the model explains 35.7% of the variance in the data. However, we understand that due to not imputing, the majority of our data points are filled with zeros and the model might have difficulty capturing those inconsistencies. However, we believe it still performs well and can be improved by training on more data.

5.2.2 Frequency Forecasting using Tweet Content

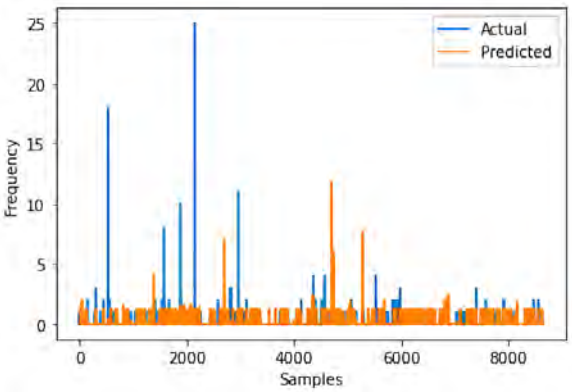

Figure 5.27: Prediction for Mean Pooling

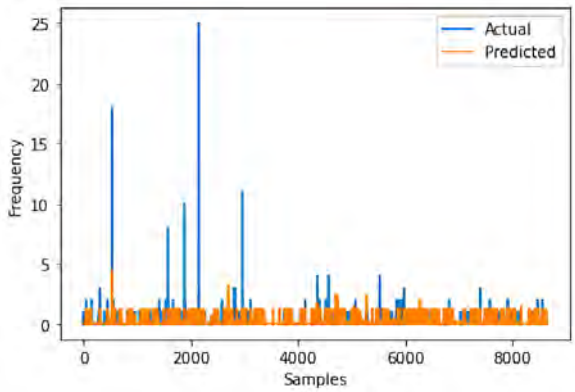

Figure 5.28: Prediction for Max Pooling

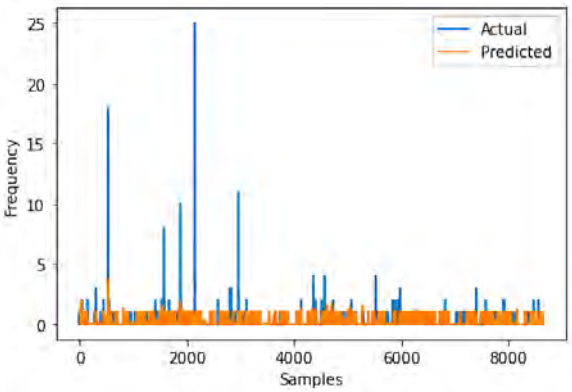

Figure 5.29: Prediction for Concatenation

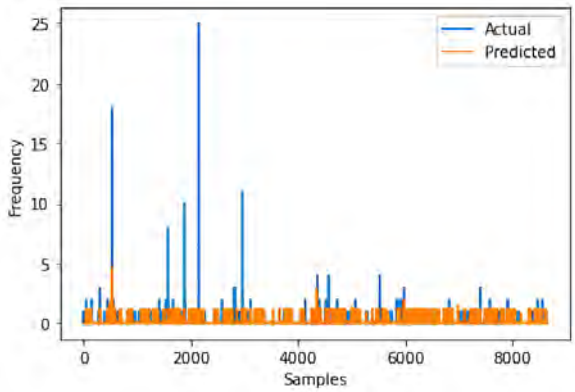

Figure 5.30: Prediction for Element-wise Addition

Approach	MSE	RMSE	MAE	R-squared
Concatenation	0.1481	0.3849	0.0444	0.2287
Mean Pooling	0.1935	0.4399	0.0497	-0.0075
Max Pooling	0.1466	0.3824	0.0489	0.2364
Addition	0.1454	0.3813	0.0425	0.2431

Table 5.5: Model performance comparison across different metrics.

