

**Social Media Analytics for Stance Mining
A Multi-Modal Approach with Weak Supervision**

Sumeet Kumar

CMU-ISR-20-104

May 2020

Institute for Software Research
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:

Kathleen M. Carley (Chair)
Tom Mitchell
Louis-Philippe Morency
Huan Liu (Arizona State University)

*Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Societal Computing.*

Copyright © 2020 Sumeet Kumar

The research reported in this thesis has been supported in part by the ONR Award No. N00014182106, ONR Award No. N0001418SB001 and the Center for Computational Analysis of Social and Organization Systems (CASOS). Any opinions, findings, conclusions, or recommendations expressed in this material are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of the funding agencies.

Keywords: Social Media, Social Networks, Opinion Mining, Semi-Supervised, Stance

To those who make the world a better place.

Abstract

People express their opinions on blogs and other social media platforms. As per a recent estimate, interactions on Twitter alone result in over 500 million tweets per day. The magnitude of this data enables new applications of opinion mining that have previously remained challenging e.g., finding users' stance (as in pro or con) on topics of interest. However, one of the major barriers to utilizing this amount of data is the cost of hand-labeling examples for machine learning. This barrier is even more apparent in stance mining, as opinions can change overtime and can be about any issues. To reduce the need for hand-labeled data by taking the complex interactions of social media users and their social influence into account, this dissertation develops semi-supervised methods for stance mining.

Most existing studies on stance mining take a simplistic view that assumes a sentence (like a Tweet) holds a perspective that is independent of the context and the author's network position. This approach to stance learning leaves three crucial unresolved challenges. First, how do we train stance-learning models on new topics with minimal labeling effort? Discussion topics change fast and new issues emerge, making it difficult to reuse prior labeled data. However, artifacts of social networks like hashtags can give noisy signal about the stance of users. To extract the signal from noise, I develop methods to find useful hashtags by exploiting how users in the pro-group and the anti-group use popular hashtags. Second, how can we use multiple interaction modalities for stance mining? Users opinions are evident in different types of interactions, e.g. tweeting, retweeting or liking. I develop a semi-supervised method based on co-training that jointly trains multiple stance classifiers using different interaction modalities resulting in a better stance prediction model. Third, how to leverage users networks for stance prediction? The current approaches to stance learning ignore important network factors such as the interactions of social media users (e.g., a persons preference can also be known from his friends preferences). I use the network alignment as one of the training signals to train the stance classifiers.

My thesis brings a new direction to the stance learning problem that is grounded in social theory, is more amenable to analyzing activities on social media, and allows effective learning from multiple types of interactions without requiring large amounts of labeled data. By labeling only a few hashtags used in Twitter conversations on a few controversial topics, my approach allows for predicting both the stance of users (as in whether they are pro or con a topic) by over 80% accuracy and the stance in conversations (as in whether they favor or deny others posts) by over 70% accuracy.

Acknowledgments

I am very thankful to my advisor Prof. Kathleen M. Carley, for her support and encouragement. I am also very grateful to my other committee members for their kind support and advice throughout this journey. I would also like to thank all my co-authors and collaborators at CMU, including Matthew Babcock, Ramon Alfonso Villa Cox, David Beskow, and Binxuan Huang. I have learned a great deal from each one of them.

I am also grateful to my family members, in particular to my wife Rashmi, for her support in this pursuit. To my daughter Shreya who could never understand why her father is working all the time, I promise to have more free time in the future. I happen to be the first person in my family tree to pursue doctoral education. This could not have been possible without the confidence and freedom that my parents have bestowed on me.

Lastly, I am also thankful to other members at the Carnegie Mellon community who have been kind and helpful throughout this journey. In particular, I would like to thank Sienna Watkins for making this journey easier in many different ways. Kind words from professors, in particular, Prof. Tom Mitchell, Prof. James Herb-
sleb, Prof. Bob Iannaci, and Prof. Hakan Erdogmus, helped me to keep going in challenging times.

Contents

1	Introduction	1
1.1	Background	5
1.2	Prior Work	6
1.2.1	Learning Stance from Social-Media data	6
1.2.2	Learning Stance from Debates	7
1.2.3	Learning Stance Using Networks Data	7
1.2.4	Machine Learning Approaches to Stance Mining	8
1.2.5	Multi-modal Stance Mining – Models to Combine Different Types of Interactions	9
1.2.6	Applications of Stance Learning in Rumor and Misinformation Identification	9
1.2.7	Stance Datasets	10
1.3	Contributions	13
1.4	Organization of this Thesis	14
2	A New Stance Dataset and Some Baseline Models	19
2.1	Introduction	19
2.2	Related Work	21
2.2.1	Stance in Social-Media Posts	21
2.2.2	Stance in Online Debates and Conversations	21
2.3	Dataset Collection Methodology	22
2.3.1	Step 1: Determine Event	22
2.3.2	Step 2: Collect Tweets	23
2.3.3	Step 3: Determine Contentious Candidates	23
2.4	Sample Construction for Annotation	23
2.5	Annotation Procedure and Statistics	25
2.5.1	Definition of Classes	27
2.5.2	Inter Annotator Agreement	27
2.5.3	Distribution of Labeled Conversations	28
2.5.4	Distribution of Users’ Stance	29
2.6	Dataset Schema and FAIR principles	29
2.7	Baseline Models and Their Performance	30
2.7.1	Input Features	31
2.7.2	Classifiers	31

2.7.3	Classifiers Training	33
2.7.4	Results and Discussion	33
2.8	Conclusion and Future Work	34
3	Learning Users' Stance by Combining Users' Networks and Users' Posts by Creating Virtual Connections	37
3.1	Introduction	37
3.2	Related Work	38
3.3	People2Vec Model	39
3.3.1	Virtual Links from Users' Activities	40
3.3.2	People2Vec Algorithm	42
3.4	Experiments and Results	43
3.4.1	Datasets	43
3.4.2	Baseline Algorithms And Model Optimization	44
3.4.3	Experimental Results	44
3.4.4	Parameter Sensitivity	46
3.4.5	Results on Stance Dataset	48
3.5	Conclusions and Future Work	49
4	Co-Training on Social Networks: A Joint Network Label Propagation and Text Classification Approach for Stance Mining	51
4.1	Introduction	51
4.2	Background and Problem Formulation	53
4.2.1	Problem Statement	54
4.3	Methodology	54
4.3.1	Label Propagation on Bi-Partite Networks with Linear Threshold	54
4.3.2	Self-Trained Text Classifier	56
4.3.3	Co-Training of Label Propagation Model and Text Classifier	57
4.4	Experiments and Results	59
4.4.1	Dataset	59
4.4.2	Data Pre-processing	60
4.4.3	Seed Hashtags and Seed Users	60
4.4.4	Training and Hyper-parameter optimization	60
4.4.5	Baseline Models	61
4.4.6	Results and Discussion	63
4.4.7	Visualization of Results	64
4.4.8	Effect of Different Seed Examples	67
4.4.9	Effect of Hyper-Parameter Selection	69
4.5	Related Work	69
4.5.1	Stance Learning	69
4.5.2	Weakly Supervised Machine Learning for Stance Mining	70
4.5.3	Co-Training	70
4.6	Conclusion and Future Work	70

