5-1-2023

Utilizing Multi-modal Weak Signals to Improve User Stance Inference in Social Media

T M Miuru Bhashithe Abeysinghe

Utilizing Multi-modal Weak Signals to Improve User Stance Inference in Social Media

by

Bhashithe Abeysinghe

Under the Direction of Rajshekhar Sunderraman, Ph.D.

A Dissertation Submitted in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

in the College of Arts and Sciences

Georgia State University

2023

ABSTRACT

Social media has become an integral component of the daily life. There are millions of various types of content being released into social networks daily. This allows for an interesting view into a users' view on everyday life. Exploring the opinions of users in social media networks has always been an interesting subject for the Natural Language Processing researchers. Knowing the social opinions of a mass will allow anyone to make informed policy or marketing related decisions. This is exactly why it is desirable to find comprehensive social opinions.

The nature of social media is complex and therefore obtaining the social opinion becomes a challenging task. Because of how diverse and complex social media networks are, they typically resonate with the actual social connections but in a digital platform. Similar to how users make friends and companions in the real world, the digital platforms enable users to mimic similar social connections.

This work mainly looks at how to obtain a comprehensive social opinion out of social media network. Typical social opinion quantifiers will look at text contributions made by users to find the opinions. Currently, it is challenging because the majority of users on social media will be consuming content rather than expressing their opinions out into the world. This makes natural language processing based methods impractical due to not having linguistic features. In our work we look to improve a method named stance inference which can utilize multi-domain features to extract the social opinion. We also introduce a method which can expose users opinions even though they do not have on-topical content.

We also note how by introducing weak supervision to an unsupervised task of stance inference we can improve the performance. The weak supervision we bring into the pipeline is through hashtags. We show how hashtags are contextual indicators added by humans which will be much likelier to be related than a topic model. Lastly we introduce disentanglement

methods for chronological social media networks which allows one to utilize the methods we introduce above to be applied in these type of platforms.

INDEX WORDS: Natural Language Processing, Social media, Opinion mining, Stance inference, LSTM, Transformer, BERT, Sentence transformer, Language embedding

Utilizing multi-modal weak signals to improve user stance inference in social media

by

Bhashithe Abeysinghe

Committee Chair:	Raj Sunderraman
Committee:	Anu Bourgeois
	Jonathan Ji
	Dror Walter
	Raj Sunderraman

Electronic Version Approved:

Office of Graduate Studies

College of Arts and Sciences

Georgia State University

May 2023

DEDICATION

This dissertation is dedicated to the important people in my life who have supported me throughout my academic journey.

To my loving wife, thank you for your unwavering belief in me and your constant support. Your encouragement and love have been my greatest source of inspiration. I remember how interested you were in listening to my ramblings and providing much needed criticism. Thank you for pushing me for greater things and this work would not have been possible without your encouragement.

To my advisor, Dr. Raj, thank you for your guidance, mentorship, and support throughout my graduate studies. Your knowledge and expertise have been invaluable in shaping my research.

To my parents, Amma and Appachchi, thank you for everything you have done for me. To my siblings, Rashmie and Praveeni, thank you for your constant support and belief in me. Your unconditional love, support, and sacrifice have been the foundation of my success.

To my friends in Atlanta, thank you for your support throughout this journey. Your friendship has been a source of joy and laughter, and I am grateful for the memories we have shared.

This work is dedicated to all of you.

ACKNOWLEDGMENTS

I would like to express my gratitude to my advisor, Dr. Raj Sunderraman, for his unwavering support and guidance throughout my graduate studies. Thank you for always having the big picture in your mind and shaping my rough ideas to be presentable. Your insightful feedback and constructive criticism have been invaluable in shaping my research and writing.

I would also like to thank my committee members, Dr. Anu Bourgeois, Dr. Jonathan Ji and Dr. Dror Walter, for their feedback and suggestions on my dissertation. Your expertise and knowledge have helped me to improve my work significantly.

I would also like to express my gratitude to Dr. Mia Bloom for providing me with guidance and support.

To my friends Kiril, Hui, Dhara, Heta, Anuja, Slava, Ayse, Kristian, Krishanu, Ramya, Saad thank you for your friendship, support, and encouragement throughout my graduate studies. Your enthusiasm and positive energy have made this journey much more enjoyable.

This work is a result of the contributions and support of all these individuals, and for that, I am immensely grateful.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

1.1 Social Media Platforms

Analyzing social media is a very important industry in today's world due to the attention it has gotten [9] and the amount of content which social media networks are generating per day [44]. There are multiple perspectives we can look at while investigating social media. Marketing is one such area that we can take as an example and it is possible to dig into the vast amounts of publicly available data and discover new knowledge about customer buying patterns, understand how the industry is evolving and even find out what the competitors are up to [44]. Initial models are typically sentiment analysis [32, 6, 114] on certain topics, products etc. and then models evolved into various other applications like "chatbots" which can maintain a proper discourse with humans, summary generators etc. This section is a brief overview of why we use the technology and methods we used along with an introduction to social media.

First we take a look at some selected social media platforms and categorize them according to our specifications. There are two types of Social Media Platforms we are investigating, first is the micro-blogging platforms. These platforms a user will have access to a *timeline* of posts from various other users that they are following. They can comment, like or share these posts with their own following, this is similar to a community network. Examples for

this platform type are Twitter [1], Facebook[2] and Parler[3].

Other type is significantly different on how a user can consume information [92], in this type a user typically joins a group of users which can be open to public or private. Users in that group can share posts to whomever in that group, other users cannot typically like or share messages posted in that group but they can reply to the messages. The difference between this and the initial platform is that the messages are typically ordered chronologically and the group may discuss multiple topics at any given time. The users who are joining after a hiatus may not have a good idea about the previous topics if they choose not to scroll up and down in a group. Due to the setup of this type of platforms, they are massively used in marketing and unfortunately spreading misinformation [85]. Examples for this type of platforms are Telegram[4] and Whatsapp[5].

To analyze above two types of platforms, various methods can be used; To investigate text content Natural Language Processing (NLP) Can be used. Other methods such as Graph Mining and Computer Vision are also parts of social media analysis, but here we focus on NLP and Graph Mining. In many ways NLP has adapted methods from other domains and there are lots of similarities of technologies that are used in the two paradigms. Also, NLP is dissimilar from computer vision in many ways. We as humans may see the world using colors in different intensities, textures, objects etc. Humans typically name objects using a Language, depending on where the person is from and how they name the object and the

[1]https://twitter.com
[2]https://facebook.com
[3]https://parler.com
[4]https://telegram.org
[5]https://www.whatsapp.com

character set of the language may have a stark difference to each other. Moreover, computer vision models are somewhat portable around the world while NLP models are not so because of the difference of the language. Due to this nature of the language there are many use cases and tasks for the NLP domain which we shall explore in the next section.

There are typical tasks that NLP has been extensively used. We shall look into some related tasks in the next.

1.2 Background

In the domain of Information Retrieval, natural language processing has been critical in exploring new avenues. It has been critical in many areas of research such as spam detection [15, 53, 68, 95, 104], spelling correction [23, 41, 69, 79], language modelling [24, 83] etc. and has seen methods which had revolutionized the field. In this section a brief introduction into a non-exhaustive list of tasks is discussed.

1.2.1 Natural Language Processing Tasks

The following discussion will enumerate 3 tasks which cover relevant areas for the topic of social opinion mining. We shall also look at what methods have excelled in each of these tasks and also some interesting applications of them.

1.2.1.1 Text Classification

Text classification (TC) is one of the most popular tasks in NLP, it is the process of categorizing text into organized groups. There are many sub categories in the task of TC such

as News Filtering and Organization, Document Organization and Retrieval, Opinion Mining, Email Classification and Spam Filtering, Sentiment Analysis, Topic Analysis, Question Answering, Natural Language Inference [10, 73]. Among a huge volume of work Aggarwal and Zhai in [10] gives a concise description of many shallow methods and in [73] digs deep into deep learning based text classifications. Minaee et al. shows that Text Classification methods can be categorized into two as 1) Rule Based and 2) Data Driven methods. All machine learning and deep learning methods come under the latter. Most shallow and early models for text classification used hand-crafted feature set like a Bag-of-Words or its extensions and derivations. Then traditional machine learning or statistical learning methods such as Support Vector Classifiers, Hidden Markov Models can be used to classify. This two phase method had its flaws, such as the feature engineering step needs to be conducted by a subject expert and will typically be a tedious task, also the performance of these models depended a lot on the domain knowledge of people working in feature engineering. Because of this bottlenecks it is apparent how these type of models are hard to generalize into various tasks.

To solve above issues, deep learning has been used. The neural network architectures used in these work ranges from a simple Feed-forward network to Convolutional Neural Networks (CNN), Graph Neural Networks, Recurrent Neural Networks (RNN), and Transformers [73, 103]. In the past Long Short Term Memory a type of RNNs has been dominant in the world of NLP [118] but slowly and surely Transformers are breaking the state-of-the-art in all sub tasks in TC [97]. Joulin et al. in [55] shows how well a simple method can be trained fast

and predict fast compared to deep learning based methods. Even though simple and shallow methods will not outperform deep models, they will still have this advantage over the large data driven methods.

1.2.1.2 Named-Entity Recognition

When retrieving information from documents it is also essential to recognize elements like names, organizations, locations etc. according to Nadeau and Sekine in [75]. Authors also mention that most of the Named-Entity Recognition (NER) research is focused in English. This refers to one of the previous issues we have raised, but since 2007 when Nadeau and Sekine published [75], there has been a lot of studies done on the domain to mitigate this issue but still the language factor is a huge issue in recognizing Named-Entities. Success in NER is key in other tasks in NLP as well [112], meaning that tasks such as question answering, information retrieval uses NER to come up with better models.

Yadav and Bethard discusses many types of NER systems available. Feature-Engineered Supervised, Feature-Inferring Neural Networks, Word Level (RNNs) were interesting to us. Feature-Engineered supervised systems will replace rule based systems which are highly language dependent and biased. Some typical features which are used in this are Capitalization, Special Characters in use, whether a word is a first of a sentence and whether a word has appeared with a known last name etc. Lample et al. shows how they have used two neural models to battle NER without using language-specific knowledge or resources but they are able to do NER on German, Dutch, English and Spanish. Lample et al. has shown very good performance in English while other results still outperform existing work they are not

as good as English.

More recent work by Li et al. discusses about metrics that are typically in use in these NER systems. Older work used human annotation against the NER system but since then evaluation metrics such as relaxed-match has been in use as well. Li et al. in [64] discusses that neural architectures which are in use to this day are mostly RNN based models because those yield the best performance, but Transformer based models such as [113] has come very close to dethroning RNN models.

1.2.1.3 Information Retrieval

This is a versatile field of NLP, any task can be posed as Information Retrieval (IR) because of the same reasons we have mentioned above. IR is the task of obtaining some information resources relevant to an information need from within a large collection. IR is used in web search engines digital libraries, expert finding etc. Guo et al. in [42] discusses about how important is a ranking algorithm for IR. There are many which they discuss about such as learning to rank (LTR) [62] and Collaborative Filtering [45, 96]. Similar to NER, IR also plays a critical role in Question Answering systems. Therefore success in question answering depends somewhat on IR as well. Using deep learning for IR mostly is at the learning distributed representations for linguistic units [63]. Lam et al. in [60] shows results of a bug localization system using IR a deep neural network combined with rVSM from [119].

1.2.2 Deep learning in social media

Information retrieval, named entity recognition and text classification all are very important in social opinion mining. For our study it is important to know what type of work is being done in respective of social media platforms. Therefore in the following sections we explore deep learning based and other interesting work, we categorize this into two using our previous social media categorization.

1.2.2.1 Micro-blogging platforms

Tang et al. in [100] discuses about a deep learning technique applied to sentiment classification. Here Tang et al. show how they were able to learn sentiment specific word embeddings and some hand crafted features which in turn used as input for a sentiment classifier, which is a dense neural network. Using SemEval[6] data they were able to obtain 87.61 F-1 score. Tang et al. provides a simple fully connected network which was able to receive acceptable performance levels. There are works that uses Convolutional Neural Networks (CNN) to handle twitter data such as [52]. They have used GloVe embedding instead of handcrafted features therefore the CNN will be selecting features without human intervention.

There are other applications of twitter data classifications which uses superficially similar methods to the above work. Such as looking into spam detection using CNNs and detecting depressed Twitter users. What is most important to us however is the work done in user opinion mining at Twitter like platforms, most of the opinion mining methods are based off of sentiment classification or stance inference. Both of these tasks fall into text classification

[6]https://semeval.github.io/

task under the topics we covered in the previous section.

User reaction to any type of information on Twitter can come in many forms. Twitter allows user accounts to compose tweets, reply to other tweets, like or retweet other tweets and quote tweets. All these can be considered as a reaction for some event or a target. Users who are reactive to the twitter chatter may compose tweets or reply to other tweets which exposes their opinions. These could very easily be captured using traditional NLP methods and also deep learning text classification methods [32, 98, 100, 114, 116]. These methods typically use features which are extracted by a LSTM or a CNN and then use some form of a deep learning or machine learning classifier such as fully connected networks or SVMs. They tend to have very good performance even in shorter text from social media networks [98].

There are other types of reactions on social media such as following or liking some other users because they agree or align with your own ideology. This means that some of these opinions will not be purely linguistic but rather they will be nonlinguistic user interaction such as following, liking and retweeting. All reactions are dependent on the user's stance on the considering event, if the user is supportive of some topic or theme, the reaction can come as a positive reaction and vice versa. Therefore, user stance has been studied as an indication of marketing strategies, political campaigns or just to measure the public opinion [12, 29].

1.2.2.2 Chronological Platforms

Not as much work is available for this type of platforms. Caetano et al. in [26] provides good insight on how a hierarchical characterization done on WhatsApp. Caetano et al. analyze data in three layers; message, user and group layers. While Caetano et al. do not use deep learning or extend their user layers into multiple groups their comprehensive analysis had us draw inspiration from the layered architecture. Similar work reflecting micro-blogging platforms are available in this platform type as well, such as the sentiment analysis [92]. Smuts shows the predictive power of the collected datasets using series of LSTM models and shows that they are the most accurate to find the crypto-currency trends. Ng and Loke in [77] shares very important work on analyzing misinformation and public opinion in telegram groups, Ng and Loke also discusses interesting questions such as how users react to fact checked misinformation in these groups. Kaur in [56] discusses different levels of misinformation such as deliberate, clickbait, parody or satrical and presents a shallow method to automatically classify them. On other types of applications Jarynowski et al. in [50] uses BERT retrained on a Russian Language Corpora in order to extract Named Entities from Telegram chats. Using this Jarynowski et al. had found the adverse effects of Sputnik V vaccine.

From the literature it is apparent that most NLP tasks can be done on chronological platforms, but one thing to note is that the nature of message ordering prevents one from looking at the community graph like structure available in a micro-blogging platform. Therefore, this prevents us from extracting proper user stances from this type of social media networks.

In order to convert the chronological structure into a threaded network we have done some research under the disentanglement domain and in give more details about it in the last chapter.

In this dissertation we present 4 articles which are tied together by the same goal of extracting user stances on social media networks. We propose the usage of user stances for public opinion mining as a better version of opinion extraction than sentiment analysis. Our arguments for this is based on the fact that the majority of social media users are not participating in the conversation. In order to extract opinions of such users we cannot use traditional natural language processing methods. To extract the social opinions of users, we go through the route of weak supervision. Further, we have created our methods by first extracting opinions of participating users and then by observing the social connections and interactions made by non-participating users we estimate their stances as well. To get to this we had to conduct multiple research experiments which spread out into multiple research articles, following figure 1.1 is the road map of our whole dissertation.

Figure 1.1 explains how we collect our data using the twitters academic API and how we process them to extract tweets, users and then articles. Afterwards we produce weak signals for user stances using unsupervised methods which can be consumed by the label propagation method. We also explain how a chronological social media network could be converted into a graph structure in the last paper which then could be sent in to the label propagation method to find user stances as well.

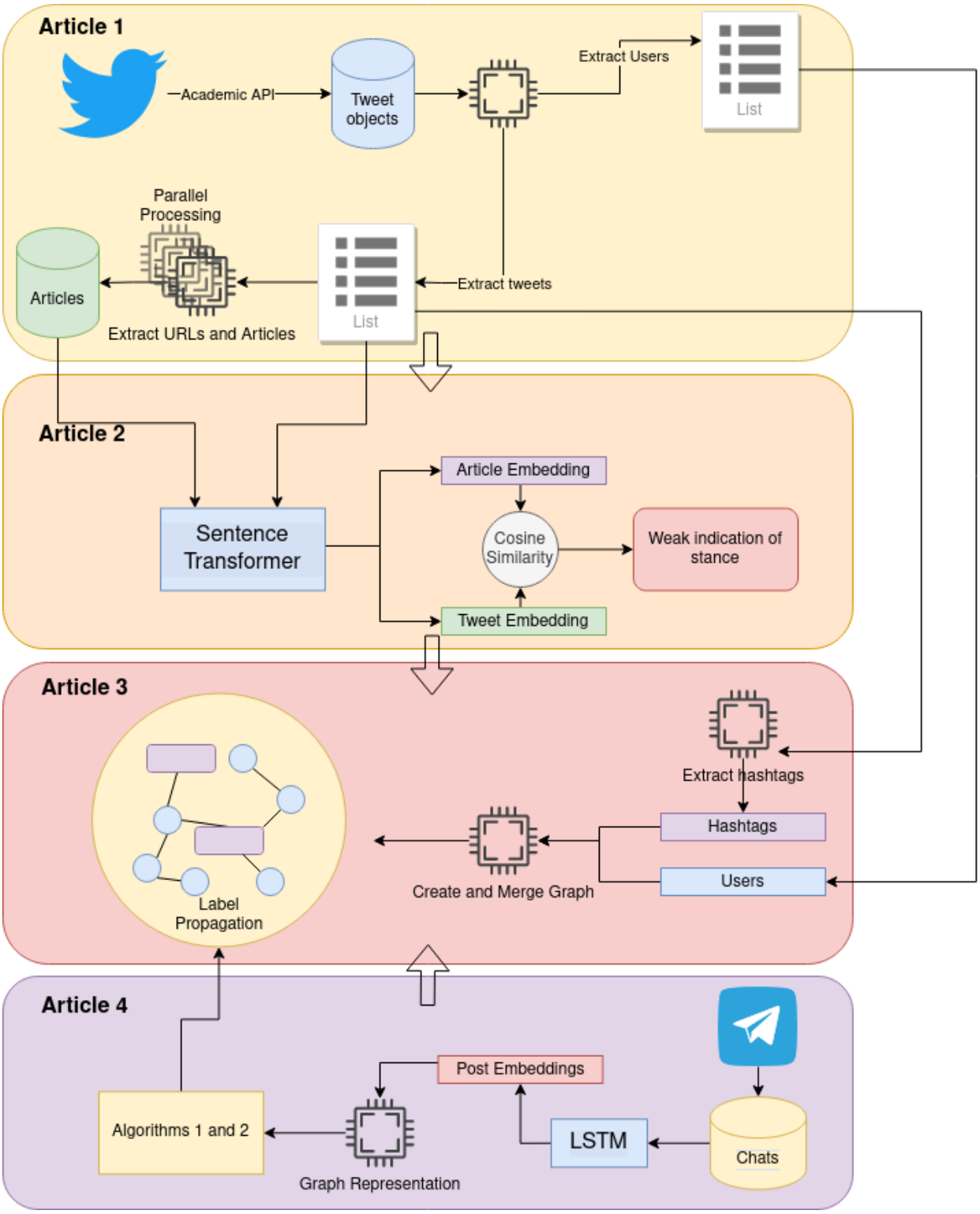


Figure 1.1 Road map of the research conducted in this dissertation

CHAPTER 2
PROBLEM STATEMENT

Stance detection can be introduced as a users' opinion towards a specific target [14, 29, 74, 87]. As we discussed previously, there are multiple levels of stance detection. Statement level stance detection is the simplest where it is needed to study the statements' alignment toward some topic. A more comprehensive stance analysis would be the user level stance inference. In this case we rather analyze a users' alignment towards some topic. Quantifying human opinion as a whole is much more challenging than just analyzing a statement. Statement level stance detection can be reintroduced as tweet level stance detection in our use case because our data is based on Twitter. Tweet level stance detection assumes that the users' stance can be inferred from one tweet. There are several issues associated with identifying the stance of the user. First is a naive model could assume stance is similar to sentiment of the text, but in fact it is not the case [13]. We explain about these in later chapters and dive a bit deeper into why sentiment analysis and stance inference is different from each other using experimental studies.

In figure 2.1 we illustrate a real example from our data. In order to detect the stance of a user typically a text component, a target and a user is needed. Here in this case the target in consideration is QAnon, even though the phrase "I am angry that he did not keep the promises" tricks a sentiment analysis framework to think that this is a negative sentiment. User believes that "QAnon does good work" meaning that the users' stance towards QAnon is favorable. This also marks the difference of the sentiment analysis against stance detection.