Comparison of the models

All the models were run on the same data to keep the comparisons as fair as possible. As can be seen from the table [5.5], all the models show remarkably similar results. Furthermore, the range of values that we are trying to predict as can be seen from the graphs [5.28], [5.27], [5.29] and [5.30] is 0 to 25 with the majority of the data being within smaller frequencies, so our errors being below 1 showcase pretty good learning capability from all the models. However, as previously stated, we chose not to use the concatenation model as there might be information loss in the case of multiple tweets in an hour resulting in a concatenated tweet exceeding the size of 512 which is the maximum BERT input size. This is further validated by looking at the predictions, where even though the metrics show quite good results for concatenation, the predicted graph showcases a complete inability to detect outlier situations. Following this, we can also safely rule out mean pooling as it gives the worst results with the highest errors as well as a negative R-squared, indicating a terrible capability to detect variations in the data. That leaves the choice between max pooling and element-wise addition. Looking at the metrics as well the graphs [5.28] and [5.30] we can see that both approaches give nearly identical results, so it comes down to the metrics where the addition approach gives slightly lower errors and a slightly higher r-squared which is why we chose the addition approach for our final model.

Final Text Content Model Performance Evaluation

Finally, we trained our addition model on the tweet content training data and then used it to predict our testing data in order to compare and combine it with the multivariate LSTM model.

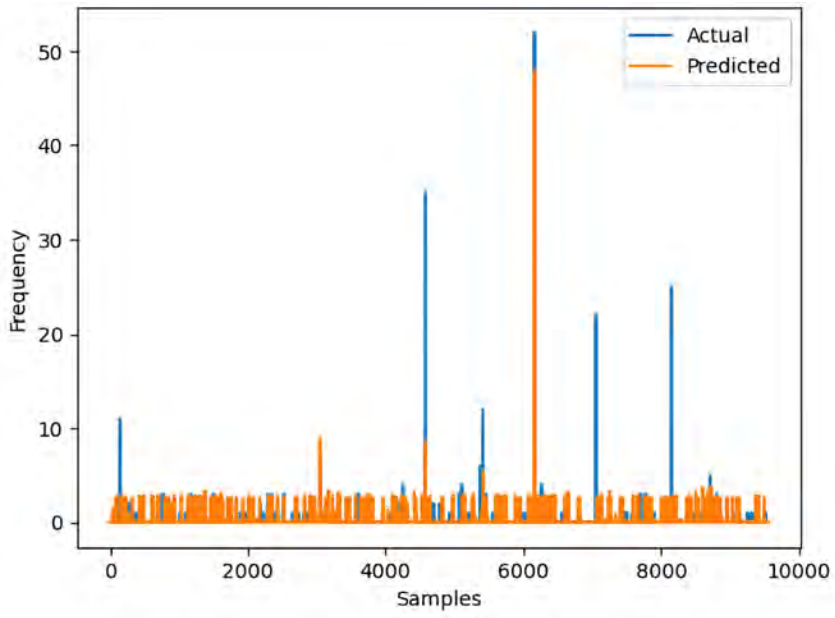

Figure 5.31: Prediction for Element-wise Addition on final testing data

As we can see from the graph, this model works quite well in predicting our test data. It gives an MSE of 0.3256, RMSE of 0.5706, MAE of 0.0755, and R-squared value of 0.4622. In the test data, our range of test data is 0 to 52 but the majority of data being within smaller frequencies shows our model is fitting into our data pretty well. Despite having some outliers which are high frequency values, which is the reason for the error increasing

the model performs within an acceptable range and can be seen from the figure [5.31] that, it predicts most of the frequencies almost accurately. Additionally, the increased R-squared shows the capability of the model to capture variations in this data quite well.