5	A Joint Network and Text based Model for Learning Stance in Conversations	73
5.1	Introduction	73
5.2	Empirical Evidence of Correlation Between ‘Stance in Conversations’ and ‘Stance of Users’	74
5.3	Methodology	76
5.3.1	Problem Statement	76
5.3.2	Step 1: Label Seed Nodes	77
5.3.3	Step 2: Label Edges from Seed Nodes	78
5.3.4	Step 3: Propagate Node Stance Labels to Other Nodes	78
5.3.5	Step 4: Predict Labels of Conversations using the Text Classifier	79
5.3.6	Step 5: Predict Node Labels using Edge Labels	80
5.3.7	Step 6: Label Mixing and Adding Confident Node Labels as New Labeled Data	81
5.3.8	Joint Model	81
5.4	Experiments and Results	82
5.4.1	Dataset Statistics	82
5.4.2	Expanding the Dataset by Including Users’ Timeline Data	82
5.4.3	Seed Hashtags and Seed Users	83
5.4.4	Baseline Models and Their Performance	84
5.4.5	Input Features	84
5.4.6	Classifiers	84
5.4.7	Training Process and Parameters	85
5.4.8	Results and Discussion	86
5.5	Related Work	88
5.5.1	Stance in Conversations	88
5.5.2	Signed Link Prediction	89
5.5.3	Joint Models for Text and Network Data	89
5.6	Conclusion and Future Work	89
6	Models of Tree Structured Conversations for Predicting Stance and Rumor Veracity	91
6.1	Introduction	91
6.2	Models of Tree Structured Social Media Conversations	93
6.2.1	Branch LSTM Model	93
6.2.2	Tree LSTM Model	94
6.2.3	Binarized Constituency Tree (BCTree) LSTM Model	96
6.3	Experiments and Results	97
6.3.1	Datasets	97
6.3.2	Feature Representation	98
6.3.3	Models Training	99
6.3.4	Stance Classification Results	101
6.3.5	Rumor Classification Results	102
6.4	Related Work	104
6.4.1	Stance Learning	104
6.4.2	Rumor and Misinformation Identification	104

6.4.3	LSTM and Convolutional Neural Networks	104
6.5	Conclusion	105
7	Conclusions, Limitation and Future Work	107
7.1	Conclusions	107
7.2	Scalability and Integration	108
7.3	Limitations	109
7.4	Future Work	111
7.4.1	What Would it Take to Use Data that Does not Have Hashtags?	111
7.4.2	Big Topics Might be Influencing Smaller topics. How to Handle This?	112
7.4.3	Extending the Stance Learning Models to Use Pictures and Memes Posted on Social Media	112
7.4.4	Analyzing Polarized Communities in Social-Networks via Stance Mining	113
7.4.5	Stance Mining for Discovering Echo-chambers on Social Media	113
A	Better Ways to Collect Twitter Data	129
A.1	Introduction	129
A.2	Twitter Data Collection Background	131
A.3	Problem Formulation	132
A.4	New Search Terms as a Knapsack Problem	134
A.5	A MAB Approach to Dynamically Update the Search Terms	135
A.6	Experiments and Results	137
A.7	Related Work	138
A.7.1	Twitter Data Collection and Event Detection	138
A.7.2	Multi-Armed Bandit Problems (MAB) and Web Crawling	138
A.8	Conclusion and Future Work	139
B	A Users Guide for Labeling Stance in Conversations	141

List of Figures

1.1	Multi-modal stance mining	2
1.2	Text based stance learning	3
1.3	Network based approaches to stance learning	4
1.4	Idea behind the proposed stance learning approach	4
1.5	The stance triangle	6
1.6	Thesis summary	17
2.1	Illustration of Stance in Conversations	20
2.2	Methodology developed for the collection of contentious tweet candidates for a specific event.	22
2.3	Dendogram derived for the Student Marches event data	24
2.4	3-dimensional representation of sample data obtained via Truncated Stochastic Value Decomposition	26
2.5	Form used for labeling replies	26
2.6	Form used for labeling quotes	27
2.7	Histogram of number of times tweets were annotated	28
2.8	Distribution of the labels among the different response types	29
2.9	Deep learning model sample diagram	32
2.10	Stance classification confusion matrix	34
3.1	Virtual links between nodes reveal hidden connections	38
3.2	An example of a random walk transition in People2ve	41
3.3	Comparison of different algorithms on BlogCatalog dataset	45
3.4	Comparison mean F1 score for different algorithms on Flickr dataset	45
3.5	F1-micro score for different α values using Euclidean distance as the measure of similarity.	47
3.6	F1-micro score for different α values using Hamming distance as the measure of similarity.	47
3.7	F1-micro score for different α value	48
3.8	F1-micro score for embeddings of different dimensions on Flickr dataset.	48
3.9	F1-micro score for different topics in a Stance dataset.	49
4.1	Multiple interactions on Twitter and summary of the proposed approach	52
4.2	Label propagation on bipartite graphs	54
4.3	Self-trained Text Classifier	56

4.4	Accuracy of the label propagation model	61
4.5	Accuracy of the label propagation for different values of θ^I	62
4.6	Accuracy of the text-classifier model	62
4.7	Classifiers performance over iterations	63
4.8	User-user Network Plots	66
4.9	User-user network with colors indicating stance obtained using text based prediction	66
4.10	User-user network based on common retweets with colors indicating the stance from the joint model	67
4.11	Trend of performance of the joint models for different values of user thresholds and interaction thresholds.	69
5.1	Stance in Conversations	73
5.2	Confusion Matrix for stance of users and stance in conversations	75
5.3	Confusion Matrix for stance of users and stance in conversations 2	75
5.4	System Design	76
5.5	Model of predicting stance of edges (conversations) as in favoring/denying from stance of users (pro/anti)	78
5.6	Step of predicting stance of nodes (pro/anti users) from stance of edges (as in favoring/denying in conversations)	80
5.7	Deep learning model diagram which was used in step 4 of the proposed system.	85
5.8	F1-Macro with iterations - weakly Supervised	86
5.9	F1-Macro with iterations - when other events text data is used while training	87
5.10	Confusion Matrix for true stance in conversations and predicted stance	87
6.1	A sample Twitter thread with stance and rumor-veracity labels	91
6.2	Models for Conversation Trees	92
6.3	Branch LSTM Model	94
6.4	Tree LSTM Model	94
6.5	BCTree LSTM Model	97
6.6	Normalized stance confusion matrix	102
6.7	Normalized rumor confusion matrix	103
7.1	Scalability and Integration	108
A.1	A sample Tweet sent after an earthquake	130
A.2	Tweets volume obtained on different days using all country names as search terms.	131
A.3	Cost function plot	132
A.4	Knapsack selection model	134
A.5	Results of earthquake data collection	138
A.6	Relevant vs non-relevant trend of data obtained using our proposed approaches	139