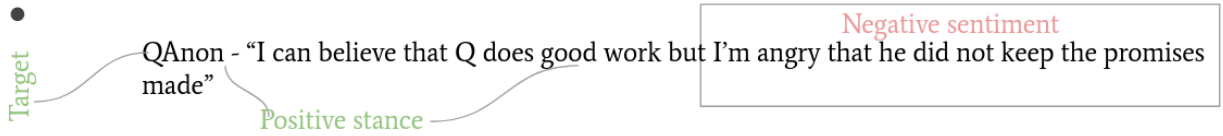


Figure 2.1 Stance detection vs. sentiment analysis

Sentiment analysis can be interpreted as the process of understanding whether persons opinion expressed via text is positive, neutral or negative [65, 71, 76]. The sentiment can be explicit ("it is a beautiful day") or implicit ("The headphones broke in two days") [71]. Typically, the sentiment extraction can be done by observing the text and linguistic features of the given text. Even for a complex problem such as aspect based sentiment analysis, many models uses text features to infer the sentiment [48, 65, 70, 71, 76, 81, 89, 110, 117]. However, in the case of stance detection, we cannot just rely on the linguistic features for a good estimation of the users' stance towards some target. We give our reasons and also what the literature has to offer about this issue in Chapter 5.

We formulate the stance detection problem as follows,

$$S_u(T) := \sigma L \odot \phi M_u \tag{2.1}$$

Here S is the function of the stance of a user u parameterized by σ and ϕ. L is linguistic features in consideration and M is users non-linguistic features and T is the target in consideration. This work is mainly finding the relationship between these items and defining the stance of a user using them.

2.1 Research Questions

Here in this work we are exploring a few research questions and objectives. We shall enumerate them and dive a bit deeper on why we have selected those now.

1. What is a comprehensive public opinion?
2. Collecting a dataset to extract public opinions about COVID19 vaccine.
3. What are weak indications for stances?
4. Finding propagation methods to infer non-participating user stance.
5. Introducing weak supervision through found weak indications.

First we need to explore why we need to quantify a comprehensive public opinion. The current state of the art for measuring the public opinion is based off of traditional natural language processing methods. We have also discovered that the majority of the social networks are typically observers or silent participants. These users will consume the information which has been created by others and align their opinions or form their own opinions. Because of this, we do not see traditional NLP based opinion quantifications extracting a comprehensive public opinion from social networks. Therefore, improvements on these methods are a must.

Then in some cases, users bring stories or news articles from outside the social network and discuss them within the network. We dive into how article based stance inference is better at determining the statement level stance compared with the traditional stance inference.

After that we try to explore how we can improve this version of stance and propagate it in the whole network so that we get the opinions of these silent participants as well.

We understand that annotating stance of the user is a convoluted matter. While you have to consider the users on-topic text contributions as linguistic features. As we explain in later chapters, stance extraction would be accurate if we understand the users social circle or rather who this user associates with and who this user interacts with the most. Knowing the stances of these users will help us to determine the stance of the original user. It seems this is a recursive problem and having experts look at users this way would be a tedious task. However, if we decide to go with unsupervised methods, this will not be an issue but unsupervised methods come with a disadvantage of not having labeled data to learn from.

To solve these issues, we have used weakly supervised methods. In our case the weak supervision comes in the form of annotated hashtags. We understand that hashtags provide meaningful contextual information related to the tweet and therefore a hashtag will be much better at determining the theme of what the user is discussing. We decided that a part of the users stance is in actuality is based on the other users they are following. If we reduce these users down to the hashtags which they are composing the tweets with, it is very easy to find the stances of those hashtags because it is written in words. We use this argument to build our propagation methods on graph datasets.

CHAPTER 3
RESEARCH DATA

Social media is filled with news articles, interesting stories and entertainment news. It is a given that along with most of these legitimate stories and articles some of them may also be misinformation. It is important to research about how users actually react to news articles online. The problem of detecting misinformation and identifying whether an article title misleads readers has been studied. However, work related to user's reaction or stance of a shared article has not been explored much. This chapter we explore how to fill that gap by introducing a public dataset which includes users text contribution, article title, article content and a link to the article.

There are various mechanisms that users can use to share news articles on social media platforms. A typical method is to share a link to the source and add some text covering users' opinions or stances. This stance can be either agreeing, disagreeing, or contradicting the article's contents. While a service can be used to check whether the content shared in a news website is factually correct, we can also use the user's stance on the article to test whether the user is spreading misinformation by using simple logic. If the user agrees with a verified misinformation article or contradicts a legitimate news article, then we can posit that the user is spreading misinformation.

Building upon this, we present a dataset. The proposed dataset includes various news articles and user opinions on the "novel coronavirus vaccine." We choose this domain since there is a lot of information that has already been verified by world and state organizations,

and also it is timely [25, 120]. The following section will cover related work, mainly focusing on the data sections.

To build a stance inference framework we first need data. Since we are looking to implement the framework on top of Twitter we are using the data we have collected from twitter. Most experiments and explorations we have conducted here are from our paper titled "Misinformation in Social Media Platforms and Web Articles: a Dataset to Infer User Stance" [8].

3.1 Existing datasets and related work

[80] discusses a method to detect clickbait titles of articles. Potthast et al. introduces clickbait as titles designed to entice readers into accompanying link. This also is an important direction related to the stance detection problem. Interestingly [80] also focuses on Twitter data and has collected 2992 tweets, including URLs in the 24^{th} week of 2015. They have also made sure to collect tweets from a selected list of publishers such as "BBC," "New York Times," etc. The annotations for these tweets come from 3 assessors. One interesting fact about this annotation procedure is that, unlike the FNC-1 dataset, [80] only considers the text content from the tweet and the image accompanying the tweet if available. But the models they presented have used the contents of the linked articles as a bag-of-features. Potthast et al. shows that their methods produce a 0.67 F-1 score. Potthast has not indicated whether the data is publicly available or not.

First Fake News Challenge[1] (FNC-1) is a dataset that is annotated and publicly available.

[1]http://www.fakenewschallenge.org/

In the train split of this dataset, there are 49972 rows. Annotations include 4 classes - "agree," "disagree," "discuss," and "unrelated". 73% of the train split provided by the organizers belongs to the unrelated class.

[22] presents a stance detection system for the FNC-1. [22] uses an n-gram matching system of a lemmatized input from both the title and the content, which will be fed to a binary classification algorithm into related and unrelated classes. Bourgonje et al. shows that this method allows them to score 89.43% weighted accuracy score in this FNC-1 dataset.

[86] introduces a simple method to detect stance on the FNC-1 dataset. They use a TF-IDF vector representation mechanism of both the headline and the body of the article, along with a cosine similarity between the two vectors as input to a neural network to classify the headlines stance into the four classes.

[101] presents a CNN and LSTM based method to classify the FNC-1 dataset. Here they have employed Natural Language Processing (NLP) techniques such as removing stopwords, stemming and then transformed the text into word embeddings using word2vec. Umer et al. conducted an ablation test with preprocessing, principle component analysis (PCA), and other dimensionality reduction methods, from which they found that the CNN-LSTM model with the PCA has the best performance.

[49] shows a topology based method to identify vaccine related misinformation in twitter. They have collected a dataset of 1.8 million tweets and filtered to only retain vaccine related tweets. [49] presents a codebook of vaccine related misinformation and themes. Again, this dataset has also not been made available.

While there is work reflecting and complementing stance detection, only one dataset has been made publicly available (FNC-1). The dataset predates "novel coronavirus pandemic" since it comes with tweets from 2017 and prior. It is also important to note that the FNC-1 dataset is curated to detect the stance of the article headlines. Not the stance of a user in a social media platform. To fill this gap, we present the Twitter Vaccine Misinformation dataset.

3.2 Twitter dataset

The second research question we try to examine a new formulation of the stance inference problem, in-order to find this we have to first collect appropriate data. This means that we need a replication of a small scale social network. This should be possible if we download data from an actual social network keeping its structure intact.

To this end, using the Twitter Academic API we have managed to process about 47Mil public Tweets (posts in Twitter) during the time period starting from January 1st 2020 to October 2nd 2021. All the posts were from a search query which we have prepared to tackle Coronavirus Vaccine related news stories. We have also included "Anti-Vaccination" related keywords so that we would have an acceptable legitimate news stories along with misinformation related to "Covid Vaccines" and the pandemic. Before getting into an explorative study we shall look into the collection parameters and alterations we had done on top of the data we had collected.

Twitter has an API[2] which allows developers to create bots and applications for Twitter

[2]https://developer.twitter.com/en/docs/twitter-api

Table 3.1 Arguments provided to Twarc

Argument	**Value**
Keywords	vaccine, antivac, vaccinescauseaids, vaccineskill, vaccninescauseautism, learntherisk, stoppoisoningyourkids, researchdontregret, leaveourkidsalone
Start date	01/01/2020
End date	11/01/2021

and researchers alike to access public tweets. There are few layers of access, and we have obtained academic API access, which allowed us to access tweets from the past. This access is very important to us because, as we have indicated earlier, we are looking at "anti-vaccination" sentiment and related misinformation. We posit that with the inception of vaccination for novel coronavirus pandemic, the "anti-vaccination" related tweets may also have had a spike. Therefore, we use the academic API to access tweets from the past.

Just as any API access layer, the academic access layer also has its limitations. The major one being the number of tweets allowed to download within 30 days. This does not come close to the number of tweets posted within a month, but since we are not expecting to access all the tweets, we shall use what is available.

We use a command-line tool called Twarc [37] which allows us to download tweets by specifying keywords along with any other search criteria such as geolocations, usernames, etc. In our case, we are looking at all tweets composed in the English language with some specific hashtags between 2020 January and 2021 October. Following is a list of arguments provided to Twarc.

A user has an option to tag the tweet they compose with a hashtag (or hashtags). If a user uses multiple hashtags, Twarc will store that tweet if one of the hashtags matches

a selected keyword. Twarc will also store a tweet if any word in the text matches a given keyword. However, this increases the noise ratio of the tweets, but we use post-processing to remove many.

One thing to note is that Twarc cannot detect whether a tweet is a retweet. To explain what a retweet is, we need to have a small understanding of how Twitter structures its platform. Twitter is a microblogging platform where users can follow other users, and this creates a social network graph where User A sees content composed and shared by all the other user A is following. Sharing other users' content is called a retweet. These do not get filtered by Twarc because of how the Twitter API is set up. The meta-data associated with a tweet allows us to see whether a downloaded tweet is a retweet, using which we create our own scripts to remove retweets.

We wanted to address the issue when users bring articles outside of twitter to discuss, because of how the user sets up the foundation of the tweet other users who are consuming the tweet could have various set-ups to the conversation. If the original user starts the conversation in a supportive manner to the article; then the conversation could go in one direction and a different direction if it was not supportive. Initial stance of the user matters for the discussion and we expect to detect this.

This again allows us to add another layer of post-processing to remove unnecessary tweets without links to the outside web. There is additional processing to do in this step. One is that Twitter generally shortens any URL added to a tweet. Twitter also provides the URL the user added when composing the tweet in the tweet's meta-data. We first look through

all the tweets to create a subset of tweets with URLs; we need to un-shorten all URLs that the user shortened while composing the tweet. This is a time-consuming task since we have to look at each URL's HTTP header with a GET request. We observe that at least 15% of these URLs have loops, dead pages, or links that do not resolve to a correct URL.

Figure 3.1 shows the list of most common domain names among a random sample of the URLs. As you can see, the users tend to link to Youtube[3] the most. Among the list, there are news websites, search results, etc. Since Youtube links typically do not resolve to articles, we remove those tweets from our dataset.

Next phase of the post-processing is to get the article contents, to this end we draw inspiration from [83] and use *newspaper3k*[4] package to extract the text content from the HTML page. This, again, is also a time-consuming task since it involves a lot of network requests. Once we have the tweets with URLs and the article's direct URL, we can start building the dataset.

At the time of publishing [8] we had managed to collect approximately 7.6 million tweets and after removing retweets we are left with about 2 million tweets. And since then we have improved the size of the dataset.

Table 3.2: Some descriptive statistics of the dataset collected

Variable	Count
Tweets	2079232
Users	810650

[3]https://www.youtube.com/
[4]https://github.com/codelucas/newspaper

Hashtags	79437
Tweets with links	259228

Hashtags are an intrinsic mechanism of Twitter where people can find topics in similar contexts. If a person is tweeting about the coronavirus pandemic they most likely will include the *#covid19* hashtag or a similar hashtag in their tweet. Interesting thing about hashtags is that, users can come up with hashtags however they want to, this allows tweets in a similar context to be grouped together and searched easier. As we mentioned above, we collected the tweets under a specific list of keywords. Looking at the hashtags we can have a small understanding of what the users had been talking under those keywords. To this end we have created a word cloud Figure 3.2.

In figure 3.2 we can see the words *#covid19, #vaccine* with a high magnitude because most of the tweets are tagged with them. We can safely assume that these high magnitude words do not add anything to the conversation other than setting the topic. Therefore we have removed the words from the wordcloud and regenerated it to look at the results, to see a better cleaner wordcloud which shows in Figure 3.3.

Twarc retains most of the Twitters tweet objects meta-data so that we can obtain the link from it. In some cases users tend to use URLs which are shortened using a shortening service, these links do need to be un-shortened. Then we draw inspiration from [83] and use *newspaper*[5] to extract the article from the link.

In Figure 3.1 we show the most common domain names found in the list of unique links before running the un-shortening algorithm. As you can see the top 7 out of 10 link types

[5]https://github.com/codelucas/newspaper

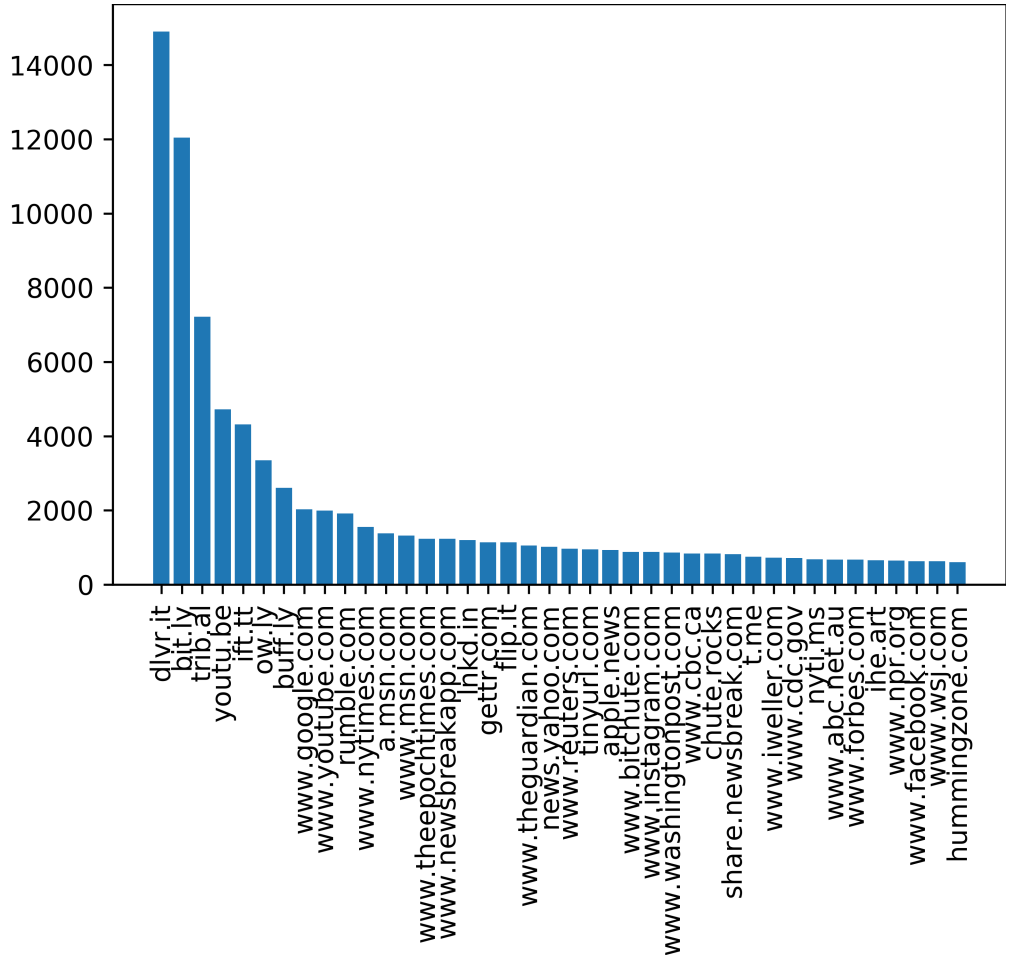


Figure 3.1 Most common domain names found in the links

are shortened. This includes the URLs shortened via Youtube[6]. This makes the case for un-shortening the URLs to look at what type of a domain they are. To this end we create a python script to extract the end URL from the shortened version, this is an important step because typically Twitter it self will shorten all the URLs when the users are composing tweets. This means that in certain cases URLs maybe shortened multiple times and the script needs to be able to unwrap all the layers. We use Python *requests* library to do this and allow it to redirect maximum 3 levels to get to a final URL. We also limit the script by

[6]https://www.youtube.com/

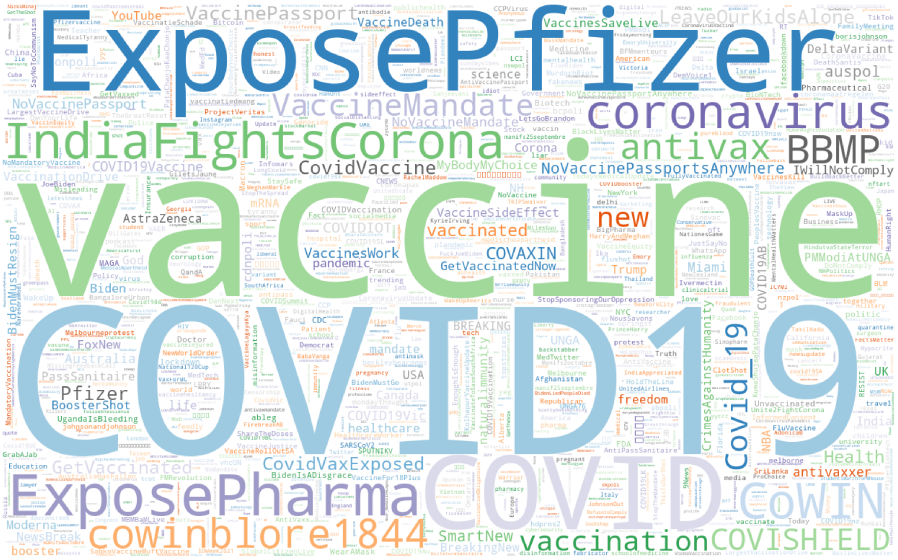


Figure 3.2 A word could generated using the hashtags used in the Twitter dataset

Figure 3.3 Same wordcloud as Figure 3.2 but we have removed some obvious hashtags such as *#covid19, #vaccine* etc.

setting a *timeout* of 3 seconds. All of this will allow the script to get to an end URL without getting stuck (as in some cases) in a loop of URLs.

Furthermore, we can estimate that at least 50,000 of the URLs from 259,228 is shortened.

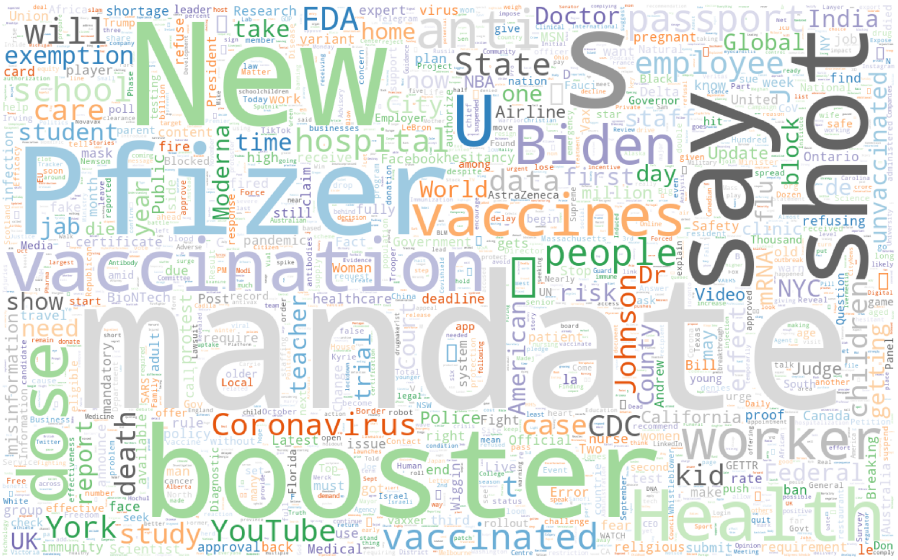


Figure 3.4 Sample of the article titles

Some of these URLS have no content associated with them or they return an HTTP[7] code of 404, meaning that the page was not found. There are about 15% URLs which do not resolve to a correct website. One reason for this could be these tweets are from January 2020 to October 2021, old tweets may have dead links, we exclude all of these from our analysis.

As an initial step, we crawled a sample of 72,000 links with our own spider implementation with the help of *newspaper* package and we present a small analysis below. First we look at the titles of the articles at a glance using a word cloud. At a glance, we can see that the article titles are mostly reporting government mandates, news stories about discoveries, vaccination situation reports etc. We see a stark difference between the hashtags word cloud and this.