5.2.3 Fusion Ensemble

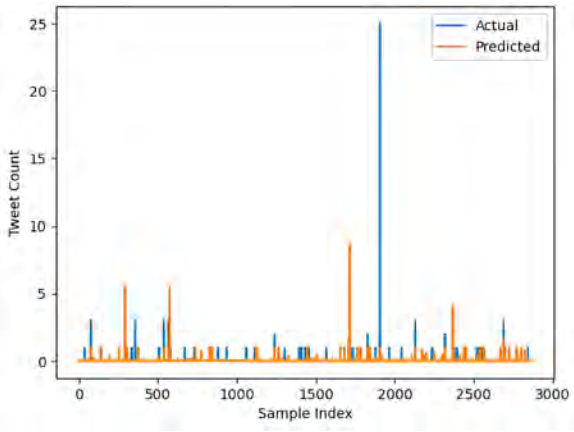

Figure 5.32: Fusion Ensemble with Random Forest Regressor

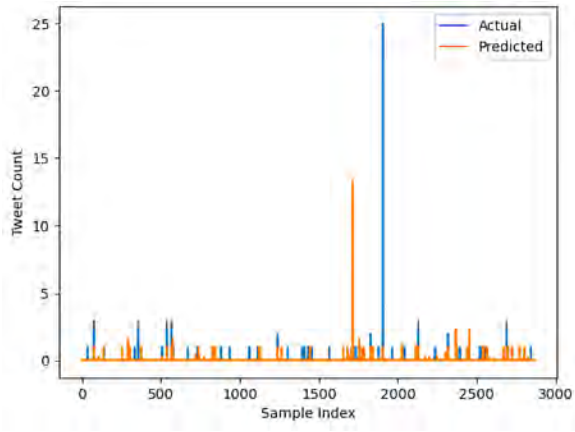

Figure 5.33: Fusion Ensemble with Cat-Boost Regressor

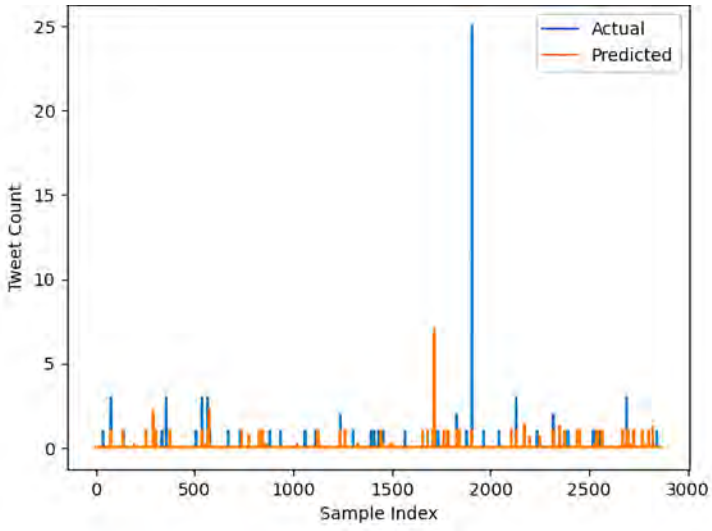

Figure 5.34: Fusion Ensemble with Gradient Boosting Regressor

Model	MSE	RMSE	MAE	R-squared
Random Forest	0.32288	0.56822	0.05200	0.08244
Gradient Boosting Regressor	0.29221	0.54056	0.04709	0.16959
Cat Boost Regressor	0.31320	0.55964	0.05263	0.10994

Table 5.6: Test Set Performance Metrics for Models

Comparison of Fusion Ensemble Models

At first glance, when looking at the table [5.6], after combining results from the previous LSTM models, the fusion ensemble model's performance seems ambiguous. However, looking at things a little closer, it seems every single one of the fusion models manages to lower the errors across the board for all 3 models from our individual final models. Again, our range of test data this time is from 0 to 25, with the majority being smaller data so our results showcase decent ability to predict our test set. Gradient Boosting Regressor has the lowest MSE, indicating it makes smaller squared errors compared to the others. This model also has the lowest RMSE suggesting better accuracy in minimizing large errors. Again it outperforms the others with the lowest MAE, indicating the smallest average error in absolute terms. Thus, the Gradient Boosting Regressor is the best overall model in comparison to the other fusion ensemble models [5.34].