List of Tables

1.1	Manually labeled datasets created in prior research	11
1.2	SemEval 2016 Stance Dataset	11
1.3	Topics summary	12
1.4	Labeled users summary	12
1.5	Stance labels for Tweets in the conversations	13
1.6	Distribution of labels across different events.	13
2.1	Distribution of relevant tweet pairs by response type that could be labeled.	24
2.2	Distribution of relevant tweet pairs by response type	25
2.3	Distribution of relevant tweet pairs by response type	28
2.4	Distribution of labeled users' stance	30
2.5	Classification results for Replies	33
3.1	Dataset Description	43
3.2	Topics summary	44
3.3	Labeled users summary	44
3.4	Best F1 Macro Score	46
4.1	Topics summary	59
4.2	Labeled users summary	59
4.3	Seed hashtags, their labels, and the count of seed users obtained using seed hashtags	61
4.4	Performance of models on different dataset	65
4.5	Performance of co-trained joint model on bias-watch datasets with different seed hashtags. LP implies label propagation.	68
5.1	Dataset summary	82
5.2	Distribution of labeled replies across different events.	82
5.3	Summary of the enhanced dataset that includes users' timeline data	83
5.4	Seed hashtags and their labels	83
5.5	Performance of models on different dataset	88
6.1	Conversation threads in the PHEME dataset	98
6.2	Stance labels for Tweets in the conversations	98
6.3	Stance learning results	100
6.4	Rumor classification results	103

7.1 Time Complexity of Methods 110

Chapter 1

Introduction

People express their opinions on social media platforms. Automated ways to categorize views of people in such user generated corpora is of immense value. In particular, in polarized communities and echo-chambers which are increasingly more common these days, mining users opinion can enable situational awareness and help to bridge the communities. To mine social-media for opinions, stance learning – which involves finding people’s opinion about a topic of interest – provides a natural way to model social interactions, therefore lending itself to study users’ opinions on controversial topics. So it’s not a surprise that stance learning has become an active area of research [49, 59, 91].

However, most existing study on stance takes a simplistic view that assumes a ‘sentence’ (like a Tweet) holds a perspective that is independent of the context and the author. This approach to stance learning ignores the complex activities and interactions of social media users (see Fig. 1.1). Bois et al. [31] (page 163) defines the act of stance as ‘a public act by a social actor, achieved dialogically through overt communicative means, of simultaneously evaluating objects, positioning subjects (self and others), and aligning with other subjects...’. In the same spirit, I approach stance in a broader context of social action wherein authors interact on social-media to align themselves with other stance takers. This approach to stance mining allows to combine the technique’s of different approaches to stance mining, and paves way to a holistic approach that uses the multi-modal nature of user interactions on social media platforms.

Two common approaches to stance learning are natural language processing (NLP) based and network based. In Fig. 1.2, we show the NLP approach to stance mining, and in Fig. 1.3 we show a result from a network based approach (adapted from [25], Fig. 1). The NLP based approaches consider stance learning as a supervised natural language processing tasks, and uses a dictionary definition of stance as ‘a mental or emotional position adopted with respect to a proposition, a person, an idea, etc’¹. The tasks often come with three different but related learning goals: 1) stance from text (e.g. a twitter post about a single topic), 2) stance about a topic from multiple tweets (could come from a single source e.g. a single user), and c) stance on multiple topics from a single text message. These supervised approaches to stance mining have gained significant attention [59, 90, 91] but are challenging, as getting over 80% accuracy on a 2 class (pro/con) stance classification is difficult [91].

¹<https://www.thefreedictionary.com/stance>

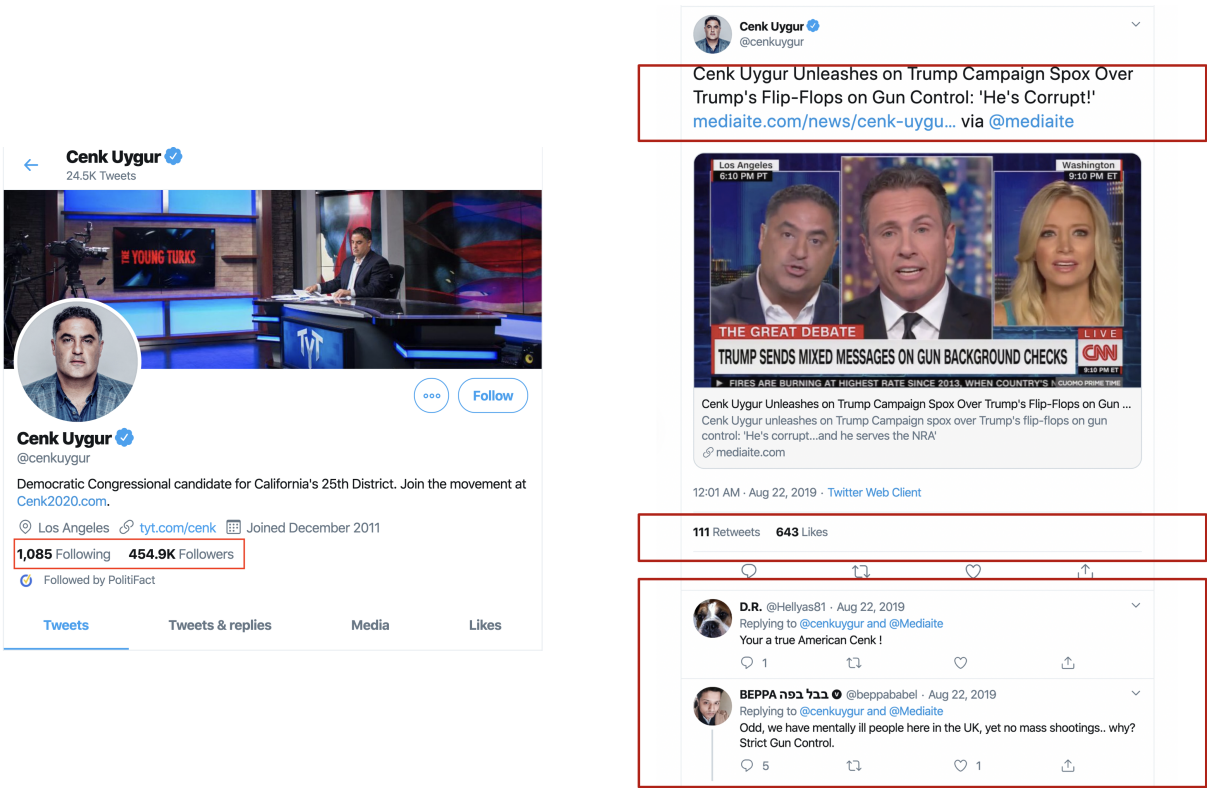


Figure 1.1: On the left, we show the Twitter profile of Cenk Uygur highlighting his ‘Followers’ and ‘Followings’. On the right, we show the activities related to one on his Tweets highlighting the ‘Retweets’, ‘Likes’ and ‘Replies’. As we can observe, the interactions in Twitter are inherently multi-modal which includes posts (often text based), networks (like follower, followings, retweets, likes etc.) and conversations (such as replies and quotes). However, most current stance learning models only consider one of the modalities, i.e. either the text or the networks for stance mining. This thesis explores approaches to learn stance of social-media users from their multi-modal interactions.