Third, we take the dataset as a whole. In the dataset, we have the users' text contribution, a link to the article, the text content, and the title. It is interesting to look at how the articles

[7]Hyper Text Transfer Protocol

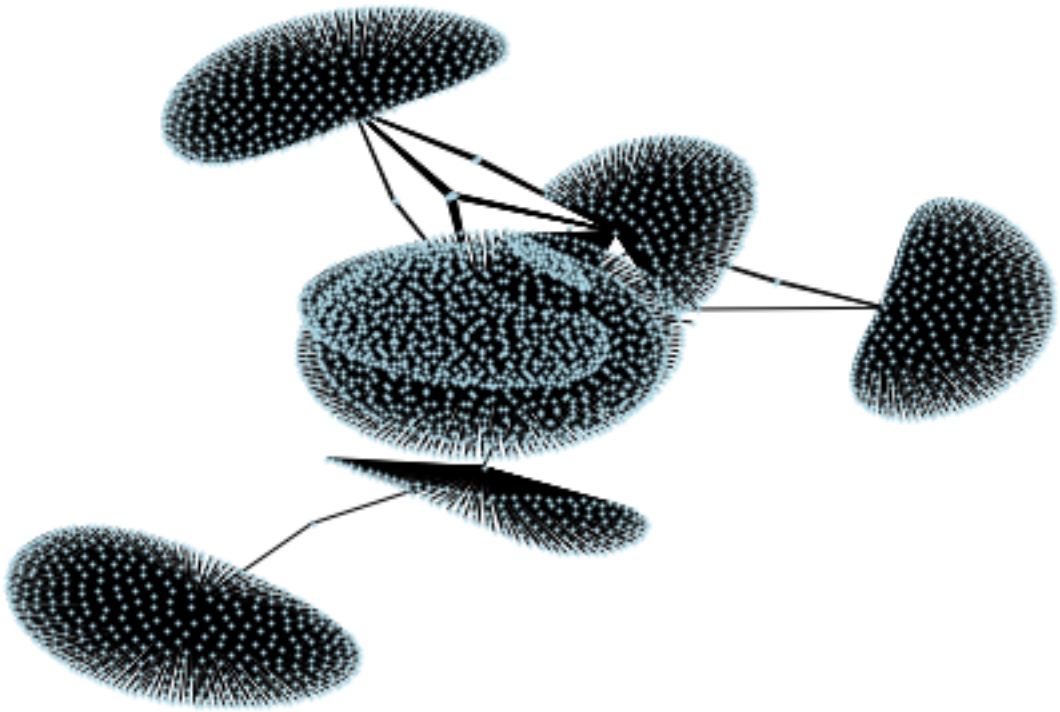


Figure 3.5 Graph structure with selected 10 authors and the links they have cited in tweets

are linked and what articles are closely related in the sense of the social community. To that end, we create a bi-partite graph using authors and URLs as nodes.

Figure 3.5 shows ten selected authors and any URLs they have cited in their tweets. To generate this graph, we have created a node for a source and a target. The selection of just ten users is due to the sheer volume of data and the brevity of the illustration. As you can see, users often share their own URLs, and at certain points, two users may share one article as well. Even though we have made available the number of likes, retweets, and comments a certain tweet received in the shared dataset, we have not incorporated that into this illustration.

We can also create other graph structures using the same dataset. Such as the user-user graph where a user is a node and we add an edge between users if one of them is following the other. Maintaining this network as a undirected graph without edges would make processing on this graph simple. We can also create a hashtag-hashtag graph where a hashtag is a node and then an edge would be added between two hashtags if they occur in the same tweet. Since we wanted a replication of a real world social network we will now check the distribution of these graphs and find out whether they follow the Heavy tail distribution which is typical for a social network.

3.2.0.1 General framework

Here, we focus on the general framework of collecting the data to fit any type of user stance inference. As pointed out in the previous section 3.1, the existing work either focuses on the articles and authors stance using the headlines but in this work, we are interested in the users' stance who is sharing this article on Twitter.

Also, we present all our collected data, the scripts to recreate the files herewith so that other researchers can use the framework to investigate other social issues not related to vaccine misinformation and users' stance. The generality of this framework is that the method does not depend on what type of search query is employed, it will still create a similar *jsonl* file, which can be used to create a dataset of user stance. Following is a small descriptive section about the framework.

We use Twarc [37] and other Python packages such as *pandas, requests* as part of the framework. Initially, as explained in the section, we use the Twitter API to access the tweets

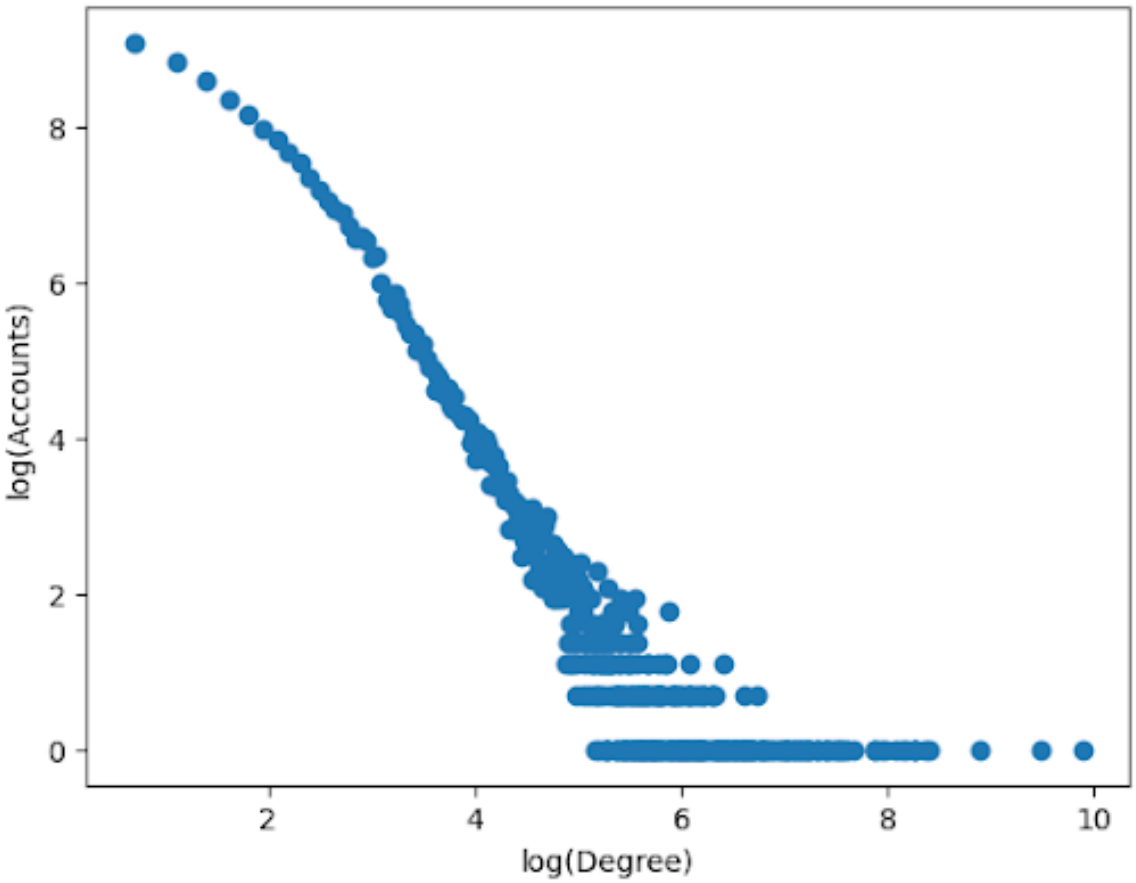


Figure 3.6 The node and degree distribution follows a typical heavy tail distribution found in social networks

via Twarc. Due to how Twarc's interactive mechanism for accessing tweets works, a single line may have multiple tweets. To handle this issue, Twarc itself provides a tool to flatten the file. This flattened file will be used to remove the retweets using the Twitter meta-data. Once we have all the unique tweets, we process them to extract tweets composed with a URL in their body. This will be the final version of the *jsonl* file.

We provide a utility function with the library called *unshorten* which will consume a *jsonl* file and un-shorten any URL provided with it; while doing so, it also extracts the username,

timestamp, likes, retweets, comments, and the author. And using this information, it will create a CSV file.

3.3 Telegram dataset

Telegram provides API access easily and in a much more flexible manner, we use a Python package called *Telethon*[8] to scrape data during 2021 March 1st to 2021 September 1st. Since Telegram does not allow for searching the whole of their groups, we have selected some interest groups and channels. Telegram channel is a broadcasting platform where the administrators will be able to send messages to a large audience. The subscribers of these channels can only consume the posted content of the channel. However, additional features can be given to a channel by the administrator by using bots [1]. Groups however allow members of these groups to post and take part in the conversation [5].

The telegram dataset will be used only in a few experiments which are useful at the last chapter, therefore the introduction to it and relevant information will also be available at that chapter relating to the experiments. We direct the reader to follow the last chapter.

[8]https://github.com/LonamiWebs/Telethon

CHAPTER 4
UNSUPERVISED STANCE INFERENCE BASED ON WEB ARTICLES

Social media is an integral part of our daily lives, allowing users share opinions and content easily. This has been beneficial in many ways, but has also become a source for sharing misinformation and disinformation. Users can take different stances on various news or story articles that are being shared online. Predicting or detecting the stance of a user is a challenging problem in the Natural Language Processing and Inference circles, yet there is little work on the disinformation front. Infering users stance on disinformation is an important during a global pandemic.

In our goal of creating an unsupervised stance inference we have published an article titled "Unsupervised User Stance Detection on Tweets Against Web Articles Using Sentence Transformers" [7]. This chapter includes experiments and results from that work and some extra experiments we have conducted after the publication.

4.1 Introduction

Here we use the twitter dataset introduced in the previous chapter to detect the stance of a Twitter user, based on a tweet and an article that they shared along with the tweet. Our process of collecting and curating the large dataset involves parallel computing methods to improve the timing performance. We present a novel unsupervised method for user stance detection that involves three phases: 1) embedding text from articles and tweets using a sentence transformer, 2) MeanShift clustering the article embeddings to determine themes, and 3) performing cosine similarity to find the semantic similarity between the tweet and

article.

Social media has become both pervasive and ubiquitous, with the latest data reporting that 57.6% of the global population are active social media users, spending an average of almost 2.5 hours/day on different platforms [57]. There are numerous social media networks that provide myriad functions to users, from creating communities for music enthusiasts, providing new opportunities for learning, and enabling an easy way to share ideas. Mainstream social media typically allows users to share content along with their own opinions. This functionality allows users to spread news among their followers instantaneously, which unfortunately has been used as a tool to spread disinformation.

Disinformation has been an issue since the inception of mass media [94]. With social media networks, it has often been used for malicious intentions to alter public opinion. Unverified rumours that are later proven to be false can have harmful impacts on individuals and society at large [123]. Disinformation is much more harmful and nefarious due to the spreader's intent, as compared to misinformation, which is unintentional. This activity has been prevalent in recent years in political discourse and COVID pandemic discussions.

In order to determine whether social media activity is disinformation or misinformation, it is critical to detect the user's stance (intent). This is a challenging problem that builds upon user sentiment analysis, and is starting to see some traction in the research community.

Since social media users publicly state their opinions, we only have the words of their post and possible article or website that may be referenced with the posting to use for the analysis. Therefore, any user stance detection on a particular topic or theme must be determined

linguistically. Knowing the user's stance leads to other various research avenues, such as understanding how news spreads in social media, preempting disinformation campaigns, opinion evaluation and rumour detection [12, 38, 124]. Our work introduces a new research problem and solution, with contributions as follows:

1. We introduce a new version of the user stance detection problem that we refer to as an *article based stance*.

2. Our approach is a completely unsupervised process.

3. We integrate topic modelling into the stance detection pipeline to identify potential targets.

4. We use semantic similarity utilizing sentence transformers to detect the user's stance.

4.2 Related Work

We now present a sampling of related work, starting with literature around the formation of the user stance detection problem and the approach taken for its classification. Initial work shows that the stance detection problem started as a post (tweet) level stance classification problem [74]. Mohammed et al. define the task as automatically detecting if the author is in favor of, against, or neutral towards a proposition or a target. As part of their work, they introduce a standardizing dataset with tweet and target pairs and investigate the problem further to explain how techniques for solving sentiment classification is significantly different from techniques to detect user stance.

Aldayel and Magdy instead show stance detection and sentiment analysis are interrelated [13]. They investigate classifying the level of a user's stance and explain the differences for specific use cases of their problem extension. With this work, they show that the stance representation consists of two main aspects, namely the content and the network structure of the user. Zubiaga et al. explore more classification classes (level), which extends the neutral class to 'querying'and 'commenting'. These additional classes further expands the user stance detection problem [123].

Aldayel and Magdy [12] show that a user level stance could also be found, where most of the prior work [11, 28, 58, 74, 115] focuses on classifying the stance at the tweet level. The dataset utilized for their work has some users that have posted more than one tweet, and show that by using these multiple tweets, it is possible to find a stance for the user instead of a single tweet. To extend to user stance classification, Aldayel and Magdy built models using the topic content, user's network interactions, preferences and connections as features for an Support Vector Machine (SVM) classifier. They show that the SVM model performs well with the interaction network features using their ablation experiments.

Zarella et al. [115] attempt to solve the problem of stance detection using the dataset introduced in [74]. Their work uses an Long Short Term Memory (LSTM) network that consumes an embedded input and transforms that into a 3-dimensional vector that is then consumed by a softmax layer to produce stance classes. The major contributions of this paper are in the pre-training phase, where Word2Phrases [72] are used as input embeddings and the LSTM layer was pre-trained using a distant Hashtag Prediction task. They demonstrate

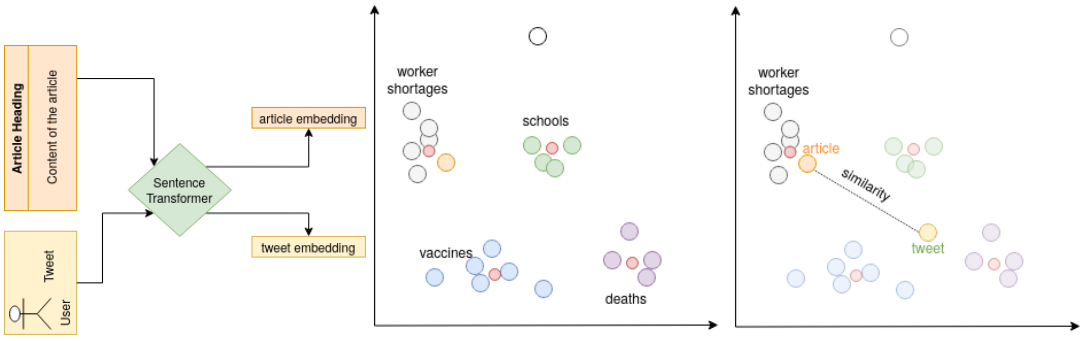


Figure 4.1 (a) Embedding text from articles and tweets using the sentence transformer. (b) MeanShift Clustering the article embeddings to find themes. (c) Cosine Similarity to find the semantic similarity of the tweet against the article

their performance using experiments to be at 67.8% F-1 score.

Chen and Ku [28] show how adding extra linguistic features improves performance of classifiers when using word embeddings as features. The proposed Convolutional Neural Netwoek (CNN) model consumes user information, topic information, along with the word embeddings to determine the user's stance. While Chen and Ku do not conduct experiments in [74] datasets, their comprehensive experiments show that the UTCNN model [28] has better performance against all the compared models.

Kochkina et al. [58] present an important application of stance detection with user's stance in rumours. They use a thread from Twitter to classify tweets into the stance in relation with the original tweet. This work is quite important to us, because it tries to solve a parallel problem to ours. Kochkina et al. tries to solve the stance detection problem using a classification approach with 4 classes of support, deny, query and comment. They use word embeddings of tweets, along with some extra linguistic features, such as the tweet's

role - whether the tweet was a source for a conversation. The proposed model is an LSTM model that consumes the whole thread of the conversation, so that the information in the conversation structure will be retained in the final output.

There is abundant prior work around stance detection as a supervised problem. However, we find the unsupervised domain has only been explored to a small degree and is in an early stage of research [13]. Darwish et al. [29] introduce an unsupervised mechanism for stance detection. They consider tweets on a particular topic and then project the users that are most active on the specified topic into a lower dimensional space. This process brings similar users closer together and spaces dissimilar users farther away. Darwish et al. use complex features, such as user information and community notions due to another study showing that these features typically yield better results as compared to content features [67]. Experimenting with many dimensionality reduction mechanisms, such as t-Stochastic Neighbor Embeddings (t-SNE), Uniform Manifold Approximation and Projection (UMAP) etc., the authors then find natural clusters in the data that lead to unsupervised stance detection. Their work also demonstrates that a better pipeline for stance detection is to use UMAP for dimensionality reduction and MeanShift for clustering. This was determined by experimenting with various subsets of data, along with the purity of clusters in the data.

Most of the work described above assumes that the target is mentioned in the post so that the corresponding stance can be extracted from the post. Augenstein et al. [18] tries to solve a challenging version of this task, where in some cases the target or the theme is not mentioned in the post. To handle this problem, they use a network architecture based

on conditional encoding. The configuration is a two-phase LSTM network in which the first network encodes the target and the second encodes a tweet using the target encoding as the initial state of the network. Their work shows that the conditional encoding mechanism surpases the baseline SVM model [18].

4.3 Motivation and Problem Statement

As we described in the previous section, work in the unsupervised domain of the stance detection is minimal. We aim to add to the challenging task of stance detection by using unlabeled data that we have collected. Prior research work primarily looks at themes that have been pre-selected from a labeled dataset or tries to cluster the user stances under one theme. The problem with this approach is that whenever there is a need to consider a new theme, it becomes necessary to train the models again. One difference with our proposed method is that we collect the user's posted text content along with the article's text content that they are sharing. By including and embedding the article, we can get a rich understanding of the content, and then by further comparing it to the post embedding, we can then determine the user stance.

Table 4.1 Two main datasets used in this work. * this sample is used as a validation set for the methods.

Dataset	**Tweets**	**Users**	**Articles**
Training data	45,000	32,905	26,822
Random Sample* (held-out set)	500	496	457

4.4 Data Sources

We explained in the previous section our desire to infer the user's stance purely using an unsupervised methodology. To accomplish this, there are several options we can consider in order to detect the user stance. Typical datasets [74] assume a theme or topic under which a post is written and then try to infer the stance of the user related to that theme. We name this problem and approach as a *theme based stance*. Solutions using this approach will create stances per user per theme. As we are considering unlabeled datasets, we are solving a different problem. Instead, we try to infer the stance of the user based on one article, and we call this problem and approach as an *article based stance*.

To infer the article based stance, we need to extract the theme with which the user is posting the content under. This is feasible when a user is sharing an article link. To do so, we follow the article link, retrieve the text content of the article, and store this along with the tweet content. Having both the tweet and the article content will allow us to infer the theme or topic the user is sharing their opinion on and then make a better detection of the user stance.

In order to feed the tweets and articles to the model, we perform minimal pre-processing to remove URLs, under the assumption that this will not alter the meaning of the text content. We remove the title of the article from the tweet's content, since this is one of the automatic additions a typical share button on a website places. Next, we find the mean length of articles to find a standard sequence length so that we can define our model and drop any text that is larger than the mean value. This will allow us to perform the pre-processing

in batches and parallelize over multiple processes.

We use a random sample from our dataset to conduct multiple experiments (Table 4.1). This is a held out dataset that the model does not see during the training process. We explain more in Section 4.6, and now describe our data collection process. For the training set data in Table 4.1, there are repeated tweets from the same user and the same article may also be repeatedly referred to multiple times. To avoid repeated users or articles, we ensure that this is not the case in the random sample used for our experiments.

Next, we describe our methods, including our transformer model, to detect a user's stance.

4.5 Methods

Most previous work either uses shallow embeddings or word2vectors [72] to represent text content. However, there have been better vector representations for texts presented previously by using transformers [24, 83, 103]. For faster computation in higher order representations, such as sentences or paragraphs, a Sentence Transformer was introduced by Reimers and Gurevych [84]. For our methodology, we use the Sentence Transformer [84] that produces rich representations using the Bidirectional Encoder Representations from Transformers (BERT [31] as the base transformer. Drawing inspiration from Darwish et al. [29], we use MeanShift as our clustering mechanism, as it works for data that have uneven cluster sizes in a non-flat geometry.

Our model has three phases which are depicted in Figures 4.1a, 4.1b and 4.1c. The

first phase is generating representations (Figure 4.1a) for both articles and tweets using the sentence transformer, which computes rich features for both and embeds the articles and the tweets in the same dimensional space. We need both the article embedding and the tweet embedding to be in the same dimensional space, because later we are comparing the two to detect the user stance. This could only be possible if the two are in the same dimensional space and embedded using the same process. For the second phase, we need to find the themes or targets. To accomplish this, we cluster the articles, as shown in Figure 4.1b. Drawing inspiration from prior work, we select the MeanShift algorithm to cluster our data. There are two benefits to this. Firstly, the number of clusters are not a parameter for MeanShift, because it is a density based clustering method. Secondly, MeanShift is better at finding more clusters in high dimensional spaces as compared with the KMeans approach or related methods [39]. We experimentally show that when we employ clustering after the dimensionality reduction, MeanShift results in significantly less number of clusters. This is not ideal in our case, because we are essentially looking at finding representative and descriptive themes in the data and a higher number of clusters is an indication of that.