Model Performance Evaluation

In comparison with the two LSTM models from [5.4] and [5.5], the Gradient Boosting Regressor has the smallest MSE of 0.29221 [5.6] indicating that on average, its squared prediction errors are smaller than those of both LSTM models. This fusion model again outperforms both LSTM models, showing the smallest error on average in terms of RMSE of 0.54056. Moreover, it has a significantly lower MAE of 0.047 suggesting the smallest average absolute prediction error. However, both LSTM models have higher R^2 scores compared to the Gradient Boosting Regressor which is explainable by the fact that the GBR was trained on a very small fraction of the data, only 70% of the predictions gathered from the output of both LSTM models test set of 10 thousand points. To clarify, the initial predictions are trained on the entire training dataset, but the ensemble models are only trained on a portion of the predictions. This is very likely to result in the ensemble models not being able to fully capture variations in data due to weights being updated on a smaller percentage of the total data. Hence, the r-square resulted in a low value in this case. However, our underlying knowledge of this setting establishes that the fusion ensemble model served its purpose of improving the model performance significantly.

6. Limitations and Future Work

Limitations

- **Limited Data Time-frame:** The dataset only covers tweets over a 39-hour period, making it difficult for our models to forecast longer time periods. Observed patterns and behaviors are particular to short-term trends rather than long-term phenomena due to the lack of extended data.
- **Hashtag Bias:** The analysis focuses on specific hashtags, such as #bitcoin, which may not accurately reflect the range of conversations surrounding cryptocurrencies. Additionally, trends in other domains were not studied due to the focused data collection on the #bitcoin trend. There is definitely room in the research to further study more comprehensive datasets with other trends centered around different hashtags.
- **Language Constraints:** Only English tweets were included in the analysis, which excludes discussions conducted in other languages. This limits the study's comprehensiveness, especially when examining the behavior of non-English speaking users.
- **Missing Contextual Data:** The dataset lacks additional contextual information about the users, such as geographic location, socioeconomic background, or the motives behind their tweets. This information could have provided deeper insights into hashtag propagation across diverse demographics.
- **Limited Sentiment Analysis Scope:** Sentiment analysis was conducted using pre-trained models like ALBERT. However, these models might overlook small but important perspectives related to financial concepts, such as Bitcoin, potentially affecting the accuracy of sentiment analysis results.
- **Community Detection Resolution:** The community detection algorithm uses a fixed resolution parameter (set to 2). Despite attempts at fine-tuning, the chosen resolution may lead to over-splitting or under-detection of communities.
- **Platforms Outside of Twitter:** The analysis focuses entirely on Twitter interactions, neglecting the impact of other popular platforms, such as Reddit or YouTube, which can significantly influence cryptocurrency debates and trends.
- **Outlier Handling:** Removing users with tweet frequencies greater than 100 may exclude influential users or bots, skewing the analysis and limiting the model's ability to handle outliers in real-world data. However, due to the limited number of extreme outliers in the dataset, this approach was followed.
- **Frequency Prediction:** We have predicted only the user tweet frequency of the next hour in order to predict the popularity of a hashtag due to the limited time when other engagement prediction operations could be done.

Future Work

- **Extending the Data Collection Period:** To obtain a broader view of trends, future studies can collect data over longer periods. Analyzing hashtags over time will provide more insight into long-term trends, user behavior, and interaction patterns.
- **Incorporating Multilingual Data:** Expanding the dataset to include tweets in multiple languages will enable a more comprehensive analysis of global discussions. This will enable research to properly analyze each trend without losing information.
- **Exploring Other Social Media Platforms:** Social trend discussions are not limited to Twitter. Hence, using data from other platforms such as Reddit, YouTube, and Instagram and analyzing trend propagation from those platform will further validate the research to understand whether these trends behave the same way on all social media platforms.
- **Analyzing Leader Influence Beyond Degree Centrality:** Degree centrality does not provide a complete picture on leaders, so future research could explore more complex metrics, such as betweenness centrality, which could provide a deeper understanding of leadership within communities.
- **Incorporating Bot Detection Mechanisms:** Since this research is trying to predict the propagation of information by humans and not bots, integrating bot detection techniques into the analysis could improve the accuracy of future studies.
- **Different Domain Study:** Although this research focused on cryptocurrency-related hashtags, future studies could apply similar methodologies to other domains such as politics, entertainment, or global events. This would help generalize findings and determine whether similar patterns emerge across various fields.
- **Incorporating additional Features:** The LSTM model could be enhanced by incorporating additional features, such as tweet sentiment, user interaction history, time of day, or location. The model could especially be provided with additional graphical features such as the degree for each user, the centrality for each user, and more, which might not only bridge the gap between the networks and the predictions better but also improve the model understanding of the propagation of trends.
- **Further Experimentation with models:** Experimenting with other neural network architectures, such as Transformer models or attention mechanisms, could improve prediction accuracy. There also remains room to experiment with other unsupervised machine-learning techniques on multiple datasets of different events to find the best predictor for the popularity of hashtags.
- **Adopting Siamese Network for Combining Predictions:** Although regression models were used for fusion ensemble in this research, future studies could explore the use of Siamese networks instead of regressors to potentially improve prediction accuracy.
- **Engagement Predication:** Future work can build models to predict users' retweets and likes per hour in order to understand engagement better and predict the propagation of a hashtag from different dimensions.