In contrast, the network based approaches commonly use a semi-supervised method where only the stance of a very few users are known (or hand labeled). Based on the stance of the known users, stance of other users could be determined. This approach to stance mining has lead to better results achieving over 80% accuracy (see [80] results). However, often networks used in such approaches is created by finding users’ involvement in certain interactions. For example, in Fig. 1.3, the network is composed on retweets based interactions. Often only type on interaction, like retweets networks, are only composed of a small fraction of users that are in the dataset, and in our experience, the fraction of users which are not in the retweets network could be even lesser than 50%. As the network based model is not able to predict the stance of many users, this limits the generalizability of these learning models. This limitation also affects the further downstream tasks like analyzing community polarization.

However, the two approaches to stance learning, i.e., stance from users’ texts and stance based on users’ networks, are not entirely independent. In fact, if we transform the main goal of

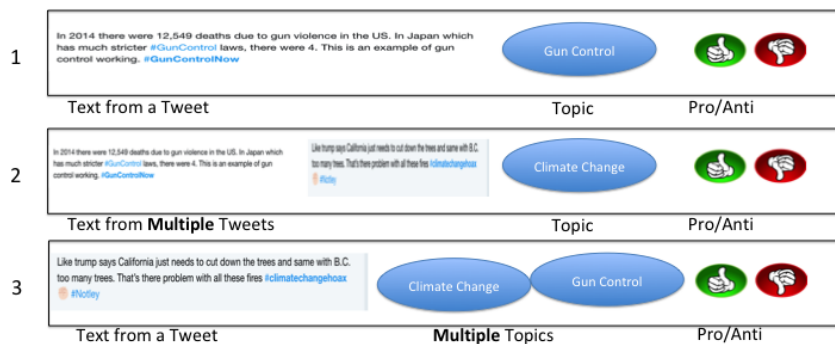


Figure 1.2: We show the common natural language processing (NLP) approaches to stance learning tasks. As shown, the general approach is to have a text (e.g., from a Twitter post) and the task is to infer the stance with respect to a topic.

a research to learning the stance of users' (and not text or retweet nodes), users' posts and users' networks are just two types of interactions users engage in. The preference to certain entities (as in pro stance about a topic or a person), leads to specific pattern in the way users engage on social-media platforms. For example, if one likes 'Cenk Uygur', it is natural for the user to follow Cenk Uygur and retweet Cenk Uygur's tweets. Another person, that does not favor 'Cenk Uygur', he or she could still follow 'Cenk Uygur', but is less likely to retweet Cenk Uygur's tweets and, perhaps, is more likely to engage in 'quoting' and 'replying' to his tweets. Therefore, we can rethink stance mining as a multi-modal problem where text based posts, networks and conversations (as in replying/quoting) are just different modes of conveying ones' stance. In fact, some of these modalities could bring complementary information about users' stance that could not be possible by just considering at one modality. We use these complementary views to propose learning models that better learn the stance of social-media users.

Stance learning systems should not focus on one particular interaction but rather consider all sets of interactions that could reveal users' stance. This thesis takes the multi-modality of stance as the central theme and look at the various ways these modalities can be used to develop better stance learning models (see Fig. 1.4). Unlike prior work, our approach positions users as central and their alignment as a crucial to learning stance. This approach, thus, brings a new direction to the stance learning problem which is grounded in theory and is more amenable to conversations on social-media. This research can be applied to improving the understanding of community polarization and partisanship.

Next, I provide some background on stance taking, and related prior work in which I also describe the existing and the new dataset that I use in the research. Then I summarize the main contributions of this these and the summary of the chapters.

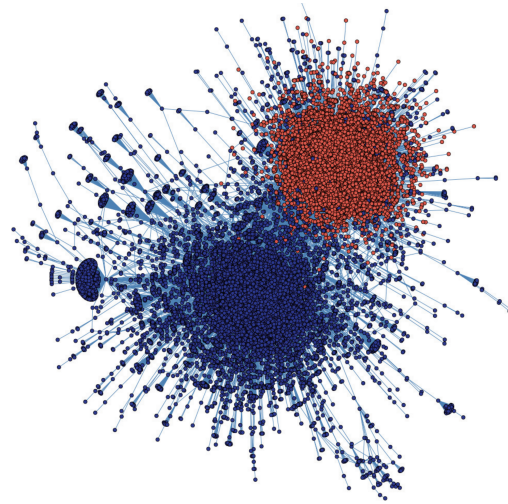


Figure 1.3: We show a result based on a network based approach (adapted from [25], Fig. 1, blue dots in the right figure are left leaning users and red dots are right leaning users).

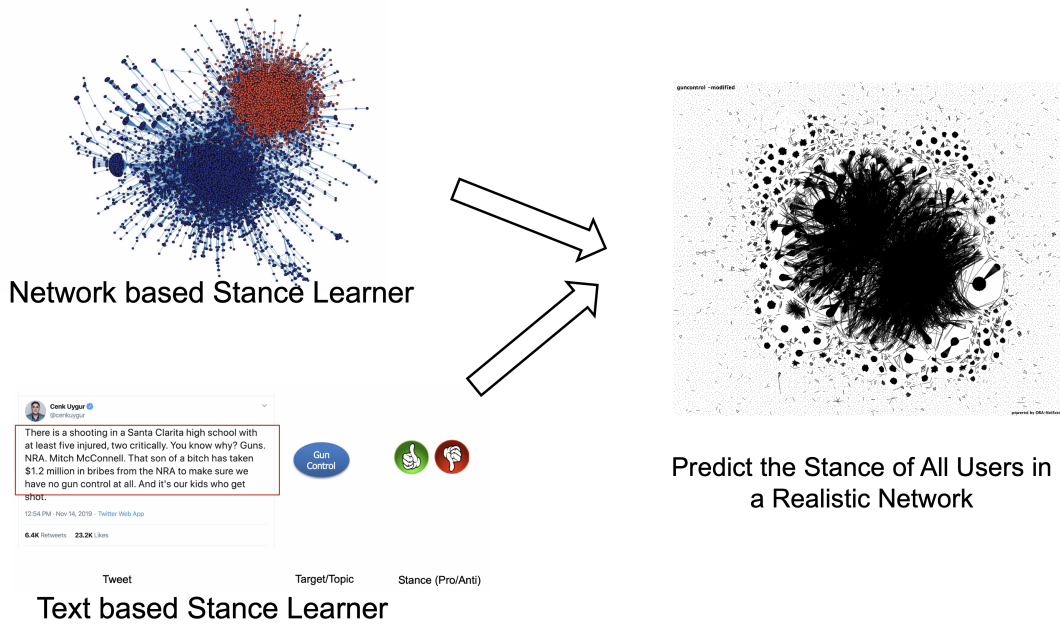


Figure 1.4: Proposed stance learning idea: The text-based approach (on the left, bottom) consider stance learning as a supervised machine learning task, whereas the network-based approach (on the left, top) commonly use a semi-supervised machine learning method. Both methods have their limitations. Because of the supervised nature, text approaches require a hand-labeled dataset which is expensive to build. The network-based approaches ignore the isolates (users not engaged in the interaction that was used to create the network), which could be a sizeable fraction of the dataset. By taking advantage of their strengths and applying the idea to a more realistic network that includes isolates (on the right), this thesis combines these two approaches.