For the third phase of our methodology, we see that the cluster centers represent themes that are generated in the word clouds from our experiment (shown in Figure 4.3). Then using cluster centers as themes, we compute the distance of the tweet embedding to the cluster center to represent the difference the text content of both documents (Figure 4.1c). Using the cosine similarity of these two documents, we employ a thresholding mechanism to determine the users' stance.

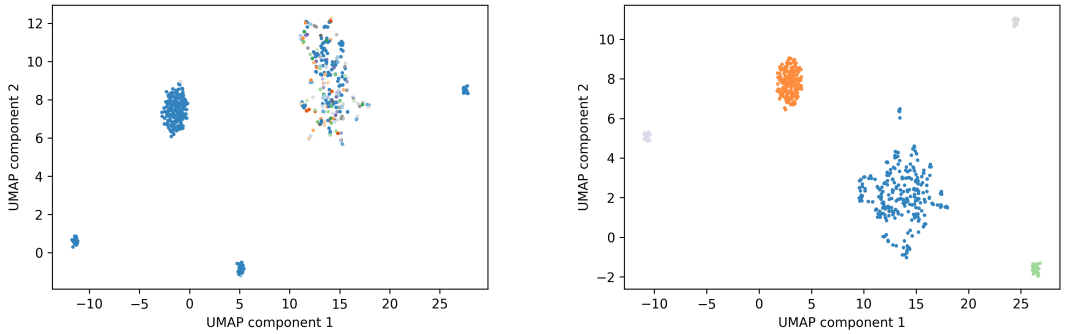


Figure 4.2 Visualization of clusters computed using MeanShift over the UMAP embeddings of the articles. Top: Clustering of the embeddings generated from the sentence transformer. Bottom: Clustering of the UMAP embeddings of the sentence transformer embeddings

4.5.1 Transformer model

The sentence transformer [84] has to use a transformer model as the backbone to produce the embeddings. One can typically use the BERT transformer [31] as the transformer backbone because it is widely believed that it generates rich representations in the embedding. We assume that the pre-trained weights of BERT needs to be refined into our dataset, as it uses specific vocabulary that BERT may have not seen in its training stages. We use the same masked language modeling approach to re-train the RoBERTa model [66], which took about 20 hours on 8 V100 GPUs.

4.6 Experiments

In this section, we describe our set of experiments, including the hardware used for them and then dive into a small discussion section.

4.6.1 Experimental setup

For the experiments that generate embeddings, we have used 8 Nvidia V100 GPUs. Since the transformer that we use as the backend of the sentence transformer has a large amount of parameters, we use distributed data parallel methods to domain adapt it to our articles. All other experiments are done on an Intel(R) Xeon(R) Silver 4110 CPU with 32GBs of RAM. Fine-tuning of the sentence transformer was done using the distributed data parallel methods because of the large number of parameters in the model and also due to the number of sample data.

For each experiment, we take a random sample of 500 tweets and their corresponding article. We compute the embeddings from the same sentence transformer used for the whole dataset. The 500 tweets and articles used in our experiments were not seen in the training phase of the sentence transformer.

As we explained in the previous section, we compute embeddings for articles and tweets. Figure 4.2 shows the output of our clustering process. Since the embeddings we are computing are high dimensional, it will not be possible to plot the output. Therefore, following the approach taken by Darwish et al. [29], we use UMAP to embed the embeddings down to a 2-dimensional space. This allows us to conduct multiple experiments and also verify previously reported results [29].

Our first experiment follows [29] and looks at the number of clusters being created using the MeanShift method. As mentioned earlier, we claim that each cluster corresponds to a theme of discussion in the social sphere. When we employ the clustering mechanism on the

high dimensional embeddings, fresh out of the sentence transformer we observe that there are comparatively large amounts of clusters (top portion of Figure 4.2). For example, in the case of the random sample, we see about 149 clusters. Next, we tested what happens when the clustering mechanism is employed after the UMAP embedding process. Essentially, since UMAP reduces the number of dimensions we see that similar embeddings get closer while dissimilar ones farther away from each other (bottom portion of Figure 4.2). Next we investigate what the clusters entail by conducting a word cloud analysis.

We combined all articles in a cluster and generated a word cloud per cluster, resulting in 149 word clouds. We chose a few interesting plots to report in this article. Figure 4.3 show a sampling of our results. It is apparent that the clusters are meaningful, as we can see the top left shows vaccination issues related to adults and children. Similarly, there are clusters that talk about school enrollment and students, COVID vaccine related deaths, employee shortages and even some sport news has been formed into a cluster.

Our next step was to investigate what happens when the clustering mechanism is employed after the dimensionality reduction with UMAP. We find that this results in a fewer number of clusters. Although this seems to be clean clustering in visualizations, it does not capture interesting themes as we did in the previous experiment. There is one cluster (Figure 4.4) that relates to the COVID pandemic, and this process fails to capture the various themes that occur in the dataset.

Lastly, as we described in our methods section, we compute the similarity between the cluster center and a respective tweet. We established that the clusters denote themes, and

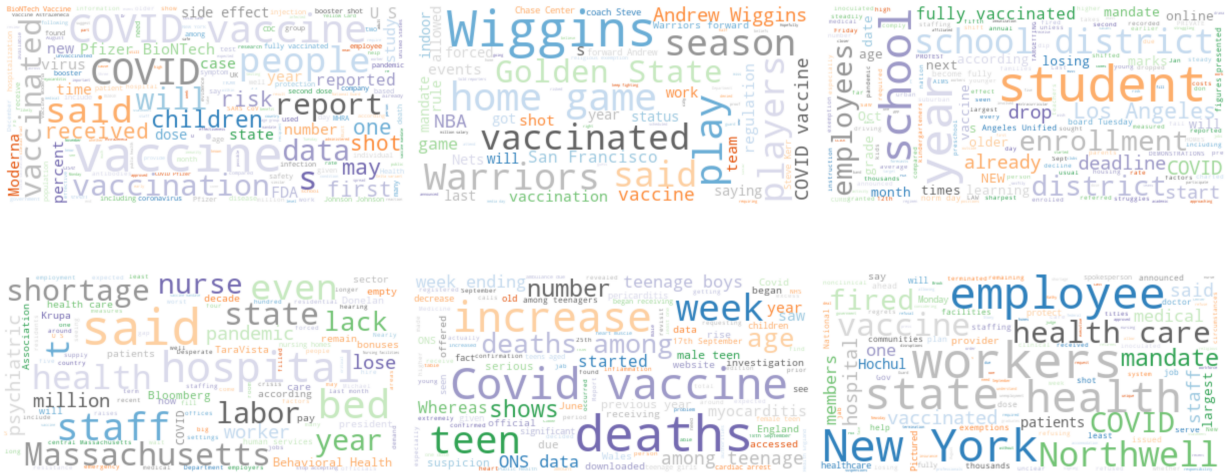


Figure 4.3 Generated word clouds for selected clusters. It is clear that the 6 word clouds relate to 6 different themes. We also observe that some of the clusters have similar content so it is possible to merge them together, but we choose not to do so to maintain the unsupervised mechanism of our method.

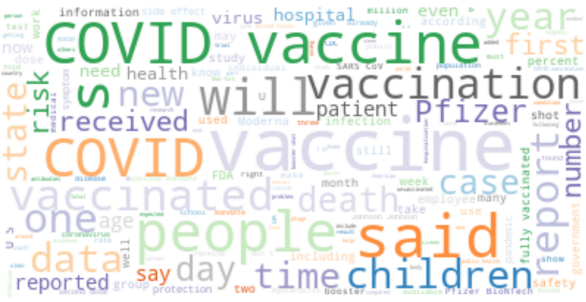


Figure 4.4 Generated word cloud when MeanShift is employed after dimensionality reduction

that the user typically takes a stance on the article that they are sharing alongside the tweet. To compute the user stance, we use the cosine similarity of the embedding of the cluster center and respective embeddings of tweets. Figure 4.5 plots the similarities of the random sample. Both the top and bottom portions of the figure, show the distribution of tweet and article similarity. We observe a mean similarity 0.223 with a standard deviation

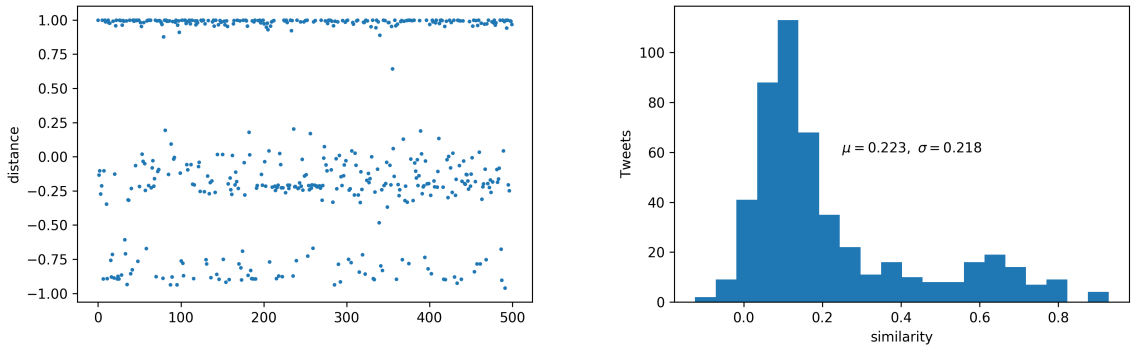


Figure 4.5 Cosine similarity of tweet embeddings against cluster centers for the same sample of 500 articles in Figure 4.2

of 0.218. We can also see that there is no clear division of 3 classes as we expected. To overcome this, we use threshold values to extract the classes. Since the unique mechanism of the methods section where we need a tweet tied with an article to generate the user stances we need similar datasets with labeled data to perform an F-score evaluations and for the clusters or to the stance detection procedure, which is currently not available.

4.7 Results

Upon conducting above experiments, we observe several interesting insights into the dataset and the process of modelling the user stance using the unsupervised mechanism we introduce. In this section, we present a brief discussion of our results.

Table 4.2 We use the Silhouette score to compute the cluster purity

Model Name	**Retrained**	**No. topics**	**Silhouette Score**
paraphrase-MiniLM-L6-v2 [84]	No	149	0.536
all-MiniLM-L12-v2 [2]	No	147	0.543
all-mpnet-base-v2 [4]	No	127	0.529
all-distilroberta-v1 [3]	No	128	0.518
all-distilroberta-v1 [3]	Yes	129	0.518

We first start with the different models we have tested. Each of these are publicly available as pretrained weights. As Table 4.2 shows, we have only retrained one model due to resource limitations. While there are slight differences between these models, they all are general purpose models that are typically used for semantic searching and clustering. We chose these models purely based on the intuition that a semantic search should provide support towards the goal of semantic similarity. We retrain the *distilRoBERTa* model and find that the original model and the retrained model are not that different from the perspective of cluster purity (Silhouette score). We observe that the difference of the silhouette score is infinitesimal, especially considering the number of clusters and data points. The reason for this is the available pretrained model of *distilRoBERTa* has been trained with 1 billion pairs of sentences. In comparison, the main dataset citeabeysingheMisinformationSocialMedia2022 is 260,000 articles. In this study we are using only a partial set of that dataset due to computational constraints which has 26,822 articles. With significantly less content, there may not be enough language to make much difference in the model weights. We also observed that increasing the epoch size by 100% still does not change the purity of the clusters.

The silhouette score is bounded between -1 to 1, where 1 means the clusters are pure and -1 is when there are samples belonging to other clusters in a selected cluster. A score value of 1 could mean that all articles in that cluster are textually similar or there is just one article in the cluster. And a score value of -1 means that all the articles are different on their textual similarity, therefore we look for a large positive silhouette score which is not 1.

The highest score corresponds to the *all-MiniLM* model, which is a general purpose model and a smaller model as compared to the *all-mpnet-base* model.

We also find that the average accuracy of any model in predicting the user stance is roughly 61%, and the maximum accuracy of 66% is recorded by the *all-MiniLM-L12-v2* model.

4.8 Conclusion

In this day and age, social media has tremendously impacted our daily lives. While the impact has been very positive in general, in recent years, we are seeing concerted efforts to spread disinformation across social media. In this context, user stance detection becomes very important. Current research in this area is limited, as it is relatively new. Extra linguistic features have been used in prior work to supplement the low linguistic clues of where a user may stand given a topic or a theme. In this work, we took a new approach to infer user stance in relation to a post (tweet) and an article using purely unsupervised methods and with no extra linguistic features. To accomplish this, we use complex language models, such as the RoBERTa model and the Sentence Transformer to embed the posts and articles into higher dimensional vectors.

Results from this work are encouraging and can be extended in the following directions. Since social networks are inherently communities that can be expressed as graphs, coupling extra-linguistic features into the strong linguistic feature embedding from the sentence transformer, will allow us to use graph mining methods with this data. Furthermore, knowing the

user stances on different social aspects, such as disinformation, will allow researchers to track the spread of disinformation. Another interesting future directive will be to find underlying linkages between communities where these communities are discussing completely different topics but they are linked by other factors such as cited articles and users.

CHAPTER 5
PROPAGATING FOUND STANCES TO SILENT PARTICIPANTS

Measuring public opinion has many interesting applications. It is a known fact that in social media networks, the majority of the users consume information without taking part in the chatter. Quantifying the opinions and stances on day to day happenings with a silent majority may not be possible with just Natural Language Processing tools. In this work we explore the potential possibility of using graph methods to propagate the found opinions of users to silent participants. For this work we use the Twitter chatter during the early pandemic times since back then, there was a lot of polarizing ideas being discussed in social media. We also know there was some misinformation relating to important medical facts and political decisions were also on the table. It also follows that understanding where users stand on certain aspects such as masks and vaccinations can be used in better informing the public about matters such as health and safety. Therefore it is important to capture a comprehensive opinion about these important matters. We report the findings of our experimental study into using Label Propagation to find user stances of silent participants. We use several implementations of Label Propagation Algorithm including some which are derivatives of the traditional method to test our hypothesis. We also present our results which perform with 0.94 with F-1 (macro) against our baselines which are at 0.49 F-1(macro).

5.1 Introduction

Social media is an integral part of our daily lives, allowing users to easily share opinions and content. In any major world event like the COVID-19 pandemic; social media networks

generate millions of posts [40]. Social scientists have been interested in learning about public opinion in social media networks and with the proliferation of social media content, the task of finding public opinion in an automated manner has been a challenging task. Due to the volume of the data and non-homogeneity of the content, it involves multiple steps when deciphering the social opinion on some aspects. There are established methods, such as sentiment analysis [48, 114] which are used to analyze people's feelings using text created by them. There are other versions of this problem, such as aspect-based sentiment analysis [89], where the goal is to extract sentiment from some entity mentioned within the text itself. Other opinion mining methodologies also exist, such as stance detection which sometimes can be different from obtaining a localized opinion via sentiment analysis. The difference in this is the fact that stance detection goes beyond linguistic features and tries to use social network features such as connectivity, neighbors, etc. to measure an opinion. The difference of the stance inference against sentiment analysis has been thoroughly studied in [13].

Stance detection is strongly paired with a target (theme) under which a user's opinion is inferred [7]. A similar case could also be seen in the aspect-based sentiment analysis, where for example the opinion about a 'movie' with regard to its 'plot' can be negative, positive, or neutral. The aspect with regards to the sentiment here is the 'plot'. This does not mean that the same user shares similar sentiments towards other movies with a similar plot. With respect to stance, if a user takes the stance of 'disapproval' (negative or against) towards a certain kind of theme, we can guarantee that their stance is the same with similar themes. Regarding the stance detection problem, for example, a user can be supportive, against,

or neutral about a certain government mandate. While aspect-based sentiment analysis uses only text features to analyze the opinion, stance detection uses both text and network features to analyze the same. This is where stance detection is different from sentiment analysis; it analyzes a much more challenging and comprehensive version of public opinion. Since stance detection looks at the problem via a social graph, it is possible to go beyond the local opinions we extract using the classification or clustering methods. We believe that the social opinion extracted by stance detection is much closer to the real public opinion because it covers the opinion of the silent majority as well.

There are two types of users in social spheres, some share their opinions publicly (or privately to their followers) and others do not. Their participation stops at a 'like' or a 'share' of the original content composed by someone else. We know there are users lurking in public spaces in social media networks without actively contributing to the conversation. It is also worthwhile to note that these silent participants make up most social media networks [78]. This means that using sentiment analysis will miss the majority of users, because it will need text contributions to extract opinions. To this end we introduce our work on inferring the stance of the silent-participants.

The following section will cover a non-exhaustive comparison of the literature regarding extraction of the stance of participating users and will go into depths of how to infer the stance of silent participants.

5.2 Related work

Stance detection is the problem of automatically evaluating whether the user is supporting, against, or neutral towards a target [93]. The targets here can be a person, organization, movement or government policy [74]. It is a problem that has been out there in the natural language understanding circles for some time now ([7, 13, 11, 18, 29, 67, 74, 87]). We understand there are two main approaches for stance detection, 1) statement level and 2) user level stance detection. There are supervised, semi-supervised and unsupervised methods for both stance detection problems; here in this work, we shall look at established methods for stance inference on user-level stances on unsupervised, supervised, and semi-supervised paradigms because it is possible to propagate user-level stance to other users based on their connectivity on a social media network.

5.2.1 Stance detection

Aldayel and Magdy look at a wide variety of features for the stance detection problem in [12]. While considering the text features, they also look at network features such as social connectivity, follower-followee relationships, and user interactions such as mentions of the handles of other users. They use a baseline model to show that it is possible to get comparable results with network feature models versus models that only use text features. This argument of just using network features is quite important for us since we shall also be using the same. When network features are combined with text features, state-of-the-art models were easily outperformed. Aldayel and Magdy bring another claim respective to the

sociolinguistic aspect, whereas human users cannot have a truly neutral stance, and therefore by removing the neutral stance we can improve the performance of the models further.

Darwish et al. introduce an important model for the stance detection problem [29]; which is used by many other works as a baseline or even as a part of the working pipelines (works such as [87]). This is an unsupervised method where most active users are being projected into lower dimensional space. They use features such as retweeted accounts, retweets, and hashtags and embed them in such a way that similar users are closer in proximity. Darwish et al. also note that this method is highly biased towards the dimensionality reduction and clustering methods and shows that the mean shift clustering and the UMAP dimensionality reduction work best.

Using additional topical tweets from the same user to evaluate the stance of a user has not yet been explored. Samih et al. look into this and share their findings [87]. They use methods that were introduced in [29]. The vectors computed using the best setting proposed by Darwish et al. in [29] are then clustered and manually inspected to be labeled as a whole cluster. Using these labels, it is possible to find any user alignment. It is important to note that Samih et al. use embeddings extracted by BERT [31] and fine-tune it using a deep neural network for stance classification.

Slovikovskaya reports their findings in [91] on integrating transformer embeddings to improve document-based stance detection [30]. Here, similar to [35] stance detection, Slovikovskaya repurposed a data set introduced in [43] where a topic was tested against a document. They report slight improvements in the F1 macro when a transformer embedding is used instead

of shallow embeddings.

Fraisier et al. propose a semi-supervised method to classify stances using the proximity to communities [36]. They create a multilayer social graph where each layer represents a different type of proximity, such as content-based, social network-based, or geography-based. Fraisier et al. experimentally show that communities are extremely homogeneous and exploit this fact to extrapolate that all profiles in a community share the same polarity of the stance. They use a semi-supervised model, SCSD where first, a set of known profiles are labeled for their stance and a social network graph is created and according to the proximity communities are computed. Then an iterative stance detection algorithm is performed on the known seed profiles to find the community's stance. They use the majority class in a community to assign the stance within. Fraisier et al. claim that having multiple proximities strengthens the stance alignment. One advantage of this method is that very little labeled data is needed to compute the stance of a large number of users.

5.2.2 Propagation

Label propagation has been used as a semi-supervised method when there are high-density areas of unlabeled data [122]. Zhu et al. claim that the k-Nearest-Neighbor methods are a similar class of supervised methods where both methods assume that closer data points tend to have similar labels. Both approaches work on graph-like data structures, thus pushing labels through unlabeled data, we could obtain the labels for all the data points when the system converges. Zhu et al. mathematically prove the convergence of the algorithm. There are derivatives of this method such as [122] where some aspects of the propagation method

could be different and improvements to the original algorithm such as [105] extending on the Gaussian function which computes the edge weights via an algorithm called Linear Neighborhood Propagation.

Label propagation and its derivatives have been used in many areas of the machine learning domain as a weakly supervised method. In the problem of stance inference, however, it has been used as a community detection method in graphs [21], but not many works use this method. [47] shows that combining label propagation with a simple model gives much better results than a graph neural network. Huang et al. propose "Correct and Smooth" method which is designed for data that do not rely on graph structure and have 100s of times fewer parameters than a graph neural network. This also follows that it has lower training times and needs much less training data.

5.2.3 Hashtags as coarse-grained topics

Wang et al. discuss how an iterative method could be used to propagate the sentiment in a Twitter graph

citewangTopicSentimentAnalysis2011. They also note how a boosting classification method can improve this method where both sentiment information and hashtag co-occurrence is embedded as information for the method. Here self-explained hashtags are maintaining their polarity while all other hashtags will be labeled by the method by the influence of the neighbors.