7. Conclusion

This research explored the #Bitcoin trend from a Twitter dataset to analyze hashtag-driven trends on social media. Using a combination of network science, natural language processing (NLP), and machine learning techniques, the study analyzed and predicted the propagation of hashtags and by extension user engagement patterns. More specifically, the research used community detection algorithms such as Greedy Modularity Maximization to identify distinct clusters of user interaction, offering insight into how hashtags influence information propagation and discourse. The study also looked into identifying the cluster leaders and finding the sentiment of the information they primarily propagate. Additionally, predictive models like Long Short-Term Memory (LSTM) were used to forecast user activity and the effectiveness of combining textual and numerical data to enhance trend forecasting accuracy was showcased.

Despite having successful results, the research did have some limitations that can be filled in future works. Future work can achieve its potential to analyze any trend for multiple purposes in different domains such as politics, finance, war, advertising, and many more. Further, the trend analysis can expand to other social media platforms like Reddit, Instagram, and Facebook which can further validate the research.

To sum up, this research demonstrates the effectiveness of hashtags as a tool for analyzing and predicting social trends, providing valuable insights for applications in marketing, media studies, and social influence analysis.

Bibliography

[1] A. Clauset, M. E. J. Newman, and C. Moore, "Finding community structure in very large networks," *Physical Review E*, vol. 70, no. 6, p. 066 111, 2004.

[2] S. M. Kywe, T. A. Hoang, E.-P. Lim, and F. Zhu, "On recommending hashtags in twitter networks," in *Social Informatics: 4th International Conference, SocInfo 2012*, 2012, pp. 337–350.

[3] C. I. Muntean, G. A. Morar, and D. Moldovan, "Exploring the meaning behind twitter hashtags through clustering," in *Business Information Systems Workshops: BIS 2012 International Workshops and Future Internet Symposium*, 2012, pp. 231–242.

[4] O. Tsur and A. Rappoport, "What's in a hashtag? content-based prediction of the spread of ideas in microblogging communities," in *Proceedings of the fifth ACM international conference on Web search and data mining*, 2012, pp. 643–652. DOI: 10.1145/2124295.2124389.

[5] Z. Ma, A. Sun, and G. Cong, "On predicting the popularity of newly emerging hashtags in twitter," *Journal of the American Society for Information Science and Technology*, vol. 64, no. 7, pp. 1399–1410, 2013.

[6] R. Socher, A. Perelygin, J. Y. Wu, *et al.*, "Recursive deep models for semantic compositionality over a sentiment treebank," in *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, 2013, pp. 1631–1642. DOI: 10.3115/v1/D13-1170.

[7] M. Chen, K. Kuzmin, and B. K. Szymanski, "Community detection via maximization of modularity and its variants," *IEEE Transactions on Computational Social Systems*, vol. 1, no. 1, pp. 46–65, 2014. [Online]. Available: https://web.eng.ucsd.edu/~massimo/ECE227/Handouts_files/TCSS-14-Modularity.pdf.

[8] K. W. Lim and W. Buntine, "Twitter opinion topic model: Extracting product opinions from tweets by leveraging hashtags and sentiment lexicon," in *Proceedings of the 23rd ACM international conference on information and knowledge management*, 2014, pp. 1319–1328.