1.1 Background

People express their opinion on different topics on social media platforms like Twitter and Facebook. Algorithms have been developed to automatically extract peoples opinion form these vast corpora of user-generated data. In particular, sentiment mining was an extremely popular field of research in the last decade [78]. However, sentiment mining aggregates users' opinions. In sentiment mining, we are interested in finding general perception of an entity. For example, the company 'Apple' may want to find about their brand perception i.e. whether they are generally discussed in a positive way or or a negative way. In such tasks, we first obtain a large set of data, e.g., by searching the social media platforms for the term 'Apple' and then obtaining a sentiment score for each data item. Finally, we average the scores obtained to get the overall perception. This approach of analysis is beneficial in tasks like learning about a brand and measuring a product's rating. However, there are other tasks, in which we are interested in information at a more granular level. For example, we might be interested in finding the perception of users about a party or a political candidate so that we can send targeted advertisements. Such tasks are better approached by evaluating opinions at the level of users. This user level aspect is further highlighted in an unrelated task, such as when the goal is to understand the view of users engaged in discussion on rumors. In discussing such controversial topics, people take sides and are invested in the discussions at various levels. As described by Kiesling et al. [60], there are three dimensions to analyzing conversations on social media a) Affect b) Alignment c) Investment. Sentiment mining stays at the level of affect. In stance learning, we consider affect and alignment. In conversations on social media, there is yet another aspect of 'Investment'. In this work, I tend to discount 'Investment' as the prime application of this work is to understand polarization on social-media about a controversial topic. When it comes to controversial topics, people take a side, i.e. either they are either pro or they are anti.

In this work, we focus in aligning users to better evaluate their stance. This approach closely follows the definition of Stance proposed bu Prof. Du Bois [31], who argues that stancetaking is the public act of simultaneously evaluating objects, positioning subjects and aligning with other subjects. This definition is illustrated in a Fig. 1.5 which is described as 'The stance triangle'. As shown in the figure, alignment (or dis-alignment) happens as users display similarity and difference with respect to their evaluations. As shown by Joseph et al. in Constance [59], in addition to a tweet, providing more context relevant information (e.g.more relevant tweets) improves the agreement between annotators who are labeling stance of users. Thus, it is natural to expect that other relevant interactions on social media would also improve the stance prediction accuracy.

There are many challenges in applying interactional stancetaking to improve stance prediction by aligning users. The primary challenge is the multiple ways people can interact on Twitter. The popular way is to 'tweet' which is a way to broadcast opinion. However, in addition to tweeting, one can retweet posts, like posts, reply to a post, make connections (like become a follower) and more. All these types of interactions have relevance in finding users stance and their alignment. My goal, in this thesis research, is to explore which of these interactions are useful for learning stance and how the learners that are trained on one type of interaction can be combined with others. As shown in Fig. 1.4, the current stance learning system important limitations. In this thesis, I intend to move from the current text based stance learning paradigm to a multi-

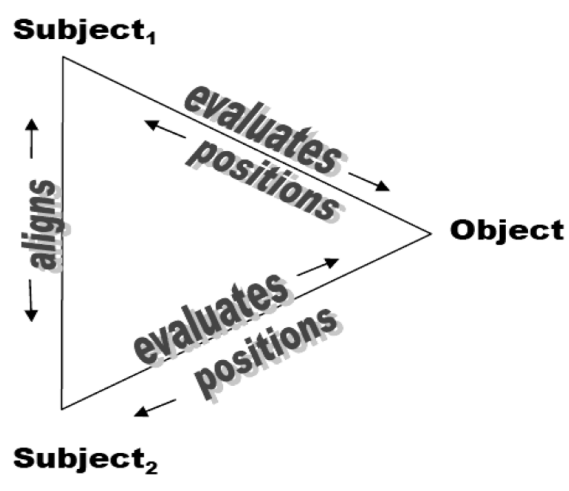


Figure 1.5: The stance triangle, adapted from Du Bois [31], page 163.

modal graph based stance learning approach that can use text as input, in addition to other kinds of Twitter interactions.

1.2 Prior Work

Prior work on Stance learning have appeared in primary three flavors a) Ideological leaning of Users on social media platforms [59, 91, 109] b) Agreement or disagreement of users about a topic in debates [49, 110, 111] c) Stance of connected users in communities [80]. I summarize recent contributions in each of these categories next. In addition, I also discuss machine learning models for stance mining, applications of stance learning e.g., in rumor identification. Lastly I describe the existing datasets for stance learning tasks.

1.2.1 Learning Stance from Social-Media data

Existing approaches on stance mining that uses Twitter posts can be categorized as one of the three types a) Single Tweet Single Target (STST) b) Single Tweet Multiple Targets (STMT) and c) Multiple Tweets Single Target (MTST). We summarize three existing datasets in Table 1.1.

Out of these three datasets, SemEval 2016 dataset contains tweets' text and stance labels for five different topics where each tweet has only one target (STST) [91]. This dataset was used in SemEval competition in 2016 (Task #6). The dataset obtained from Twitter contained text from on five controversial topics and was used in a SemEval, 2016 competition ². Many researchers used this dataset and tried many different types of algorithms to predict stance [5, 79, 126]. However, as reported in [91], none of them exceeded the performance achieved by a simple algorithm that uses word n-grams and word-embedding as features. Even neural-network models [4] that uses bi-directional conditional encoding did not perform better than simple n-grams

²<http://alt.qcri.org/semeval2016/task6/>

based models, perhaps indicating that the amount of labeled data was not sufficient to train the models.

More recently, two more text based stance datasets (ConStance and MultiStance) were released. In ConStance, the dataset contains multiple tweets given a target (MTST) which are four different politicians [59]. MultiStance is composed of tweets that have multiple targets (STMT) [109]. Researchers have also looked at different ways to train stance models. By analyzing temporal tweeting activity of politicians and the way issues can be framed on Twitter, [58] designed a weakly supervised model for learning stance. They suggested to use content, frames jointly, and temporal activity to build local models and combine them through Probabilistic Soft Logic.

Unlike text sentiment, stance is user based, and stances of users don't change frequently. Text content of a Tweet could be a useful stance indicator. However, using just one tweet fails to leverage the consistency in users' stance. Therefore, there is need to move from a single tweet based stance models to multiple tweets from a user (or the entire timeline) based models. ConStance dataset highlights the benefit of multiple tweets. Future methods should explore ways to find all relevant tweets of a user that are relevant to stance target and use them for estimating users' stance. This thesis is an attempt in that direction.

1.2.2 Learning Stance from Debates

Earlier work on stance learning revolved around debates [49, 110, 111]. Using a manually annotated corpus, Somasundaran and Wiebe [110] constructed an arguing lexicon to recognize stance in online debates and used an SVM classifier resulting in above 60% accuracy. The authors used data on debate posts in four domains 'Gun rights', 'Gay rights', 'Abortion' and 'Creationism'. Ozer et al. tried to cluster politically motivated users into communities [95]. They used structural balance theory to build a user-connectivity network using endorsements and tried three non-negative matrix factorization approaches to detect communities. Tu et al. proposed context-aware embeddings that attempt to use semantic relationships between users to preserve the diverse roles of interacting users [118]. Some researchers have also considered adding different types of constraints in models, e.g. [49] used author constraints (AC) in which two posts written by the same author for the same domain should have the same stance. In [48], the authors explored more constraints that include ideology constraints and user-interaction constraints. By modeling the problem as an optimization problem and solving it by integer linear programming (ILP), they report an improvement in accuracy in the range of 2-10%. Due to the short text and noisy nature of social media, in most cases, this line of research on debates can't be directly applied to social media data. However, as we show in the chapter 6, stance in debates and stance obtained from social media data are not independent and can very well be integrated. More details on this unifying approach is discussed in chapter 6 of this thesis.