5.3 Problem Statement

Inferring the stance of a silent participant is a node labeling problem. User stances are typically extracted by utilizing linguistic and extra-linguistic features. Silent participants typically do not have topical linguistic features to use, therefore we exploit graph methods to propagate the found stance. If we introduce the found stance as annotations we can re-introduce the stance inference problem in our dataset as identifying the label of the node based on the influence it receives from surrounding labels of a graph.

In our version of the problem, we identify two unique steps using which unlabeled nodes get predicted with a label. First, seed labels are needed which the label propagation method consumes to predict labels. Second the label propagation algorithm needs to be run iteratively until convergence. Following is an explanation of how the label propagation method is implemented in our case.

$$C_x = f(C_i, C_{i+1}, C_{i+2}, ..., C_{i+k}), i \in N(x) \tag{5.1}$$

Here $f(C_i, C_{i+1}, C_{i+2}, ..., C_{i+k})$ returns the influencing label in the given parameters, and $N(x)$ returns a randomly ordered list of neighbors of the given node x. Traditional label propagation algorithm is not changed in our implementation but what has been changed is how we create the graph, how we compute the influence and how we select labels for the graph. The graph creation mechanism is explained in Section 5.4 For selecting labels we experiment with two methods, 1) informed labeling of hashtags, 2) frequency-based labeling.

We also experiment with two influence quantification methods which can be seamlessly integrated with the label propagation method and the outcomes are discussed in Section 5.6.

5.4 Methods

Data: We have collected a data set during the pandemic where there were many polarizing ideas from Twitter users. In the dataset, we have tweet text content, other meta-data, and user information such as the users' followers. Using the dataset, we have also created a graph of users where the edges represent the follower-followee connection. We have not given edge weights at this time to keep computations simple. The graph has 10Mil nodes and close to 28Mil edges which follows the power-law distribution which we see in typical social networks. We have also created another graph for hashtags where hashtags are nodes and an edge is created when any two hashtags co-occur in the same tweet. Similar to the previous graph, this also does not carry any edge weights and also follows a power-law distribution. We know that by adding two power law distributions the result will be a power law, but with the minimum of the two parameters of the term distributions [111]. So we merge the two graphs and create edges when users have tweeted a specific hashtag, also looking at the node vs. degree histogram we see that the distribution is still power law as expected.

This dataset is not annotated, and due to the volume and due to complexity of the user stance annotation, we decided that labeling will be counter-productive. If it needs to be labeled, we need to take a user to inspect what they have tweeted in the past relating to a specific theme and then also find the users' followers and extract their stances for the

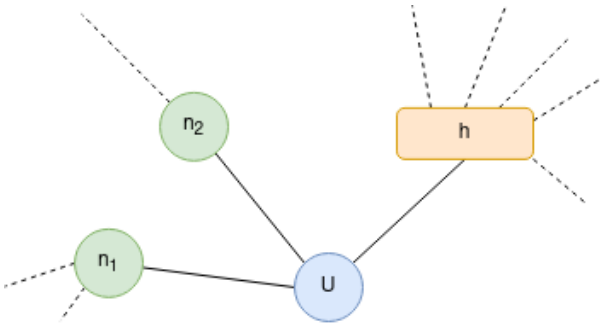


Figure 5.1 User and Hashtags in the same graph. User U shares 3 neighbors out of which two (n_1, n_2) are other users and one (h) is a hashtag.

same problem; only this will properly label a user. However, hashtags are being used as a community-driven mechanism to add additional context to a tweet [107]. Extracting the stance of the hashtag is not so convoluted in comparison with a user stance. These hashtags are then annotated with a stance being supportive, neutral, and against; also including a theme alongside the stance. We hypothesize that hashtags can be an estimator for the user stance on Twitter due to how hashtags are being used. Following a series of computations on the graph, we label selected hashtags. It is also important to note that not all hashtags are labeled; the majority of the hashtags are still unlabeled. One of the contributions of this work is to find the best hashtags to label instead of relying on the frequency of the hashtags.

Now that the hashtags are adjacent to users in the same graph we consider them to be homogeneous in terms of the type of the node. Meaning that a hashtag with a stance could influence its neighboring nodes which may be other hashtags or users. We can test our hypothesis by propagating the stances of hashtags in the graph union and then checking to see what stances they have received. We can evaluate by holding some hashtags which were originally labeled as unlabeled hashtags and computing a performance metric such as

an F-1 score.

Algorithms: The traditional label propagation method considers the structure of the graph partially, looking at available edges where information could propagate. However, the pressure/influence of some information that could propagate through an edge may not be considered. In our case, we experiment with adding control of how much a certain label influence other nodes. There are two possible methods we would like to test this with; using the node degree and the PageRank value as a quantified influence. To elaborate in the case of node degree, the node we are considering has 3 neighbors which have degrees 3, 2, and 6 respectively (Figure 5.1). The labels of these 3 nodes are also to be considered and they are *Against, Against, Supportive*. We can see the majority label of this node's neighbor set is *Against*. But the pressure of the *Supportive* node is much higher than the other labels. Therefore we consider the label to be assigned in this node to be *Supportive*. Algorithm 1 shows the implementation details. Then again in the case of page rank, we initially compute the page rank value matrix and use the page rank values of the neighboring nodes similarly to the previous degree method.

Algorithm 1: Modified Label Propagation (MLP)

```
input  : Graph G(N, E) and |N| > 0
output: Graph with all n ∈ N labeled
while True do
    for n ∈ N do
        nx = G(n).neighbors
        dx = G(n).degrees
        lx = G(n).labels
        n.label = sum(lx * dx)
    if |n ∈ N.labeled| == |N| then
        break
```

Table 5.1 Centrality measurements for the large component graph

Katz Centrality	User	Page Rank Centrality	User
0.0004	jilevin	0.011	jilevin
0.0004	bynsny	0.009	bynsny
0.0003	thejimjams	0.008	CubaSinFrontera

The following section covers our experimental study of this hypothesis and our findings.

In [8] we introduced a dataset which can be used to infer the stances of users using both text contributions and network relations which are available as twitter meta data in the dataset it self. Using the general framework provided with the dataset in [8], we collected more data relating to our study, here we needed to collect likers of certain tweets and followers and followees of users.

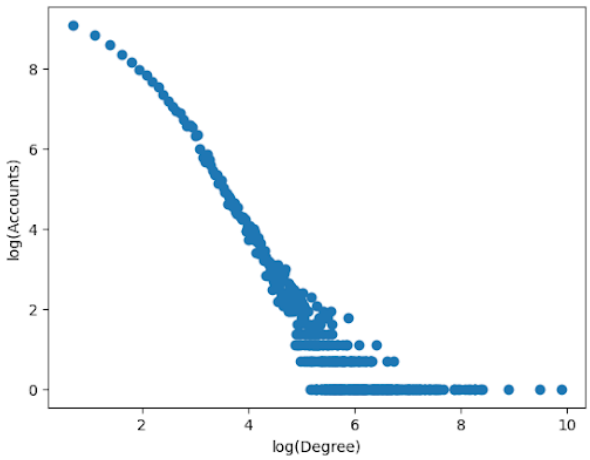


Figure 5.2 This illustrates the distribution of the natural log of node degree (x-axis) and the natural log of users (y-axis) and we verify that it is a Power Law/Heavy Tail distribution which typically can be found in social media networks. Here we have merged the two graphs of users and hashtags in order to visualize the distribution of the merged graph.

5.5 Experiments

As a first step, we have done an exploratory study into the graph of users. We have mentioned in a previous step that the dataset has about 10Mil nodes and 28Mil+ edges. This is a large graph that may not necessarily fit in a typical computer. An adjacency matrix representation of this graph will require a matrix to be stored in memory with a 10Mil X 10Mil shape. As the edges do not represent weights we can store unsigned integers, which still require an unreasonable amount of Gigabytes of memory. Therefore we have looked at other options such as sparse matrix representations which allow us to store the matrix in memory.

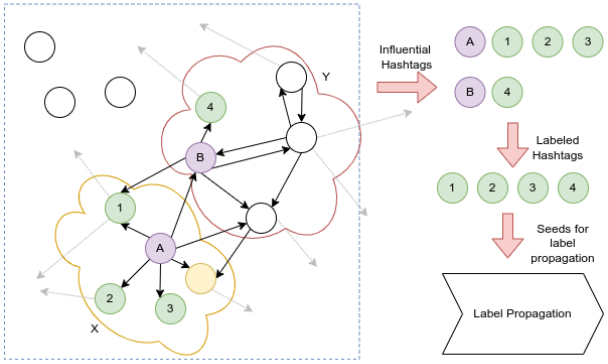


Figure 5.3 Finding hashtags for labels in order to improve the labels propagated through the label propagation algorithm. Here green nodes are considered to be hashtags that are shared by an influential user. Yellow and white nodes are users who are consuming tweets shared by influential users.

Since we need to run graph algorithms on this dataset, we need to convert the adjacency matrix into a graph object with cuGraph[1]. We have experimented with other graph algorithm libraries such as NetowrkX which relies on the CPU alone. We have noted that they do not

[1]https://github.com/rapidsai/cugraph

successfully construct the graph object or just fail to run at all. cuGraph is accelerated on the GPU therefore it runs much faster and is able to hold the large graph in memory as well.

Using cuGraph we then found out that the average out-degree of this directed graph is 2.8 and the maximum degree $6.087060e+05$. We understand that there are some nodes with a large amount of in & out degrees according to both the power-law distribution of nodes and also the minimum and the maximum number of degrees from descriptive statistics of the graph. Then we sought to find if the graph also has weakly connected small components. We assume that the weakly connected and smaller components do not contribute much to the large graph and we remove them from our further analysis.

Centrality measurements on graphs can extract a quantitative measurement for influential nodes within the graph. So we use this to find nodes which in our case users who seem to be important and influential.

5.5.1 Informed Labeling to improve performance

Typical methods for user stance detection will find clusters or communities in the dataset and then label them manually and assign the same label to all users who are within the cluster or community. For good measure, we have done a Louvain community extraction for the large component graph and then ran a similar centrality measurement on the communities as well. This brings out much more diversity in the influential users and we find 41 communities. Superficially it seems like some of these communities are consisting of users who share political stories and some of these communities are consisting of medical personalities who share stories related to the pandemic, vaccines, and how vaccines decreased the spread

of the virus. The community extraction seems to be meaningful in some and not so much in some other communities which we inspected. However, the majority of the communities seem to have some form of meaningful users within them. We use this to extract the 10 influential users using the page rank centrality for each of the communities.

Now that we have 410 influential users spread across 41 communities, we can now use the hashtags these influential users composed their tweets with as coarse-grained contexts for stance classification. We have found 700 unique hashtags among the 410 users which we have labeled as seeds for our label propagation algorithm. We find that this method of labeling hashtags is more informative rather than labeling the hashtags which are frequently used. This is because of our assumption that hashtags and users are both flattened into the same graph where both are taken to be homogeneous. Hence, an influential user's hashtag is similarly influential.

Out of 700 hashtags, 620 hashtags were sent as seed labels for the label propagation method. Rest were kept as ground truth labels to compute the accuracy of the method, in the following section we report the performance of our label selection method against frequency hashtag labeling. Since having a graph where we have merged both hashtags and Twitter users and we also explained how we are considering this to be a homogeneous graph. A hashtag graph is created to inspect the co-occurrence of hashtags, but the hashtag-user graph depicts more complex relationships between the hashtags. We show the results of these predictions in Table 5.6.

Table 5.2 Hashtags and frequency of use. * These hashtags were not labeled via the informed labeling mechanism. This list only illustrates the top 15 hashtags.

Hashtag	**Frequency of Use**
NoVaccinePassportsAnywhere	66149
NoVaccineMandates	65016
Greece*	64062
COVID19	37052
vaccine	29216
ExposePfizer*	14319
Vaccine	13254
ExposePharma*	9543
IndiaFightsCorona*	6796
CoWIN*	6087
cowinblore1844*	5955
BBMP*	5676
COVID	5493
coronavirus	4434
antivax	4173

5.5.2 Labeling users based on their majority hashtag

Previously we considered only hashtags to have labels, But it is also possible to label users based on their hashtag usage. If the user has composed tweets with a larger number of hashtags that belong to a specific class on the hashtag domain we can claim that this user belongs to the same class as the hashtag. This is extending fron the use of hashtags as coarse-grained topics. This method will allow us to propagate the label of the hashtag further in the graph and will receive better labels for more nodes.

5.5.3 Discovering labels for ambiguous hashtags

By using hashtags as coarse-grained topics we may see interesting results at the end of propagating the labels. One such situation is that we receive labels for hashtags that are not in English. We also receive inferences for hashtags that seem to be out of the vaccine or COVID-related domain such as “LeaveOurKidsAlone” and we can see that users typically

Table 5.3 Discovering labels for hashtags

Hashtag	**Label inferred**
PassSanitaire (French)	Supportive
vacciné (French)	Supportive
遥 (Chinese)	Supportive
mRNA ワクチン	Against
LeaveOurKidsAlone	Against
Reds	Against
saferflights	Supportive
VaccineSideEffects	Supportive
GetJabbed	Supportive
UnitedNations	Against
ReligiousExemption	Against
Vax	Supportive
Impfapartheid	Against

use this hashtag to show disagreement about vaccinating children. Some of the selected hashtags which were labeled are depicted in Table 5.3.

5.6 Results & Discussion

In this section, we present the results of the label propagation mechanism to infer the stance of users and other experiments we have conducted. To compare our method, we have done a naive clustering and extrapolated the majority label of the cluster following [29] to get the labels for users and hashtags. This will be the baseline model for our experiments which they will try to beat. Results are obtained by conducting the same experiment five times and averaging the outcomes, this is needed because Label Propagation needs a random start,

Table 5.4 Results - LP: Traditional Label Propagation, FL: Frequency Labeling, IL: Informed Labeling.

Method	**Precision**	**Recall**	**F-1 (macro)**
Louvain Communities: FL + IL	0.45	0.53	0.49
LP: FL	0.90	0.91	0.89
LP: IL	0.91	0.91	0.91
LP: FL + IL	0.94	0.94	0.94
DegreeLP: FL + IL	0.76	0.71	0.68
PageRankLP: FL + IL	0.75	0.70	0.67

and depending on which nodes the random start is set, the outcome may be different.

As we can see in Table 5.4, it is apparent that the proposed label propagation method outperforms naive clustering mechanisms. Even though the clustering is done using the graph structure the issue with the extrapolating method is that the same label is given within the cluster. Extrapolating methods fail when there are no labels in a given cluster. If a label needs to be assigned, then the cluster needs to be manually explored. The other issue is that even if the label of an influential user is in the minority then the extrapolation will discard this information and overwrite the label.

The traditional label propagation method considers the influence of a node, therefore an influential node will not get overwritten rather it will propagate its label to most of the neighbors. We also experiment with two other label propagation mechanisms which we call DegreeLP and PageRankLP. While both of these novel methods fail to outperform the traditional method, they produce some interesting results. When we remove the 'Neutral' stance label following [12] we see that PageRankLP performs much better than when we include this label, it also performs closer to the traditional Label Propagation methods and we believe that in typical settings there will rather be more stance labels than less and therefore it is suitable to stick with the traditional methods.

While the difference between informed labeling and frequency labeling is not largely significant, we believe in large graphs such as ours it will be significant enough. However, to increase the performance, even more, we have combined both frequency labeling and informed labeling. We first reduce the number of influential users we extract in the informed

labeling method to find 600 hashtags and then label them. Then we find the frequency order and if a hashtag is not already labeled from the previous step we add the labels to increase the number of total labels to 700. Table 5.4 show this increased performance in all methods we have tested.

Propagation of the labels into the user-hashtag graph has revealed some interesting results. As we shared in Section 5.4 from the informed labeling method we have found influential users using the computing the centrality. We also clustered the graph and then computed centrality in various communities we found. This follows that we have influential users in several communities. We do not share the labels assigned for the users due to concerns of privacy, you can write to blind@review.com to obtain a full copy of the results. But to evaluate the method and to find the performance of the ablation tests we use the hashtags since we annotated the hashtags in a previous step.

5.7 Conclusion

The goal of this paper was to find the user stance of the silent participants of online chatter. While there are established natural language processing methods to extract the user stance, it is challenging to find the opinion of the silent participants using natural language processing methods. Therefore, we explore what graph mining methods has to offer for this problem and use Label Propagation to show that this method is a feasible and viable option. Our promising results show that it is indeed the case. In our experiments, we noted that in the graph where 73% of users had less than 100 tweets, and none of them were related to

the topics (vaccine, masks) we are interested in. We claim that 73% of users are the silent majority and then we extract the opinions of all of these users. We also introduce a novel way to improve performance by labeling the correct hashtags. Using this we gained improved performance in the traditional label propagation method and other methods which we have experimented with. We also introduce two modified label propagation methods called DegreeLP and PageRankLP which do not produce promising results but have interesting edge cases which we can explore further. Future directives for this work lie in the domain where we can apply the label propagation method for all the nodes in the larger graph using GPU acceleration within the algorithm itself.

CHAPTER 6
DISENTANGLEMENT

Most online message threads inherently will be cluttered and learning to de-clutter or disentangle the threads create new avenues for analysing them. The need for disentangling the clutter is much higher in platforms, such as Telegram, wherein the discussion is taking place does not provide functions to retrieve reply relations of the messages. This introduces an interesting problem to which [106] phrases referred to as a structural learning problem. Platforms like Telegram allow users to join groups of other users. Besides the chronological order of the posts, there is no other structure to the conversation in Telegram. In this section, we propose a deep learning architecture to discover the discourse structure given the minimal structure provided by such platforms.

6.1 Introduction

Identifying whether a certain post from a group creates traction and brings more users to discuss the topic is challenging, thereby making predictions within a discourses difficult. There are many reasons for a user to engage in a discussion: novelty, previous replies, time of the posting etc. [16]. A discourse usually ends due to reduced engagement and a new discourse is formed by a new message posted into this thread which indicates a different theme or topic. In addition, a discourse is not always chronologically continuous. In a discussion, some messages are replies for previous messages while some others are unrelated and form schisms [34, 106]. Even if platforms like Telegram provide reply-to functionality, not all replies are linked with reply-to tags. Reply-to tags are also not indicative of the

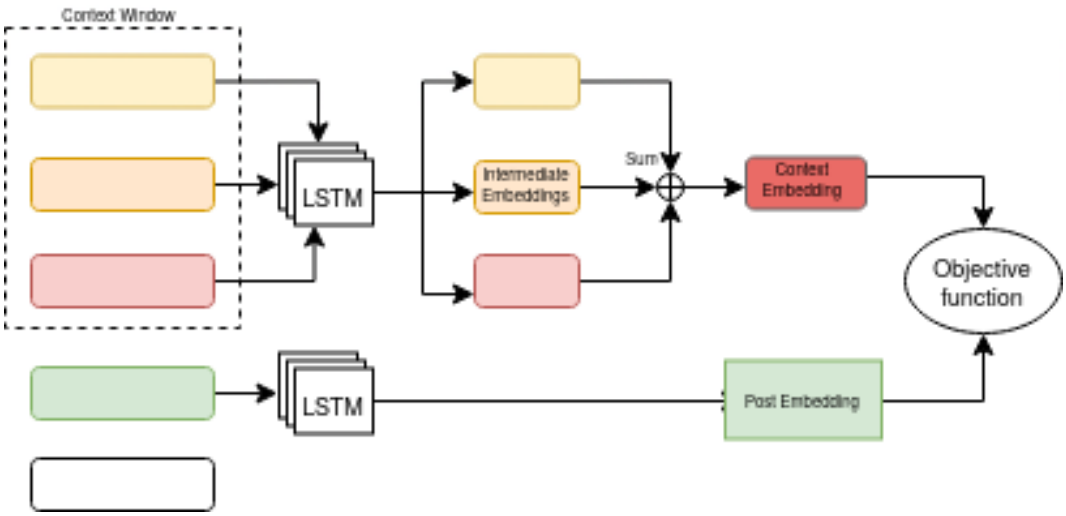


Figure 6.1 Forward pass of the LSTM architecture

entire conversation even when used correctly, because platforms like Telegram only link a single message to the reply, instead of the whole conversation. Considering these constraints, reconstructing the reply structure is an interesting and relevant problem that needs to be explored. This problem was phrased as a structural learning problem in natural language processing domain [106].

There are many applications when a correct reply structure is necessary: to extract correct topics from each discussion, to recommend other threads to users and perform correct question-answer mining are some of them. The one which is important to us in this proposal is related to the 2nd research question.