[9] A.-L. Barabási, *Network Science*. Cambridge University Press, 2016.

[10] Y. Gong and Q. Zhang, "Hashtag recommendation using attention-based convolutional neural network," in *Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI)*, 2016, pp. 2782–2788.

[11] Y. Li, T. Liu, J. Jiang, and L. Zhang, "Hashtag recommendation with topical attention-based lstm," in *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, 2016, pp. 3019–3029.

[12] R. Wang, W. Liu, and S. Gao, "Hashtags and information virality in networked social movement: Examining hashtag co-occurrence patterns," *Online Information Review*, vol. 40, no. 7, pp. 850–866, 2016.

[13] O. Varol, E. Ferrara, F. Menczer, and A. Flammini, "Early detection of promoted campaigns on social media," *EPJ Data Science*, vol. 6, pp. 1–19, 2017.

[14] A. Reyes-Menendez, J. R. Saura, and C. Alvarez-Alonso, "Understanding# worldenvironmentday user opinions in twitter: A topic-based sentiment analysis approach," *International Journal of Environmental Research and Public Health*, vol. 15, no. 11, p. 2537, 2018.

[15] İ. Türker and E. E. Sulak, "A multilayer network analysis of hashtags in twitter via co-occurrence and semantic links," *International Journal of Modern Physics B*, vol. 32, no. 04, p. 1 850 029, 2018.

[16] A. Géron, *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*, 2nd. O'Reilly Media, 2019, ISBN: 9781492032649.

[17] F. A. Rodrigues, "Network centrality: An introduction," in *A Mathematical Modeling Approach from Nonlinear Dynamics to Complex Systems*, ser. Nonlinear Systems and Complexity, E. E. Macau, Ed., vol. 22, Springer, Cham, 2019, pp. 177–196. DOI: 10.1007/978-3-319-78512-7_10. [Online]. Available: https://doi.org/10.1007/978-3-319-78512-7_10.

[18] Y. Zhang, "Language in our time: An empirical analysis of hashtags," in *Proceedings of The World Wide Web Conference*, 2019, pp. 2378–2389.

[19] O. Calzone, "An intuitive explanation of lstm," *Medium*, 2020. [Online]. Available: https://medium.com/@ottaviocalzone/an-intuitive-explanation-of-lstm-a035eb6ab42c.

[20] I. J. Cruickshank and K. M. Carley, "Characterizing communities of hashtag usage on twitter during the 2020 covid-19 pandemic by multi-view clustering," *Applied Network Science*, vol. 5, pp. 1–40, 2020.

[21] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, "Albert: A lite bert for self-supervised learning of language representations," in *International Conference on Learning Representations (ICLR)*, 2020. [Online]. Available: https://paperswithcode.com/paper/albert-a-lite-bert-for-self-supervised.

[22] D. Rothman, *Transformers for Natural Language Processing*. Packt Publishing, 2020. [Online]. Available: https://www.packtpub.com/product/transformers-for-natural-language-processing/9781839212910.

[23] D. Antonakaki, P. Fragopoulou, and S. Ioannidis, "A survey of twitter research: Data model, graph structure, sentiment analysis and attacks," *Expert Systems with Applications*, vol. 164, p. 114 006, 2021.

[24] K. Petersen and J. M. Gerken, "#covid-19: An exploratory investigation of hashtag usage on twitter," *Health Policy*, vol. 125, no. 4, pp. 541–547, 2021.

[25] J. Camacho-Collados *et al.*, "Tweetnlp: Cutting-edge natural language processing for social media," *arXiv preprint arXiv:2206.14774*, 2022.

[26] N. Developers, *Greedy modularity communities*, https://networkx.org/documentation/stable/reference/algorithms/generated/networkx.algorithms.community.modularity_max.greedy_modularity_communities.html, Accessed: 2024-10-08, 2023.

[27] D. Jurafsky and J. H. Martin, *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*, 3rd. Pearson, 2023.

[28] NLTK Project, *Nltk wordnet lemmatizer*, Accessed: 2024-10-12, 2024. [Online]. Available: https://www.nltk.org/api/nltk.stem.wordnet.html?highlight=lemmatize.