1.2.3 Learning Stance Using Networks Data

Social theory of homophily and some work on understanding bias, partisanship and user-to-user sentiment clearly show the utility of user-to-user connections for stance learning. Lu et al. proposed BiasWatch, a bias propagation method to infer opinion bias of Twitter users [80].

Using a few seed hashtags, they identified other supporting and opposing hashtags using co-occurrence and information gain. Then they used this expanded set of hashtags to find partisan users. Finally, a label propagation approach based on content and retweeting similarity is used to estimate bias of rest of the users in the dataset. A limitation of this work is the step of finding features similarity between every user, which is very resource consuming and does not scale to large number of users.

On similar lines, Collopi et al. used content shared in Tweets to build a machine learning classifier to identify users aligned to Democrats and Republicans [23]. They separately analyzed two different networks a) Symmetric, where users follow back b) Non-symmetric where users don't follow back their follower. They used 1,683 Democrat users and 8,868 Republican users data as training set to predict the political orientation of users using linguistic features of the tweets. They observed that political homophily structure differs between Democrats and Republicans. For the Republicans, they found low levels of political homophily, in contrast to high levels for the Democrats. Ozer et al. tried to cluster politically motivated users in to communities [95]. They used structural balance theory to build a user-connectivity network using endorsements, and tried three non-negative matrix factorization approaches to detect communities. Their model merged user content and connectivity information based on endorsements.

There is also a line of research on signed networks [73]. Wang et al. proposed SiNE to learn signed network embedding [123]. SiNE learns a low dimensional representation of nodes while preserving the ideas of structural balance theory. In [127], authors build people to people network. Gurini et al. used sentiment based community detection for user recommendation [46]. West et al. uses social network structure to understand user-to-user sentiment analysis [127]. Choi et al. tried document level sentiment to infer opinions [22]. The authors used integer linear programming to jointly model sentiment, entity factions and constraints (homophily, social balance and reciprocity). In our work, we showed that users attributes are useful for grouping users [67] but we did not consider users' sentiment. The ideas from the work on signed network is applied in chapter 6 in which we use different networks to infer stance.

1.2.4 Machine Learning Approaches to Stance Mining

Deep neural networks (DNN) have shown great success in many fields [51]. Researchers have used DNNs for various NLP tasks like POS tagging, named entity recognition [24]. In DNNs, our prime interest is in multi-modal techniques [92] that can combine multiple types of features extracted from social media interactions. On stance learning, [136] used an average of word vectors from each tweet as an input to their LSTM model. They trained their LSTM model using branches of twitter conversations where mean-of-words is used as input and stance label is used as output. Sequential classifiers like LSTMs are biased to use prior inputs to predict outputs. For example, an LSTM can be used to predict the next word for the partial sentence 'The flower is ...', given a large amount of training data the model estimate that 'beautiful' or 'fresh' is more likely to the last word. However, when it comes to tasks like stance classification in threaded discussions, each reply is made against another post (and not necessarily to the series of earlier posts). Does the sequential nature of responses make it more suitable for an LSTM like model? Or would a model that can learn the differences between the source and reply tweets are more ideal for stance classification? These are some of questions that we consider in chapter 3.

1.2.5 Multi-modal Stance Mining – Models to Combine Different Types of Interactions

Use of multi-modal interactions for stance prediction is non-trivial. Because different interactions result in text (from posts) or graphs or a combination of both, multi-modal stance learning is challenging. Though there is limited research in combining different modalities for stance learning, there is a quite some work on multi-modal machine learning [8, 92]. A common technique is to embed all modalities in a continuous representation space. A good stance learner model should be able to use all text and graphs data. For converting text to embedding space, multiple ways to encode sentences (like [63]) are available. To convert graphs to a continuous representation space, several techniques have been proposed recently. For example Perozzi et al. [97] introduced DeepWalk, a model to learn latent vector representations for nodes that encode social relations by using short random walks. Therefore, using a continuous representations space for all types of interactions, we can build a joint model based on inputs of different modalities. We explored this idea in chapter 4, in which converted users' attributes to user-to-user connections based on a similarity metric [67]. Chapter 5 and chapter 6 propose new methods to jointly train classifiers based on different modalities.

1.2.6 Applications of Stance Learning in Rumor and Misinformation Identification

This task of stance learning got popularity when it was realized that stance could be used to identify fake news³ and rumors⁴. Finding misinformation on social media platforms has been an active area of research in recent years. In this type of work, the researchers use the content of a post to determine the veracity typically uses language features. For example, Rashkin et al. [100] discuss language characteristics of real news compared to satire, hoaxes, and propaganda. They found that Linguistic Inquiry and Word Count (LIWC) and sentiment lexicon features are useful to understand the differences between fake and more reliable digital news sources. This can be further extended by including information present in more reputable sources like using knowledge graphs to find factual discrepancies [50]. When network features are available (e.g. when an article is posted on social media), researchers have shown that including social-network features in addition to content features, outperform lexical based models [121].

There is another line of research on rumor detection that uses stance in the reply posts. This kind of work was initiated by the PHEME project⁵ and was popularized in a SemEval 2018 competition (task 8)⁶. The task involves predicting stance ('supporting', 'denying', 'commenting' and 'querying') in replies to rumor posts on Twitter and is described in [137] and [138]. A number of researchers used this dataset and proposed many algorithms. For this dataset, [28] proposed an LSTM that uses branch of conversation trees to classify stance in reply posts, and [136] used and compared many different sequential classifiers. My work in chapter 3 extends

³<http://www.fakenewschallenge.org/>

⁴<http://alt.qcri.org/semeval2017/task8/>

⁵<https://www.pheme.eu/>

⁶<http://www.aclweb.org/anthology/S17-2006>

this thread of research by using different ways to learn stance from social media conversations [68].

In summary, in most NLP based research described above that used social-media data, the stancetaker (as in author of text) are not explicitly considered, and aligning authors was not an important goal. However, on controversial topics in which authors take clear stance (e.g. in polarized communities), authors alignment is important as asserted in [31]. Therefore there is a need to extend the text-based stance learning to use multiple posts from the same user. My research explores this critical aspect of stance learning. One modality that is rarely used for stance mining is the conversations e.g., replying and quoting on Twitter. Conversation threads can reveal authors' alignment with other users based on the stance they take in replying to other's posts. Yet another problem with text based stance learner is the availability of datasets. Most existing dataset are on a few hand picked topics. As we know, discussions on topics change a lot on social media. There is need to build stance learning models that do not need hand-labeled examples for training but rather can be trained using weak supervision. A good research trying to overcome this challenge is BiasWatch ([80]), in which the authors use a few hand picked hashtags. Even in this line of work, there is a need to build more robust systems that could work with noisy hand-picked hashtags. Furthermore, various interaction networks on Twitter (e.g. follower, mentions, likes) would result in graph structured data which needs networks based machine learning models. In this thesis, in chapters 3, 4, 5 and 6, we propose joint models that combines the various sources of stance information (see Fig. 1.4). To combine different modalities of information (e.g., users' follower graphs and users' likes), one can embed features from all information sources into one continuous representation space (chapter 3). But there could be better ways to learn from modalities where different modalities bring complementary information (chapters 4 and 5). We expect that the methods proposed for multi-modal stance learning techniques in this thesis could be useful in many areas. For example, we can apply these models to analyze polarized communities.