6.2 Previous work

Among earliest work, [27] discusses a similar structural learning problem, i.e., an automatic detection of signature blocks and reply lines in plain-text email messages. In this work,

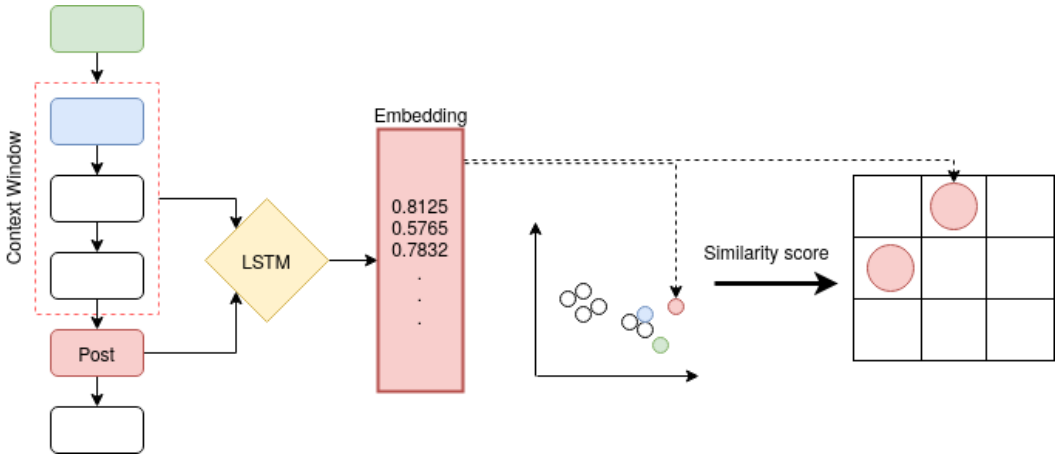


Figure 6.2 Processing of posts with the context window paradigm 1 and generating the similarity matrix

Carvalho and Cohen use machine learning methods such as SVM, CRF, and Perceptrons. Using linguistically inspired handcrafted features they obtain 99% and 98% accuracy. This article laid groundwork for most of the later work: [27] is for signature detection and reply lines detection for emails but our task is slightly different from this. Our definition of the problem as a structural learning problem inspired us to look at [34] which provides much needed perspective. Elsner and Charniak in [34] define an 'utterance' as a part of a single conversation wherein the conversation many participants are reacting to one another. [34] is attacking the problem with a model which is two fold. One binary classifier and the other a partition extractor. Binary classifier determines each pair of messages as alike or different, using a maximum entropy classifier. Elsner and Charniak uses handcrafted features to obtain the classification and the classifiers performance is measured by 'F-1 score'. Time, user handles, mentions, message specific features -to whether the message uses any technical jargon- were among many other features used. This comprehensive feature vector allows

authors to form reply relations from a very simple model. For partitioning, they have used a form of correlation clustering [20]. Elsner and Charniak use experiments to show that their two fold model has acceptable F-1 scores in contemporary work.

Wang et al. in [106] defines the problem much similar to ours. Using a modified conditional random fields (ThreadCRF) implementation, they try to retain the linear features in the discussion. Wang et al. looks at a discussion thread where the messages are uniquely short, hence they do not rely purely on the content similarity. Wang et al. claim it is needed to encode structural dependency and linguistic features among others to obtain good reply relations. For ThreadCRF they feed two different kinds of features, namely Node and Edge features. They define the loss as a log likelihood function and optimize it with L-BFGS [121]. Wang et al. show that different ways to predict the tree like reply relation structures can trick naive accuracy type of metrics, hence they have also introduced novel metrics which seem to capture the parent label relation of the tree structure more accurately.

$$p(Y_n|X_n) \propto exp(\sum_{k=1}^{K} \lambda_k f_k(Y_n, X_n)) \tag{6.1}$$

$\{f_k(Y_n, X_n)\}_{k=1}^{k}$ is a set of features of X_n and Y_n $\{\lambda_K\}_{k=1}^{k}$ are the weights

Given $\Theta = \{\lambda_K\}_{k=1}^{k}$, task for ThreadCRF can be a Maximum a Posteriori inference problem: Each given thread X, authors aim to find the otimal replying structure Y', such that,

$$Y' = argmax_{y \in \varphi} p(Y|X, \Theta) \tag{6.2}$$

φ is the set of all the possible replying structures for X

It is important to consider situations where there is no or missing meta data such as the author, user mentions etc. In [108] authors phrase this problem as a thread structure recovery. Main focus of this work is to identify the explicit parent-child relationships between contributions. Wang et al. were drawing inspiration from [90] and improving on their work on clustering based algorithm to a graph based connectivity-matrix. Just like most other literature, this is also a two fold mechanism. Using a shallow similarity measure such as the TFIDF [54], authors create a connectivity-matrix. Then they treat this as a graph and penalize similarity by distance, time windows, etc. In [108] authors use redefined 'F-1' scores similar to [106]. Wang et al. are using three notable assumptions in their work which extends to many other work in the same domain as well. Aumayr et al. work is very similar in most aspects to [108] but [19] uses features such as reply distance, having quoted components from a different post and thread length in his work. Also their choice for the learning component is an Support Vector Machine [88].

Jiang et al. in [51] presents a model which aligns to our work in a similar tangent with a model with which it defines message pair similarities and uses a ranking algorithm to disentangle messages. Jiang et al. shows the performance of the model using Mean Average Precision with several datasets. [99] takes a different approach but with a very similar foundations to all the previous work. Tan et al. uses an LSTM network with attention to obtain a context vector which then will be used in a classification. Tan et al. shows promising results with this method evaluated with F1 score.

Event detection is quite important in our work, in most cases online discussion boards a discussion may stop and another will be born, also a discussion could divide into several other discussions (schisms). While there is no straightforward mechanisms to detect schisms happening in real time, event detection comes quite close to it. [17] discusses about the problem of short length message Twitter threads and the problem of detecting events while considering the amount of spelling and grammatical errors, and authors outlines a large sequence of work in their article under different classifications of detection tasks, detection method and event types. Atefeh and Khreich also have collected a list of commonly used features which serves into the important problem of event detection.

6.3 Problem Statement

In this article a 'thread' is a linear, chronologically ordered conversation where two or more users take part in a discussion. Problem in simpler terms can be phrased like this; Given a thread P_i compute the reply relations R. $P_i = \{p_{i,1}, p_{i,2}, ..., p_{i,k}\}$, We need to extract post and replies relation $p_{i_j} \rightarrow p_{i,k}$ for all the posts in the thread. A post $p_{i,l}$ could have multiple replies,$p_{i,l} \rightarrow \{p_{i,m}, p_{i,n}...\}$

Inspired by the Skipgram model in [72] we model our inputs and labels as a singular post p_i in a thread and the context window of k posts $p_{i-k}, p_{i-k-1}, ..., p_i$ respectively. The context will allow the network to encode positional information with respect to its linguistic features into the embeddings so that not only linguistic features are captured. In a typical conversation context is usually previous questions, answers and comments on a similar theme.

If the reply relation is not extracted during the time the conversation is happening (not simultaneous) it is also possible to include posts which come after p_i into the context such as, $p_{i-k}, p_{i-k-1}, ..., p_i, p_{i+1}, p_{i+2}, ..., p_{i+k}$. This gives the network a unique perspective of the conversation which in an online setting would be difficult to incorporate.

Schisms in a conversation can happen due to various reasons [33], while we do not observe all reasons we base that a schism can happen either due to a conversation dying out because users didn't engage in it, or there was a more interesting point or a question raised by a participant. We believe if we can detect schisms, we can efficiently model the post to parent relationship by correctly clustering messages.

To handle first type of schisms we can use the language similarity measure, since the vector embeddings itself incorporates the language with the positional information we can use the same here. The other can be modeled as a Hawkes process. In this work we implement the self exciting Hawkes processes in a naive manner, using a Laplace transformation.

$$L_i = \frac{t_i}{\mu} e^{\frac{t_i}{\mu}} \tag{6.3}$$

$\lambda(i) = K + \sum L_i$ is the typical self excited Hawkes process' intensity function, in this work we do not define the background intensity K, making our intensities to be equal to lambda transformation of the time distribution.

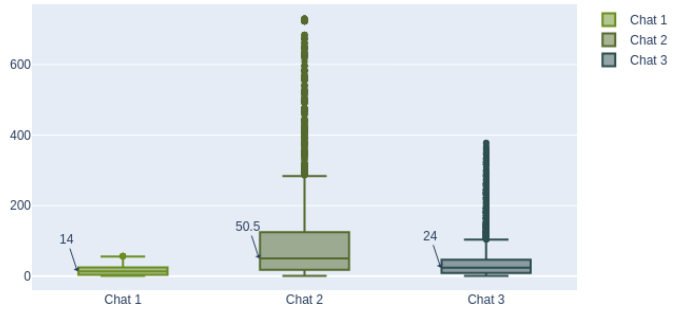


Figure 6.3 Distribution of length of the messages in each chat

6.4 Data

Telegram has been very successful in the recent years [77] as an instant messaging service (IM). While users can send text messages, photos, videos etc. to other users the features that stands out in Telegram are its Channels, Groups and the anonymity given to the users. We have crawled some specific channels in Telegram. Due to the nature of the conversations happening in these channels some users have been banned or deleted meaning that in some cases the crawler does not have access to the author of the message even though the message content exists in the thread. Moreover, when predicting the reply relations of a threaded structure the author of a message is quite important. We have taken the challenge of not using any metadata except for the time the post was created at.

There is in interesting distribution of the message length in words. While most of the messages in each thread are short occasionally there are messages which has hundreds of words as well. These occasional messages are most of the time advertisements which seems to be selling or willing to buy things from users in the thread. Interestingly our encoded post embeddings are sensitive for the length of the message therefore it is easy to see these long

messages as they seem to be outliers. For further reading we direct the readers to Section 6.6.1.

We obtained three discussion threads and from here onward we shall call them 'Chat 1', 'Chat 2' and 'Chat 3'. Each chat has messages, a unique identifier for any given message and a timestamp associated with it. Table 6.1 is a descriptive table of how many messages and how long the discussion spans in each of these discussion threads. We have observed that there are short and long conversations present in the message threads, but conversations are not labeled.

While there are labeled datasets made available by works such as Kummerfeld et al. in [59], the data in [59] has time information only up to the hour and minute. This is not ideal for the Hawkes process to use because of how frequent the users have posted messages to the thread. Meaning that in some instances there are 10-30 messages posted within a minute, Hawkes process needs exact timestamps to correctly determine ranges of schisms. It is also important to note that, to evaluate our methods we use a self collected Twitter conversations. We've collected a conversation from the account *@POTUS* account with about 8000 messages. Since Twitter maintains the reply structure in tweets we used Twarc [37] to download tweets from the said account. Even though we organize these tweets in a chronological order, since it is not how Twitter maintains their *Timeline*, even at a much later time a user can come in and read the original tweet due to the community like mechanisms of the platform. we understand our methods will have an issue relating the latter replies to the early messages.

We have also observed first type of schisms occur in the conversation where it is quite apparent in the Figure 6.6

6.5 Model

We create an LSTM [46] model with a goal to learn vector representations to the posts in regards of their position of the thread and the language in the post it self. As discussed in the previous section to encode positional information we use the context window of surrounding posts. A forward pass of the network is illustrated in the Figure 6.1. We created two paradigms for the context window, first has access to both messages which came before and after of the considering message. This allows the network to grasp positional information of the message from both sides, but this may not be the ideal in a real world setting. Claiming that a reply to a message will depend on the messages came before it, not the ones after. The second is when you take messages which came before a particular post and only them. We test both paradigms and evaluate results in the Section 6.6.

In order to factor in language features we have used an embedding layer prior to the LSTM layer, the vector embeddings will be learnt simultaneously to the post vectors. This elevates the model to learn its own language features depending on the training input, rather than depending on pre-trained vector representations this will make the model to be biased

Platform	**Thread**	**Number of messages**	**Time span (m)**
Telegram	Chat 1	4832	1650.797
	Chat 2	3685	2415.881
	Chat 3	5597	2160.231
Twitter		6434	2820.138

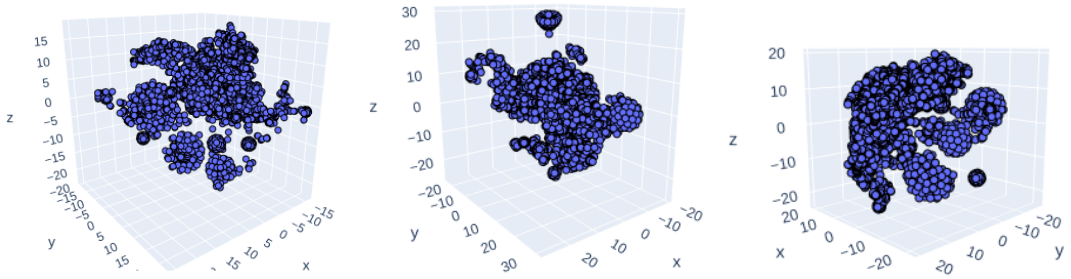
Table 6.1 Dataset description

towards the language used in the discussion boards in Telegram. This is important because in preliminary inspection in the data set, authors noticed that some words are quite unique (gc, pm, dm, atm, acct, smh) to the chat it self. To this end we use an embedding layer prior to the recurrent layer. Embedding layer requires a unique word list as its input, in order to create the word list we remove emojis, user handles and URLs. We have also removed long series of underscores, periods, tilde and equal symbols which indicate underlines and partitions in the posts. These special characters are used mostly to divide a post into parts. This will help us to create an accurate unique word list, the indices of this word list will be used to create the input for the embedding layer. Following is a simple representation of the model and its objective function.

$$l_i, W_i = \phi(p_i, C_i) \tag{6.4}$$

$$loss = \frac{\sum_{i=1}^{n} l_i W_i}{\sqrt{\sum_{i=1}^{n} l_i^2}\sqrt{\sum_{i=1}^{n} W_i^2}} \tag{6.5}$$

Here p_i, C_i are post content and its context respectively. The model will process both these values and return a vector for the post and a vector for its context called l_i, W_i respectively (Figure 6.1) For simplicity's sake we have reduced the model to a function named ϕ but in reality this is achieved through a series of functions and gates, we refer the reader to [46] for further reading of LSTMs gate system. In order to be processed by the LSTM network C_i needs to be some form of an embedding as well, to extract the post embeddings



(a) (b) (c)

Figure 6.4 Figure 6.4a shows a first paradigm context window while Figures 6.4b and 6.4c shows the second type, note that the original figure was rotated so that the clusters are evident in the illustration. In each illustration you can see dimensions x, y & z which are calculated with the t-Stochastic Neighbor Embeddings [102] rotated in a way so that we can see the clusters.

of the context window we utilize the same LSTM model and will iteratively produce an intermediate embedding for all the posts in the context window and concatenate and sum them in-order to produce the context embedding. To learn the efficacy of l_i we employ a cosine similarity function, since both l_i & W_i are vectors in the same dimension it is a matter of distance between the two vectors.

After learning l_i, it is needed to encode the time and the self excited Hawkes process to these vectors. This is required because a vanilla LSTM will not consider the time difference between two posts while it takes each post occurrence as events which happened in constant time. This is where the Hawkes process is important, it factors in where time difference is needed. In this work, the time encoding is a separate process similar to how [108] break down the process into two parts; We also handle the problem of extracting reply relations in two phases. First learning embeddings, then comes computing similarity matrix and processing the graph.

6.6 Experiments

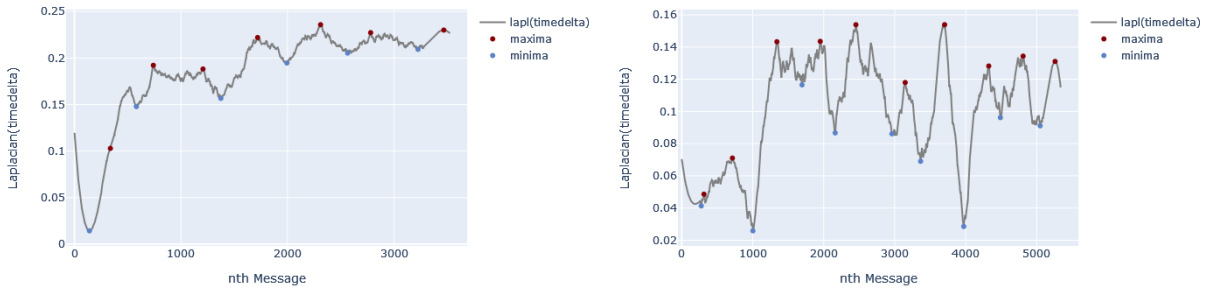
Complete process of training the post embeddings to recreating the reply relations will be discussed here onward. We also dive into the Naive Hawkes implementation and other methodologies from [108] and [19] that discusses the same problem but with different approaches.

6.6.1 Computing Post Embeddings

First we shall discuss about the quality of the vector embeddings of the posts learned by our model. Linguistic and positional features were expected to learn as we trained these embeddings. To get a better understanding about our embeddings we compute t-Stochastic Neighbor Estimation and get 3-dimensional vectors for each of the embedding. We can use these vectors to visualize our embeddings and their quality. Using this it is trivial to examine the quality of the embeddings.

It is apparent in the figure that obvious clusters exists in which when we explore more into them they are quite similar in language and the length of the post content. We used Agglomerative clustering and clustered into different number of clusters, and came into the above conclusion. It was also needed to change the size of the context window to see how that parameter will affect in the quality of the computed embeddings.

Then in-order to employ the naive Hawkes process, a time distribution of the posts to the thread is needed. Using the laplace transformation on top of the time distribution is trivial, the resultant curve is noisy. Hence a smoothing operation is required to find the peaks and



(a) (b)

Figure 6.5 This illustration shows the naive Hawkes process of Chats 2 and 3

bases of the curve. To this end, we used [82]. Kernel size is a hyper-parameter in this and using a few experiments we found out that 501 fits to all our datasets, in lap and lap we show kernel 103 being used to smooth and peaks found.

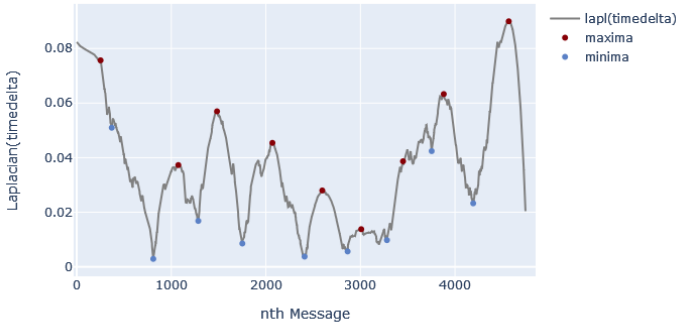


Figure 6.6 This is a zoomed in view of the Hawkes process of Chat 1, close inspection shows that right after the first peaks the algorithm has caught 2 bases which is not the ideal case. We have kept the parameters in a reasonable range where it stops getting too smooth and too coarse.

In figure **??** closer to a peak, it shows that the next message posted into the thread will most likely be a reply to the topic being discussed. Two peaks are most likely replies to

different discussion, two peaks close by to each other means that there was a schism and a lot of engagement by the users in those conversations. And bases mean that the conversation died and a new topic is being discussed.

6.6.2 Algorithms

First drawing inspiration from [108] we create a similarity matrix for the post embeddings in order to create a graph with edges as the cosine similarity. This will create a dense graph which then we are looking to prune. Using the Hawkes intensities calculated in an earlier stage we define ranges a discussion could have lasted in terms of message indices, typically the two bases surrounding a peak. All messages outside the said range will have the edges pruned. We calculate the average similarity score and then prune any edge with less than average score. This increases the number of conversations in the thread and allows the algorithm to find small conversations which happen inside bigger conversations.

This algorithm will yield the pruned graph structure. It is satisfactory to assume that a post with an earlier time stamp will never be a child to a post which came after that. The resulting graph is a directed and sparse structure which represents the thread. Post

Algorithm 2: Pruning - PRN

input : ranges and embeddings
output: Pruned Graph
similarityMatrix = `cosineSimilarity` (embeddings)
avg = `averageScore` (similarityMatrix)
similarityMatrix [similarityMatrix < avg] = 0
pruned = `zeros` (similarityMatrix.shape)
for *(min, max)* `in` ranges **do**
  pruned [min:max, min:max] = similarityMatrix [min:max, min:max]

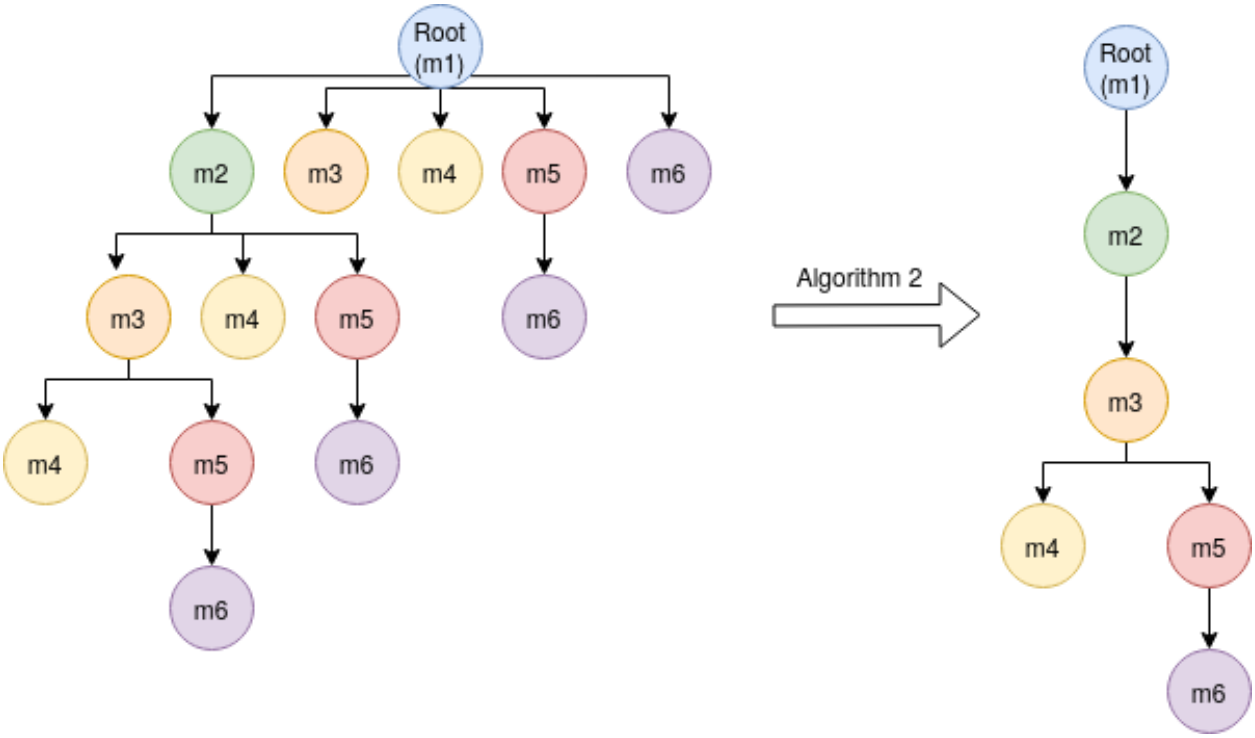


Figure 6.7 Algorithm 2 input and output

processing of the graph will give us multiple discussion threads with one root message for each thread. This first run of the pruning algorithm will help to determine the root messages of the threads, however this tree like graph is has isolated sub graphs and each of these are shallow having maximum depth one, we can use another algorithm to compute the path of the message structure to reconstruct the threads parent post structure essentially creating a sparse pruned representation of the same graph.

Before the second stage of the pruning algorithm, the graph indicates edges from all the children from a parent. This means that a parent can have edges to its grand children. Goal of Algorithm 2 is to increase the depth of the isolated sub graphs, essentially this will prune the repetitive nodes making it a sensible discussion structure.

Algorithm 3: Thinning - THN

```
input  : Graph pruned (pruned) with Algorithm 1
output: Graph after removing extra linkages
for node in graph do
    visited.add (node)
    children = graph.children (node)
    pruned.setchildren (children)
    for itm in visited do
        if node == itm then
            continue
        for child in pruned.children (node) do
            if child in pruned.children (itm) then
                pruned.children (itm).remove (child)
```

6.6.3 Evaluation

Our methodology extends previous work and here we compare our work with [19, 106, 108] using a self collected dataset from Twitter. Also for our experiments we select the context paradigm where messages earlier and after for the considered post is available for the network to process.

6.6.3.1 ThreadCRF

We are recreating the feature space according to our own data set and some empirical knowledge coming from [106]. The natural language processing calls, `_idf()`, `tfidf()`, `bow()` were implemented and integrated to a processing pipeline. It provides a set of features to formulate the feature space which can be categorized into node based and edge based features. Most of the edge based features are using meta information which we do not have in our data set. `hasbar()`, `hasurl()`, `position()`, `sim_with_previous()` were some of the

features included in our implementation. The Conditional random field implementation and the optimizer which we are using in our implementation are parts of `scipy`[1] and `sklearn`[2]. Wang et al. have modified typical CRF so that they could incorporate the edge features. Since we do not have the meta data for the edge based features we are using the bare implementation of CRF. We direct readers towards [106] for further reading.

Just like most CRFs, ThreadCRF implementation also has its own disadvantages in this particular problem. One of which is the computational complexity at the training stage. Also in our use case, when you take smaller dependencies of linear features any CRF will be limited in predicting the structure of the thread because its outputs will also be local to that small dependency window. Mainly due to the missing meta data ThreadCRF fails to perform up to its task in our data set, most of the conversations it produces are flat trees meaning that all messages are children of root. Comparing ThreadCRF methodology against our data set puts [106] in a huge disadvantageous position because ThreadCRF expects more meta data than we have. Hence, we saw the need to test our method against [19, 108].

6.6.3.2 TF-IDF+

Both [19, 108] uses the shallow embedding for posts and either SVMs or just simple similarity matrices to track the reply relation. We shall explore both methodologies in our own implementations and libraries [109] with the self collected Twitter dataset.

[1] https://www.scipy.org/
[2] https://sklearn.org/

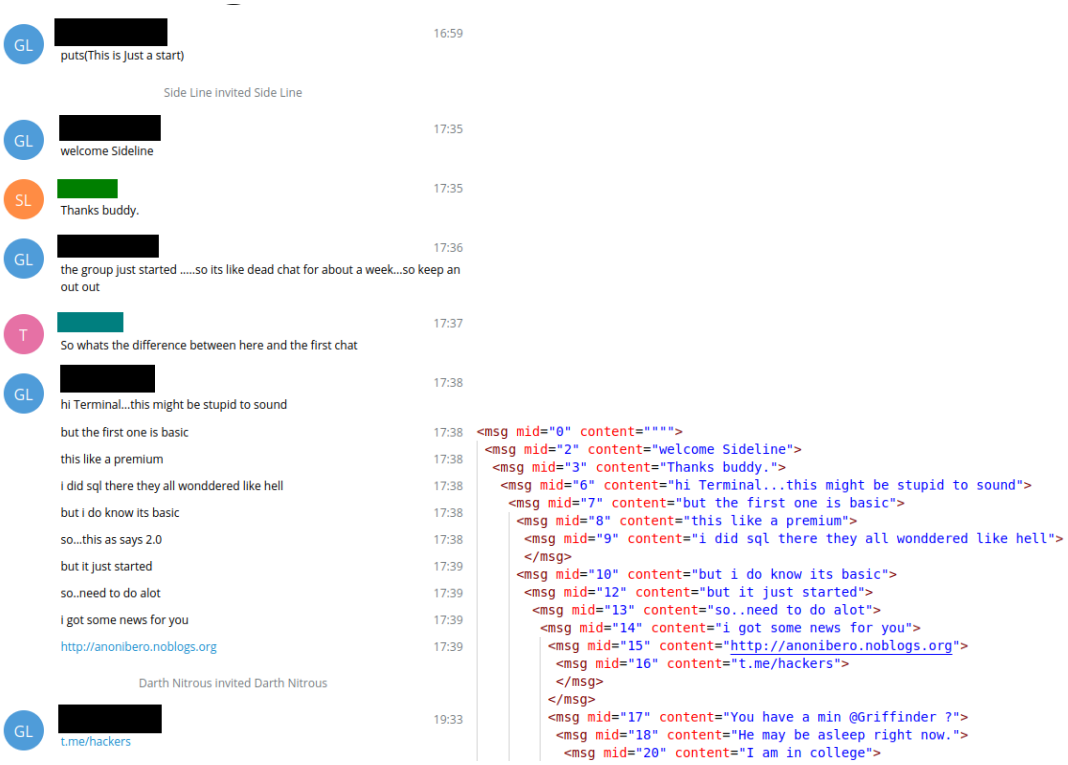


Figure 6.8 A screenshot of the original chat and the disentangled representation

6.6.3.3 Metrics

We evaluate our methods in two ways, first the post embeddings can be evaluated against TFIDF proposed by previous methods using SVMs and Random Forrest classifiers. Since there are no graphing mechanisms involved here, we use standard precision and recall metrics.

We use the F β score in the evaluation study (see table graph), which has weights for precision and recall.

6.7 Discussion

The positional and the linguistic feature embedding seem to have learned the good numerical representation in order to be structurally disentangled. We claim this because compared to the presented baselines such as [19] using a shallow embedding like TFIDF performed low in our setting. The main reason for this is the reconstruction system [19, 108] relies on meta-data which is not available in the data we are using. Without the said meta-data POSLAN was able to capture the representation very well as 6.2 shows it achieves about 0.91 F score. It is also noteworthy to see that by using the POSLAN vectors, it is possible to surpass the F score presented in [108] by 22.5%. Although this is a good gain, it is still needed to train the Random Forest with its own parameters making the final system more complicated. POSLAN has only one training component which is the computation of embeddings. We then use a simple graph processing algorithms PRN and THN which merely has one parameter to extract the disentanglement.

POSLAN system achieves gains compared with the two baselines. Motivation to build new graph processing algorithms comes from the series of experiments we did with shallow embeddings and graph processing algorithms. We noticed that the Graph Representation methods from [108] reduces the F β score down to 0.103. From this we create our own graph processing methods and utilize them inorder to increase the gain. We are able to bring the F score to up to 0.519.

Similar to how disadvantageous evaluation procedure is for [19, 108] in the meta-data restricted dataset, due to the Hawkes process the model assumes that conversations dying

out and schisms happens in the Twitter dataset as well. This is not the case, in Section data we explain the drawbacks of using the Twitter dataset.

One drawback in our graph processing methods is that because of the top down approach, we do not consider one message could be a reply to multiple previous messages. Since [108, 19] does not have this assumption of Hawkes process you can see the graph recreation method presented by Wang et al. works better with shallow embeddings. To show how our graph processing method works, we have taken an excerpt of the real chat and the reconstructed chat structure.

Despite the one draw back, our focus is at the disentanglement of Telegram group chats, while we prove the POSLAN vectors provide the best embeddings our graph processing mechanisms work for meta-data restricted datasets (Figure 6.8b).

6.8 Conclusions

This article presents a disentanglement method for threaded discussions where the reply relation will be discovered using a position and linguistic feature encoded embedding with the help of a graph pruning algorithm. From the results it is apparent that the positional and the language feature encoded post embeddings do help in disentangling chat and is an improvement upon the baselines. In order to improve upon the prior work we have used the LSTM encoder for the positional and linguistic features and Hawkes process method to determine where a conversation starts after a previous one dying.

While our method removes two phases of training similar to [108] it is also not a complete

Method	Precision	Recall	F β Score
TFIDF + SVM[19]	0.499	0.499	0.499
TFIDF + RF	0.520	0.501	0.501
POSLAN + SVM	0.499	0.500	0.499
POSLAN + RF	**0.994**	**0.863**	**0.918**

Table 6.2 Results of methods without graph representations; SVM - Support Vector Machine Classification, RF - Random Forest Classification

Method	Precision	Recall	F β Score
TFIDF + GR	0.500	0.642	0.397
POSLAN + GR	0.499	0.490	0.103
POSLAN + PRN + THN	**0.527**	**0.514**	**0.519**

Table 6.3 Results of methods with graph representations; GR - Graph Representation introduced in [108]

end to end process. The graph processing method has its own parameters and the LSTM network obviously has its own. We have fine tuned our parameters for the data but for further scaling and managing it would be much easier to have one system which needs to be trained or parameter tuned once.

CHAPTER 7
CONCLUSIONS

In this chapter we shall revisit the research questions and show how our work that we had discussed earlier solves them. We explained how each chapter of the work presented here are from publications that we have done in the past and all of them are peer-reviewed articles. Each paper tries to handle one or more research questions that we had presented.

1. What is a comprehensive public opinion?

2. Collecting a dataset to extract public opinions about COVID19 vaccine.

3. What are weak indications for stances?

4. Finding propagation methods to infer non-participating user stance.

5. Introducing weak supervision through found weak indications.

In Chapter 2, 3 and 5 we discussed how the majority of users in social networks are not participating in the public conversation. While they still consume the content, they also are forming their own opinions based on the content. The opinions will not be visible for traditional natural language processing methods because the users are not participating in the conversation. Instead, we will see the users opinions behind the users they are following and the tweets they are liking and retweeting. If we find the opinions of these silent or non-participating users we will have extracted a comprehensive public opinion which includes silent participants as well. This answers our first research question.

Next we explore what type of data we have in-order to infer stances on social networks. We also find that existing datasets such as [74, 35] predate the novel-coronavirus pandemic and using such datasets to find public opinions will not be ideal. To this end we have collected our own data from Twitter which is a social network. This dataset includes various levels of data, first we have linguistic data from tweets and some other meta-data provided by the Twitter API. In some tweets we see that sometimes links are being cited; we have managed to scrape all the articles which these URLs are linking to, this will in turn be extra-linguistic features. We have also collected the user accounts which these tweets were composed using and all their followers as well. This allows us to look at the data in various levels such as linguistic and graph data. At the time of publishing we had not seen any other dataset which provides this level of information for the stance inference problem.

After finding data, we look at how we could potentially extract the stances. There are levels of stances such as statement level and user level stance. We are trying to extract something in-between called the article based stance. We explore the differences of this version of stance inference and others in Chapter 4. Here we mainly look at how the users express their opinion based on the article which they have shared along with the tweet. To this end we have created unsupervised methods. The motivation to build unsupervised methods come from the fact that the nature of our data and the complexity of the problem. To annotate the data we may have to go into multiple steps of experts reading these articles and inspecting the users. Therefore, we try to solve this issue by an unsupervised method where the algorithm learns the hidden natural structure. We utilize a sentence transformer

to embed articles and tweets into a multidimensional vector so that we understand how different the users comment on the article is to the article this can be then used as the stance of the user towards that article. The process involves a clustering method called "MeanShift". Here we do not have to determine the clusters in advance and the clustering seem to extract the complex themes/targets the articles are written under. After this we use a cosine similarity measurement to express the difference of opinion as stance of the user against the article.

Lastly in the process of finding public opinion via user stances, we need to be able to tell the stances of users who are consuming other opinions. To this end we adapt the Label propagation method. We also understand by using the unsupervised methods we were not able to reach good performances previously. Therefore, we add weak supervision to this task via hashtags. A user on Twitter may post their tweets and possibly attach a hashtag to the tweet as well. Other users who are consuming these tweets will read the tweet and the hashtag. Typically, these hashtags are context indicators or topics of what the user describes in the tweet. Therefore, we can also reduce these tweets down into a hashtag. This means all the users could also be reduced to their most used hashtags. It is easier to tell the stance of a hashtag now, because this is not as convoluted as annotating a user anymore. Therefore, we claim that the weak supervision can come in forms of hashtags. Non-participating users will not be able to go through this reduction process. But we could potentially create a heterogeneous graph representing both hashtags and users. We could add edges between users and users if there is a follower relationship. If a user had tweeted something with a

hashtag, we could add an edge between a user and a hashtag. If two hashtags are appearing in the same tweet, we can add edges between them. Using this heterogeneous graph we experiment with a seed set of labels and Label Propagation to find stances of other users and hashtags. We have reported our experiments in Chapter 5 where our models reach 94% macro averaged F-1 score.

In Chapter 6, we explore a completely different idea. Up until now we discussed how it is possible to infer the stances of users who are not participating. But this discussion was based off of threaded social media networks. However, there is another social media network we observe called the Chronological networks (Whatsapp, Telegram). In these network typically there are no connections between users, not even following relationships. Also, there is no relationship with messages, it is quite difficult to understand which message links to what other message. This also implies that it is going to be difficult to understand which users are really connected. If we can represent the messages and in turn users within a graph it is going to be possible to run the same Label Propagation algorithm in Chapter 5. To this end we propose a LSTM network which learns the positional and linguistic information of a message in a message thread. Positional information will allow us to understand where the message is originally and what surrounding messages are available for us to determine the parent of this message. Linguistic features will linguistically evaluate whether the message is relating to some other message within a range. In-order to determine ranges we utilize point processes and a naive implementation of Hawkes process. We experimentally show that using the embeddings created by this system it is possible to disentangle the chronological

chat we receive 91.8 F-1 score.

REFERENCES

[1] Channels FAQ. https://www.telegram.org/faq_channels.

[2] Sentence-transformers/all-MiniLM-L12-v2 · Hugging Face. https://huggingface.co/sentence-transformers/all-MiniLM-L12-v2, .

[3] Sentence-transformers/all-distilroberta-v1 · Hugging Face. https://huggingface.co/sentence-transformers/all-distilroberta-v1, .

[4] Sentence-transformers/all-mpnet-base-v2 · Hugging Face. https://huggingface.co/sentence-transformers/all-mpnet-base-v2, .

[5] Telegram FAQ. https://www.telegram.org/faq#q-what-39s-the-difference-between-groups-and-channels.

[6] VADER: A Parsimonious Rule-Based Model for Sentiment Analysis of Social Media Text | Proceedings of the International AAAI Conference on Web and Social Media. https://ojs.aaai.org/index.php/ICWSM/article/view/14550.

[7] Bhashithe Abeysinghe, Gyandeep Reddy Vulupala, Anu G. Bourgeois, and Rajshekhar Sunderraman. Unsupervised User Stance Detection on Tweets Against Web Articles Using Sentence Transformers. In *Proceedings of Parallel and Distributed Processing Symposium*, Virtual, June 2022. IEEE.

[8] Bhashithe Abeysinghe, Gyandeep Reddy Vulupala, and Rajshekhar Sunderraman. Misinformation in Social Media Platforms and Web Articles: A Dataset to Infer User Stance. In *Proceedings of 16th IEEE International Conference on Semantic Computing*,

Online, February 2022. IEEE.

[9] Mariam Adedoyin-Olowe, Mohamed Medhat Gaber, and Frederic Stahl. A Survey of Data Mining Techniques for Social Network Analysis. page 25, 2013.

[10] Charu C. Aggarwal and ChengXiang Zhai, editors. *Mining Text Data*. Springer US, Boston, MA, 2012. ISBN 978-1-4614-3222-7 978-1-4614-3223-4. doi: 10.1007/978-1-4614-3223-4.

[11] Abdulrahman I. Al-Ghadir, Aqil M. Azmi, and Amir Hussain. A novel approach to stance detection in social media tweets by fusing ranked lists and sentiments. *Information Fusion*, 67:29–40, March 2021. ISSN 1566-2535. doi: 10.1016/j.inffus.2020.10.003.

[12] Abeer Aldayel and Walid Magdy. Your Stance is Exposed! Analysing Possible Factors for Stance Detection on Social Media. *Proceedings of the ACM on Human-Computer Interaction*, 3(CSCW):1–20, November 2019. ISSN 2573-0142. doi: 10.1145/3359307.

[13] Abeer ALDayel and Walid Magdy. Stance detection on social media: State of the art and trends. *Information Processing & Management*, 58(4):102597, July 2021. ISSN 0306-4573. doi: 10.1016/j.ipm.2021.102597.

[14] Abeer Aldayel and Walid Magdy. Characterizing the role of bots' in polarized stance on social media. *Social Network Analysis and Mining*, 12(1):30, February 2022. ISSN 1869-5469. doi: 10.1007/s13278-022-00858-z.

[15] Zulfikar Alom, Barbara Carminati, and Elena Ferrari. A deep learning model for Twitter spam detection. *Online Social Networks and Media*, 18:100079, July 2020. ISSN 2468-6964. doi: 10.1016/j.osnem.2020.100079.

[16] Pablo Aragón, Vicenç Gómez, and Andreaks Kaltenbrunner. To Thread or Not to Thread: The Impact of Conversation Threading on Online Discussion. *Proceedings of the International AAAI Conference on Web and Social Media*, 11(1):12–21, May 2017. ISSN 2334-0770.

[17] Farzindar Atefeh and Wael Khreich. A survey of techniques for event detection in twitter. *Computational Intelligence*, 31(1):132–164, 2015.

[18] Isabelle Augenstein, Tim Rocktäschel, Andreas Vlachos, and Kalina Bontcheva. Stance Detection with Bidirectional Conditional Encoding. *arXiv:1606.05464 [cs]*, September 2016.

[19] Erik Aumayr, Jeffrey Chan, and Conor Hayes. Reconstruction of threaded conversations in online discussion forums. In *Fifth International AAAI Conference on Weblogs and Social Media*, 2011.

[20] Nikhil Bansal, Avrim Blum, and Shuchi Chawla. Correlation clustering. *Machine learning*, 56(1):89–113, 2004.

[21] Javier Borge-Holthoefer, Walid Magdy, Kareem Darwish, and Ingmar Weber. Content and Network Dynamics Behind Egyptian Political Polarization on Twitter, October 2014.

[22] Peter Bourgonje, Julian Moreno Schneider, and Georg Rehm. From Clickbait to Fake News Detection: An Approach based on Detecting the Stance of Headlines to Articles. In *Proceedings of the 2017 EMNLP Workshop: Natural Language Processing Meets Journalism*, pages 84–89, Copenhagen, Denmark, 2017. Association for Computational

Linguistics. doi: 10.18653/v1/W17-4215.

[23] Eric Brill and Robert C. Moore. An improved error model for noisy channel spelling correction. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics - ACL '00*, pages 286–293, Hong Kong, 2000. Association for Computational Linguistics. doi: 10.3115/1075218.1075255.

[24] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language Models are Few-Shot Learners. *arXiv:2005.14165 [cs]*, July 2020.

[25] Talha Burki. Vaccine misinformation and social media. *The Lancet Digital Health*, 1 (6):e258–e259, October 2019. ISSN 2589-7500. doi: 10.1016/S2589-7500(19)30136-0.

[26] Josemar Alves Caetano, Jaqueline Faria de Oliveira, Helder Seixas Lima, Humberto T. Marques-Neto, Gabriel Magno, Wagner Meira Jr, and Virgílio A. F. Almeida. Analyzing and characterizing political discussions in WhatsApp public groups. *arXiv:1804.00397 [cs]*, April 2018.

[27] Vitor R Carvalho and William W Cohen. Learning to Extract Signature and Reply Lines from Email. *Proceedings of the Conference on Email and Anti-Spam*, page 8, 2004.