1.2.7 Stance Datasets

There are two datasets on stance learning that were built in prior research (by other researchers) which I use in my research. These datasets are described in Tbl. 1.2 and Tbl. 1.4. Because the focus of this work is in weak supervision, the two labeled datasets are used only for the validation purpose and not for training of the models (unless specified otherwise in the chapters). For semi-supervised methods to work, we often take benefit of the structure of the data itself in which case, having more data helps. With this goal, we also augment theses datasets with additional tweets from the users in these datasets i.e. for each user in the dataset, we obtain their entire timeline. Because we retrieved the users' timeline recently, there are users that were in the original dataset but are not present in our dataset (e.g. some users got suspended or deleted their accounts). That is why, the summary statistics in tables (shown next) could be different from the statistics of the datasets we describe in the later chapters.

In addition to the two existing datasets, we also use the dataset described in Tbl. 1.6 for applying stance mining for rumor detection (in chapter 3). Moreover, we built a new dataset that has stance labels (as in favor or denial) for replies in conversation threads (see sec. 1.2.7). The labeling of conversations (as in favoring vs denying) is different from the labeling of users'

stance (as pro/con a topic) as we highlight in chapter 2.

I summarize these datasets next.

Existing Datasets

As explained in Fig. 1.2, existing approaches on stance learning that uses Twitter data can be categorized as one of the three types a) Single Tweet Single Target (STST) b) Single Tweet Multiple Targets (STMT) and c) Multiple Tweets Single Target (MTST). We summarize the three existing datasets in Table 1.1.

Table 1.1: Manually labeled datasets created in prior research. These are of three types. a) Single Tweet Single Target (STST) b) Single Tweet Multiple Targets (STMT) and c) Multiple Tweets Single Target (MTST).

Reference	Type	Targets/Topics	Count (total/train/dev/test)
SemEval 2016 [91]	STST	Atheism, Climate Change is concern, Feminist Movement, Hillary Clinton, Legalization of Abortion	4163/ 2914/ NA/ 1249
MultiStance [109]	STMT	Donald Trump, Hillary Clinton, Bernie Sanders, Ted Cruz	4455/ 3119/ 446 /890
Constance [59]	MTST	Donald Trump, Hillary Clinton, Neutral	1130/ 562/ 250/ 318

SemEval 2016 [91] contains text and stance labels for five topics (see table. 1.2). SemEval 2016 Stance Dataset (Single Text Single Target) In this section, we describe a human-labeled dataset that was built in prior research [91] and was used for a competition in SemEval 2016. The dataset is summarized in Tab. 1.2.

Table 1.2: SemEval 2016 Stance Dataset

Topic	Train	Test
Atheism	513	220
Climate Change	395	169
Feminist Movement	664	285
Hillary Clinton	689	295
Legality of Abortion	653	280

This dataset, which I think is the most popular dataset for stance learning, has text data on five topics that are ‘Atheism’, ‘Climate Change’, ‘Feminist Movement’, ‘Hillary Clinton’ and ‘Legality of Abortion’. This dataset has no development/validation set.

Dataset on stance labels (pro/con about a topic) of users along with their connections Lu et al. creates a dataset of Twitter users along with their bias on three different topics [80]. We summarize the users in Table 1.3 and their labeled stance in Table 1.4.

Table 1.3: Topics summary

Events	Users	Tweets	RTUsers	Endtags
Guncontrol	70387	117679	15635	5505
Obamacare	67937	123320	14807	7376
Abortion	111463	173236	26818	9784

Table 1.3 summarizes Biaswatch topics. In the table, RT users mean the number of users that were retweeted in data and the information is used create the user-retweet graph. Similarly, endtags show the number of unique hashtags used at the end of tweets, and are used to build the user hashtags networks. We use endtags as prior research has shown that hashtags that appear at the end convey stance signal [34].

Table 1.4: Labeled users summary

Events	Neutral	Pro	Anti	Total
Gun-control	60	156	288	504
Obamacare	33	108	363	504
Abortion	55	169	280	504

Dataset on stance labels (favor vs denial) for replies in rumour threads This dataset was created as a part of the Pheme project which aims to find and verify rumors shared on social-media platforms [137, 138]. The dataset consists of Twitter conversation threads on nine different events and contains three types of annotations. Each thread is labeled as either rumor or non-rumor. Rumors are annotated as true, false or unverified. For a subset of the true rumors, we also have stance labels for each reply in the threaded conversations. The stance labels are ‘supporting’, ‘denying’, ‘commenting’ and ‘querying’.

A new dataset on stance in conversations

In addition to using some of the existing datasets, we also developed a new dataset for this thesis. This was needed as no existing datasets have stance labels both for users’ stance and stance in conversations. The new dataset dataset is composed of stance labels of replies to the posts on controversial topics of ‘Iran Deal’. ‘Santa Fe Shooting’, ‘Student Marches’, and some general conversations not representing any specific topic. In addition to stance in replies, we are also creating labels for users’ stance about these topics. In a way, this is the first and only dataset that

Table 1.5: Stance labels for Tweets in the conversations

Event Name	Supporting	Denying	Querying	Commenting
Charlie Hebdo (CH)	239	58	53	721
Sydney siege (SS)	220	89	98	700
Ferguson (FG)	176	91	99	718
Ottawa shooting (OS)	161	76	63	477
Germanwings-crash (GC)	69	11	28	173
Putin missing (PM)	18	6	5	33
Prince Toronto (PT)	21	7	11	64
Ebola Essien (EE)	6	6	1	21

has stance labels for users as a well as conversations, and therefore, could be considered the first multi-modal stance mining dataset. The dataset is summarized in Tbl. 1.6.

Table 1.6: Distribution of labels across different events.

	General Terms	Iran Deal	Santa Fe Shooting	Student Marches
Comment	656	293	246	153
Explicit Denial	521	350	471	253
Implicit Denial	202	116	116	49
Explicit Support	138	118	85	47
Implicit Support	415	327	279	215
Queries	88	42	21	19

Table 1.6 presents the label distribution for the different events. We observe that the labeled dataset is skewed towards denials as. This is intentional as we tried to overcome the limitation of the earlier rumor threads based stance dataset described in the earlier paragraph (see Tbl. 1.5). The process of constructing the dataset and additional details are available in chapter 2.

1.3 Contributions

Here we highlight the main contributions of this thesis. This thesis aims to resolve the three crucial challenges of stance mining:

1. How do we train stance-learning models on new topics with minimal labeling effort?
2. How can we use multiple interaction modalities for stance mining?
3. How to leverage users' networks for stance prediction?