[28] Wei-Fan Chen and Lun-Wei Ku. UTCNN: A Deep Learning Model of Stance Classificationon on Social Media Text. *arXiv:1611.03599 [cs]*, November 2016.

[29] Kareem Darwish, Peter Stefanov, Michaël Aupetit, and Preslav Nakov. Unsupervised User Stance Detection on Twitter. *Proceedings of the International AAAI Conference on Web and Social Media*, 14:141–152, May 2020. ISSN 2334-0770.

[30] Richard I. A. Davis. Fake News , Real Consequences : Recruiting Neural Networks for the Fight Against Fake News. https://www.semanticscholar.org/paper/Fake-News-%2C-Real-Consequences-%3A-Recruiting-Neural-Davis/d5e2b45e6bb7bcb507f257d1fe773191255ad21b, 2017.

[31] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv:1810.04805 [cs]*, May 2019.

[32] Chedia Dhaoui, Cynthia M. Webster, and Lay Peng Tan. Social media sentiment analysis: Lexicon versus machine learning. *Journal of Consumer Marketing*, 34(6): 480–488, September 2017. ISSN 0736-3761. doi: 10.1108/JCM-03-2017-2141.

[33] Micha Elsner and Eugene Charniak. You Talking to Me? A Corpus and Algorithm for Conversation Disentanglement. page 9.

[34] Micha Elsner and Eugene Charniak. Disentangling Chat. *Computational Linguistics*, 36(3):389–409, September 2010. ISSN 0891-2017, 1530-9312. doi: 10.1162/coli_a_00003.

[35] William Ferreira and Andreas Vlachos. Emergent: A novel data-set for stance clas-

sification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1163–1168, San Diego, California, 2016. Association for Computational Linguistics. doi: 10.18653/v1/N16-1138.

[36] Ophélie Fraisier, Guillaume Cabanac, Yoann Pitarch, Romaric Besançon, and Mohand Boughanem. Stance Classification through Proximity-based Community Detection. In *Proceedings of the 29th on Hypertext and Social Media*, pages 220–228, Baltimore MD USA, July 2018. ACM. ISBN 978-1-4503-5427-1. doi: 10.1145/3209542.3209549.

[37] Alex Galarza. Documenting the Now. *Journal of American History*, 105(3):792–793, December 2018. ISSN 0021-8723. doi: 10.1093/jahist/jay444.

[38] Kiran Garimella, Gianmarco De Francisci Morales, Aristides Gionis, and Michael Mathioudakis. Reducing Controversy by Connecting Opposing Views. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, WSDM '17, pages 81–90, New York, NY, USA, February 2017. Association for Computing Machinery. ISBN 978-1-4503-4675-7. doi: 10.1145/3018661.3018703.

[39] Bogdan Georgescu, Ilan Shimshoni, and Peter Meer. Mean shift based clustering in high dimensions: A texture classification example. pages 456–463, 2003.

[40] Anna Glazkova, Maksim Glazkov, and Timofey Trifonov. G2tmn at Constraint@AAAI2021: Exploiting CT-BERT and Ensembling Learning for COVID-19 Fake News Detection. volume 1402, pages 116–127. 2021. doi: 10.1007/978-3-030-73696-5_12.

[41] Andrew R Golding. A Winnow-Based Approach to Context-Sensitive Spelling Correction. page 24.

[42] Jiafeng Guo, Yixing Fan, Liang Pang, Liu Yang, Qingyao Ai, Hamed Zamani, Chen Wu, W. Bruce Croft, and Xueqi Cheng. A Deep Look into neural ranking models for information retrieval. *Information Processing & Management*, 57(6):102067, November 2020. ISSN 0306-4573. doi: 10.1016/j.ipm.2019.102067.

[43] Ivan Habernal, Henning Wachsmuth, Iryna Gurevych, and Benno Stein. The Argument Reasoning Comprehension Task: Identification and Reconstruction of Implicit Warrants. August 2017. doi: 10.18653/v1/N18-1175.

[44] Wu He, Shenghua Zha, and Ling Li. Social media competitive analysis and text mining: A case study in the pizza industry. *International Journal of Information Management*, 33(3):464–472, June 2013. ISSN 0268-4012. doi: 10.1016/j.ijinfomgt.2013.01.001.

[45] Jonathan L Herlocker, Joseph A Konstan, and John Riedl. Explaining collaborative filtering recommendations. In *Proceedings of the 2000 ACM Conference on Computer Supported Cooperative Work*, pages 241–250, 2000.

[46] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[47] Qian Huang, Horace He, Abhay Singh, Ser-Nam Lim, and Austin R. Benson. Combining Label Propagation and Simple Models Out-performs Graph Neural Networks, November 2020.

[48] Doaa Mohey El-Din Mohamed Hussein. A survey on sentiment analysis challenges.

Journal of King Saud University - Engineering Sciences, 30(4):330–338, October 2018. ISSN 1018-3639. doi: 10.1016/j.jksues.2016.04.002.

[49] Amelia Jamison, David A Broniatowski, Michael C Smith, Kajal S Parikh, Adeena Malik, Mark Dredze, and Sandra C Quinn. Adapting and extending a typology to identify vaccine misinformation on Twitter. *American Journal of Public Health*, 110 (S3):S331–S339, 2020.

[50] Andrzej Jarynowski, Alexander Semenov, Mikołaj Kamiński, and Vitaly Belik. Mild Adverse Events of Sputnik V Vaccine Extracted from Russian Language Telegram Posts via BERT Deep Learning Model, June 2021.

[51] Jyun-Yu Jiang, Francine Chen, Yan-Ying Chen, and Wei Wang. Learning to Disentangle Interleaved Conversational Threads with a Siamese Hierarchical Network and Similarity Ranking. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1812–1822, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1164.

[52] Zhao Jianqiang, Gui Xiaolin, and Zhang Xuejun. Deep Convolution Neural Networks for Twitter Sentiment Analysis. *IEEE Access*, 6:23253–23260, 2018. ISSN 2169-3536. doi: 10.1109/ACCESS.2017.2776930.

[53] Nitin Jindal and Bing Liu. Review spam detection. In *Proceedings of the 16th International Conference on World Wide Web - WWW '07*, page 1189, Banff, Alberta, Canada, 2007. ACM Press. ISBN 978-1-59593-654-7. doi: 10.1145/1242572.1242759.

[54] Thorsten Joachims. A probabilistic analysis of the rocchio algorithm with TFIDF for text categorization. Technical report, Carnegie-mellon univ pittsburgh pa dept of computer science, 1996.

[55] Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. Bag of Tricks for Efficient Text Classification. *arXiv:1607.01759 [cs]*, August 2016.

[56] Sawinder Kaur. Automating fake news detection system using multi-level voting model. page 21.

[57] Simon Kemp. Digital 2021 October Global Statshot Report. https://datareportal.com/reports/digital-2021-october-global-statshot, October 2021.

[58] Elena Kochkina, Maria Liakata, and Isabelle Augenstein. Turing at SemEval-2017 Task 8: Sequential Approach to Rumour Stance Classification with Branch-LSTM. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 475–480, Vancouver, Canada, 2017. Association for Computational Linguistics. doi: 10.18653/v1/S17-2083.

[59] Jonathan K. Kummerfeld, Sai R. Gouravajhala, Joseph J. Peper, Vignesh Athreya, Chulaka Gunasekara, Jatin Ganhotra, Siva Sankalp Patel, Lazaros C Polymenakos, and Walter Lasecki. A Large-Scale Corpus for Conversation Disentanglement. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3846–3856, Florence, Italy, 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1374.

[60] An Ngoc Lam, Anh Tuan Nguyen, Hoan Anh Nguyen, and Tien N. Nguyen. Bug Localization with Combination of Deep Learning and Information Retrieval. In *2017 IEEE/ACM 25th International Conference on Program Comprehension (ICPC)*, pages 218–229, May 2017. doi: 10.1109/ICPC.2017.24.

[61] Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. Neural Architectures for Named Entity Recognition. *arXiv:1603.01360 [cs]*, April 2016.

[62] Hang Li. Learning to rank for information retrieval and natural language processing. *Synthesis lectures on human language technologies*, 7(3):1–121, 2014.

[63] Hang Li and Zhengdong Lu. Deep learning for information retrieval. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1203–1206, 2016.

[64] Jing Li, Aixin Sun, Jianglei Han, and Chenliang Li. A Survey on Deep Learning for Named Entity Recognition. *IEEE Transactions on Knowledge and Data Engineering*, pages 1–1, 2020. ISSN 1558-2191. doi: 10.1109/TKDE.2020.2981314.

[65] Chenghua Lin and Yulan He. Joint sentiment/topic model for sentiment analysis. *Proceedings of the 18th ACM conference on Information and knowledge management*, page 10, 2009.

[66] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv:1907.11692 [cs]*, July 2019.

[67] Walid Magdy, Kareem Darwish, and Ingmar Weber. #FailedRevolutions: Using Twitter to Study the Antecedents of ISIS Support. *arXiv:1503.02401 [physics]*, March 2015.

[68] Benjamin Markines, Ciro Cattuto, and Filippo Menczer. Social spam detection. In *Proceedings of the 5th International Workshop on Adversarial Information Retrieval on the Web - AIRWeb '09*, page 41, Madrid, Spain, 2009. ACM Press. ISBN 978-1-60558-438-6. doi: 10.1145/1531914.1531924.

[69] Eric Mays, Fred J Damerau, and Robert L Mercer. Context based spelling correction. *Information Processing & Management*, 27(5):517–522, 1991.

[70] Walaa Medhat, Ahmed Hassan, and Hoda Korashy. Sentiment analysis algorithms and applications: A survey. *Ain Shams Engineering Journal*, 5(4):1093–1113, December 2014. ISSN 2090-4479. doi: 10.1016/j.asej.2014.04.011.

[71] Yelena Mejova. Sentiment Analysis: An Overview. page 35, 2009.

[72] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119, 2013.

[73] Shervin Minaee, Nal Kalchbrenner, Erik Cambria, Narjes Nikzad, Meysam Chenaghlu, and Jianfeng Gao. Deep Learning–based Text Classification: A Comprehensive Review. *ACM Computing Surveys*, 54(3):1–40, June 2021. ISSN 0360-0300, 1557-7341. doi: 10.1145/3439726.

[74] Saif Mohammad, Svetlana Kiritchenko, Parinaz Sobhani, Xiaodan Zhu, and Colin

Cherry. SemEval-2016 Task 6: Detecting Stance in Tweets. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 31–41, San Diego, California, 2016. Association for Computational Linguistics. doi: 10.18653/v1/S16-1003.

[75] David Nadeau and Satoshi Sekine. A survey of named entity recognition and classification. page 20, 2007.

[76] Tetsuya Nasukawa and Jeonghee Yi. Capturing Favorability Using Natural Language Processing. *Proceedings of the 2nd international conference on Knowledge capture*, page 8, 2003.

[77] Lynnette Hui Xian Ng and Jia Yuan Loke. Analyzing Public Opinion and Misinformation in a COVID-19 Telegram Group Chat. *IEEE Internet Computing*, 25(2):84–91, March 2021. ISSN 1941-0131. doi: 10.1109/MIC.2020.3040516.

[78] Blair Nonnecke and Jenny Preece. Silent Participants: Getting to Know Lurkers Better. In Christopher Lueg and Danyel Fisher, editors, *From Usenet to CoWebs: Interacting with Social Information Spaces*, Computer Supported Cooperative Work, pages 110–132. Springer, London, 2003. ISBN 978-1-4471-0057-7. doi: 10.1007/978-1-4471-0057-7_6.

[79] Joseph J Pollock and Antonio Zamora. Automatic spelling correction in scientific and scholarly text. *Communications of the ACM*, 27(4):358–368, 1984.

[80] Martin Potthast, Sebastian Köpsel, Benno Stein, and Matthias Hagen. Clickbait detection. In *European Conference on Information Retrieval*, pages 810–817. Springer,

2016.

[81] Rudy Prabowo and Mike Thelwall. Sentiment analysis: A combined approach. *Journal of Informetrics*, 3(2):143–157, April 2009. ISSN 1751-1577. doi: 10.1016/j.joi.2009.01.003.

[82] William H Press and Saul A Teukolsky. Savitzky-golay smoothing filters. *Computers in Physics*, 4(6):669–672, 1990.

[83] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[84] Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. *arXiv:1908.10084 [cs]*, August 2019.

[85] Gustavo Resende, Philipe Melo, Julio C. S. Reis, Marisa Vasconcelos, Jussara M. Almeida, and Fabrício Benevenuto. Analyzing Textual (Mis)Information Shared in WhatsApp Groups. In *11th ACM Conference on Web Science*, 2019.

[86] Benjamin Riedel, Isabelle Augenstein, Georgios P. Spithourakis, and Sebastian Riedel. A simple but tough-to-beat baseline for the Fake News Challenge stance detection task, May 2018.

[87] Younes Samih and Kareem Darwish. A Few Topical Tweets are Enough for Effective User Stance Detection. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 2637–2646, Online, April 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.eacl-main.227.

[88] Craig Saunders, Mark O Stitson, Jason Weston, Leon Bottou, A Smola, et al. Support vector machine-reference manual. 1998.

[89] Kim Schouten and Flavius Frasincar. Survey on Aspect-Level Sentiment Analysis. *IEEE Transactions on Knowledge and Data Engineering*, 28(3):813–830, March 2016. ISSN 1558-2191. doi: 10.1109/TKDE.2015.2485209.

[90] Dou Shen, Qiang Yang, Jian-Tao Sun, and Zheng Chen. Thread detection in dynamic text message streams. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 35–42, 2006.

[91] Valeriya Slovikovskaya. Transfer Learning from Transformers to Fake News Challenge Stance Detection (FNC-1) Task, October 2019.

[92] Nico Smuts. What Drives Cryptocurrency Prices? An Investigation of Google Trends and Telegram Sentiment. page 4.

[93] Parinaz Sobhani, Saif Mohammad, and Svetlana Kiritchenko. Detecting Stance in Tweets And Analyzing its Interaction with Sentiment. In *Proceedings of the Fifth Joint Conference on Lexical and Computational Semantics*, pages 159–169, Berlin, Germany, 2016. Association for Computational Linguistics. doi: 10.18653/v1/S16-2021.

[94] Brian G Southwell, Emily A Thorson, and Laura Sheble. *Misinformation and Mass Audiences*. University of Texas Press, 2018.

[95] Nikita Spirin and Jiawei Han. Survey on web spam detection: Principles and algorithms. *ACM SIGKDD Explorations Newsletter*, 13(2):50–64, May 2012. ISSN 1931-0145, 1931-0153. doi: 10.1145/2207243.2207252.

[96] Xiaoyuan Su and Taghi M. Khoshgoftaar. A Survey of Collaborative Filtering Techniques. *Advances in Artificial Intelligence*, 2009:1–19, October 2009. ISSN 1687-7470, 1687-7489. doi: 10.1155/2009/421425.

[97] Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. How to Fine-Tune BERT for Text Classification? *arXiv:1905.05583 [cs]*, February 2020.

[98] Shiliang Sun, Chen Luo, and Junyu Chen. A review of natural language processing techniques for opinion mining systems. *Information Fusion*, 36:10–25, July 2017. ISSN 1566-2535. doi: 10.1016/j.inffus.2016.10.004.

[99] Ming Tan, Dakuo Wang, Yupeng Gao, Haoyu Wang, Saloni Potdar, Xiaoxiao Guo, Shiyu Chang, and Mo Yu. Context-Aware Conversation Thread Detection in Multi-Party Chat. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6456–6461, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1682.

[100] Duyu Tang, Furu Wei, Bing Qin, Ting Liu, and Ming Zhou. Coooolll: A Deep Learning System for Twitter Sentiment Classification. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 208–212, Dublin, Ireland, 2014. Association for Computational Linguistics. doi: 10.3115/v1/S14-2033.

[101] Muhammad Umer, Zainab Imtiaz, Saleem Ullah, Arif Mehmood, Gyu Sang Choi, and Byung-Won On. Fake News Stance Detection Using Deep Learning Architecture (CNN-LSTM). *IEEE Access*, 8:156695–156706, 2020. ISSN 2169-3536. doi: 10.1109/

ACCESS.2020.3019735.

[102] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of machine learning research*, 9(11), 2008.

[103] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is All you Need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[104] Alex Hai Wang. DON'T FOLLOW ME: SPAM DETECTION IN TWITTER. page 10.

[105] Fei Wang and Changshui Zhang. Label Propagation Through Linear Neighborhoods. page 8.

[106] Hongning Wang, Chi Wang, ChengXiang Zhai, and Jiawei Han. Learning online discussion structures by conditional random fields. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information - SIGIR '11*, page 435, Beijing, China, 2011. ACM Press. ISBN 978-1-4503-0757-4. doi: 10.1145/2009916.2009976.

[107] Xiaolong Wang, Furu Wei, Xiaohua Liu, Ming Zhou, and Ming Zhang. Topic sentiment analysis in twitter: A graph-based hashtag sentiment classification approach. page 10, 2011.

[108] Yi-Chia Wang, Mahesh Joshi, William W Cohen, and Carolyn Penstein Rosé. Recovering implicit thread structure in newsgroup style conversations. In *ICWSM*, 2008.

[109] Zeyi Wen, Jiashuai Shi, Qinbin Li, Bingsheng He, and Jian Chen. ThunderSVM: A fast SVM library on GPUs and CPUs. *The Journal of Machine Learning Research*, 19

(1):797–801, 2018.

[110] Casey Whitelaw, Navendu Garg, and Shlomo Argamon. Using appraisal groups for sentiment analysis. In *Proceedings of the 14th ACM International Conference on Information and Knowledge Management - CIKM '05*, page 625, Bremen, Germany, 2005. ACM Press. ISBN 978-1-59593-140-5. doi: 10.1145/1099554.1099714.

[111] C. Wilke, S. Altmeyer, and T. Martinetz. Large-scale evolution and extinction in a hierarchically structured environment, March 1998.

[112] Vikas Yadav and Steven Bethard. A Survey on Recent Advances in Named Entity Recognition from Deep Learning models. *arXiv:1910.11470 [cs]*, October 2019.

[113] Hang Yan, Bocao Deng, Xiaonan Li, and Xipeng Qiu. TENER: Adapting Transformer Encoder for Named Entity Recognition. *arXiv:1911.04474 [cs]*, December 2019.

[114] Lin Yue, Weitong Chen, Xue Li, Wanli Zuo, and Minghao Yin. A survey of sentiment analysis in social media. *Knowledge and Information Systems*, 60(2):617–663, August 2019. ISSN 0219-3116. doi: 10.1007/s10115-018-1236-4.

[115] Guido Zarrella and Amy Marsh. MITRE at SemEval-2016 Task 6: Transfer Learning for Stance Detection. *arXiv:1606.03784 [cs]*, June 2016.

[116] Lei Zhang, Shuai Wang, and Bing Liu. Deep learning for sentiment analysis: A survey. *WIREs Data Mining and Knowledge Discovery*, 8(4), July 2018. ISSN 1942-4787, 1942-4795. doi: 10.1002/widm.1253.

[117] Zhenhua Zhang, Qing He, Jing Gao, and Ming Ni. A deep learning approach for detecting traffic accidents from social media data. *Transportation Research Part C:*

Emerging Technologies, 86:580–596, January 2018. ISSN 0968-090X. doi: 10.1016/j.trc.2017.11.027.

[118] Chunting Zhou, Chonglin Sun, Zhiyuan Liu, and Francis C. M. Lau. A C-LSTM Neural Network for Text Classification. *arXiv:1511.08630 [cs]*, November 2015.

[119] Jian Zhou, Hongyu Zhang, and David Lo. Where should the bugs be fixed? More accurate information retrieval-based bug localization based on bug reports. In *2012 34th International Conference on Software Engineering (ICSE)*, pages 14–24, June 2012. doi: 10.1109/ICSE.2012.6227210.

[120] Xinyi Zhou, Atishay Jain, Vir V. Phoha, and Reza Zafarani. Fake News Early Detection: A Theory-driven Model. *Digital Threats: Research and Practice*, 1(2):1–25, July 2020. ISSN 2692-1626, 2576-5337. doi: 10.1145/3377478.

[121] Ciyou Zhu, Richard H Byrd, Peihuang Lu, and Jorge Nocedal. Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization. *ACM Transactions on mathematical software (TOMS)*, 23(4):550–560, 1997.

[122] Xiaojin Zhu and Zoubin Ghahramani. Learning from labeled and unlabeled data with label propagation. 2002.

[123] Arkaitz Zubiaga, Maria Liakata, Rob Procter, Geraldine Wong Sak Hoi, and Peter Tolmie. Analysing How People Orient to and Spread Rumours in Social Media by Looking at Conversational Threads. *arXiv:1511.07487 [cs]*, February 2016. doi: 10.1371/journal.pone.0150989.

[124] Arkaitz Zubiaga, Elena Kochkina, Maria Liakata, Rob Procter, Michal Lukasik, Kalina

Bontcheva, Trevor Cohn, and Isabelle Augenstein. Discourse-aware rumour stance classification in social media using sequential classifiers. *Information Processing & Management*, 54(2):273–290, March 2018. ISSN 0306-4573. doi: 10.1016/j.ipm.2017.11.009.