On social-media discussion, topics change fast, and new issues emerge, making it difficult to reuse prior labeled data. This leads to challenge 1, i.e., how do we train stance-learning

models on new topics with minimal labeling effort? As we show in this thesis, simple artifacts of social networks like hashtags can carry noisy signal about the stance of the users who have used those hashtags. The challenging part is to extract the signal from noise. We proposed a semi-supervised learning approach that uses two or more models and plenty of unlabeled social-media data to build stance classifiers (chapters 4 and 5). In the proposed approach, multiple stance classifiers are jointly trained, which results in reducing the effect of noise and improving the performance of the classifiers over the training iterations. As the models get trained using only the stance given by a few seed hashtags, the approach is very flexible and works well on new topics.

Users' opinions are evident in different types of interactions, e.g., tweeting, retweeting, or liking. Though having multiple interactions allows different ways to learn and predict users' stance, and it is not apparent how we can correctly use the various interaction modalities for stance mining. This leads to problem 2. As discussed in the last paragraph, we proposed a semi-supervised method for jointly training models based on different interactions. The proposed method could learn from complementary information in different interactions to train better classifiers. Though in this work, we only considered three interactions comprised of two networks (namely user-hashtags and user-retweets) and one text classifier (based on users' tweets or user-to-user conversations), the simplicity of our approach allows extending the method to other interactions.

The current approaches to stance learning ignore important network factors, e.g., a person's preference can also be known from his friends' preferences. Since stance is a public act, the actor's social network position should matter. This is described as the third challenge on how to leverage users' networks for stance prediction. We use the network alignment as one training signal to train the stance classifiers (chapters 3, 4, and 5 uses networks). One of the classifiers used for jointly training the model is a network-based label propagation classifier. The label propagation model effectively utilizes the similarity in the connected nodes to improve the overall stance classification performance.

Based on experiments on six topics, we estimate that with 2-4 hashtags as weak labels, the user' stance classifiers could reach an accuracy of over 80%. For learning the stance in conversations, which is a more challenging problem, we achieve accuracy over 70%. More importantly, the proposed methods result in inductive classifiers that could be used with newer data without a need to retrain the models. Furthermore, as demonstrated in chapter 6, the stance learners could be applied to problems that are relevant in society, such as the spread of misinformation.

I summarize the main contribution of this thesis as:

Learning the stance of social media users is challenging as topics change, and new discussion topics emerge. By intelligently extracting data from the different interaction modalities, and jointly training the models, we can build usable stance learners that require minimal labeling effort..

1.4 Organization of this Thesis

This thesis is divided in seven chapters and two additional appendix items.

1. Introduction
2. A New Stance Dataset and Some Baseline Models
3. Learning Users' Stance by Combining Users' Networks and Users' Posts by Creating Virtual Connections (Extension of my SBP-Brims 2018 paper [67])
4. Co-Training on Social Networks: A Joint Network Label Propagation and Text Classification Approach for Stance Mining
5. A Joint Network and Text based Model for Learning Stance in Conversations
6. Models of Tree Structured Conversations for Predicting Stance and Rumor Veracity (Extension of the work published at ACL 2019 [68])
7. Conclusions and Future Work
8. Appendix 1: Better ways of collecting Twitter data: What to Track on the Twitter Streaming API? A Knapsack Bandits Approach to Dynamically Update the Search Terms (Extension of the work published at ASONAM 2019 [69])
9. Appendix 2: A users guide for labeling stance in conversations

Chapter one (this chapter) introduces this thesis work, provides a a background along with motivations. Here, I also describe the prior work on stance mining. In addition, I also introduce the common datasets that I have used throughout the thesis.

In the second chapter, I describe a new dataset that we created for this research along with many baseline models. Recently, there is a renewed excitement in the field of stance mining as we see new models attempting to improve the state-of-the-art. However, for training and evaluating the models, the datasets used are often small. Additionally, these small datasets have uneven class distributions, i.e., only a tiny fraction of the examples in the dataset have favoring or denying stances, and most other examples have no clear stance. Because of this, models trained on one event do not generalize to other events. In this chapter, we create a new dataset by labeling stance in responses to posts on Twitter (both replies and quotes) on controversial issues. To the best of our knowledge, this is currently the largest human-labeled stance dataset for Twitter conversations with over 5200 stance labels. The dataset described in this chapter is used in later chapters for training and verifying the performance of different stance learning methods.

The third chapter provides a way to learn stance from users' networks and users' attributes. In most social network studies, in which group a user is, is determined as a function of explicit ties. For example, given a set of random walks through the network, it is possible to learn a vector for each node which contains a latent representation of the node. These latent representations have useful properties that can be easily exploited by statistical models for tasks like identifying groups and inferring implicit links. However, most existing representation learning methods ignore node attributes. In many cases, there is a rich body of information and events associated with nodes that also can be used for node clustering and to infer ties. In this chapter, I propose *People2Vec*, an algorithm to learn representations that takes into account proximity between users due to their attributes and social media activities and apply it to learning stance of social-media users.

In the fourth chapter, I extend the idea of using multiple interactions (as tweeting, retweeting,

using hashtags) to new training paradigm where we jointly train different stance classifiers with weak supervision. This chapter tackles the two important challenges of stance mining: 1) Discussion topics change fast, and new issues emerge. How do we train stance-learning models on new topics with minimal labeling effort? 2) Users' opinions are evident in different types of interactions. How do we learn from multiple types of interactions effectively? In this research, we tackle both these challenges by extending the co-training approach to jointly train two classifiers: 1) a label propagation model on networks, and 2) a text classification model using text features. We use the weak stance signals given by two to four labeled hashtags for training the models. Though training examples obtained using hashtags are noisy, co-training effectively handles the noise, thereby enabling stance mining on new topics with minimal labeling effort.

In the fifth chapter, I extend the idea developed in chapter four to train stance classifiers (for users' posts and networks), to include users' conversations (e.g. replies). Prior research to predict the stance of social-media users on controversial topics has been broadly explored as two separate threads of research: 1) learning stance of users based on their social media posts (as in Pro/Con about a topic), and 2) learning stance taken in conversations while replying (as in favoring and denying a post) to social-media posts. Though these threads represent different aspects of stacetaking behaviour on social-media forums, it is natural to ask if they are two faces of the same phenomenon, and if so, what would be a unifying approach. As we would expect, given a discussion on a controversial topic, a user is more likely to deny a post (while replying) to a post of another user who has the opposing stance. To better leverage this pattern, we propose a joint model for training stance classifiers that learn stance from : 1) users' networks, 2) users' conversations.

In the sixth chapter, I use stance in conversations (as favor vs denial) for detecting rumors on Twitter. Learning from social-media conversations has gained significant attention recently because of its applications in areas like rumor detection. In this chapter, I propose a new way to represent social-media conversations as binarized constituency trees that allows comparing features in source-posts and their replies effectively. Moreover, I propose to use convolution units in Tree LSTMs that are better at learning patterns in features obtained from the source and reply posts. Our Tree LSTM models employ multi-task (stance + rumor) learning and propagate the useful stance signal up in the tree for rumor classification at the root node. The proposed models achieve state-of-the-art performance, outperforming the current best model by 12% and 15% on F1-macro for rumor-veracity classification and stance classification tasks respectively.

In the seventh chapter, I conclude, provide limitations of this work and suggest directions for the future research.

In Appendix, I provide a way to get more relevant Twitter data using the Twitter streaming API. As the second item in appendix, I provide the user manual used for labeling stance in conversations (details in chapter 2).

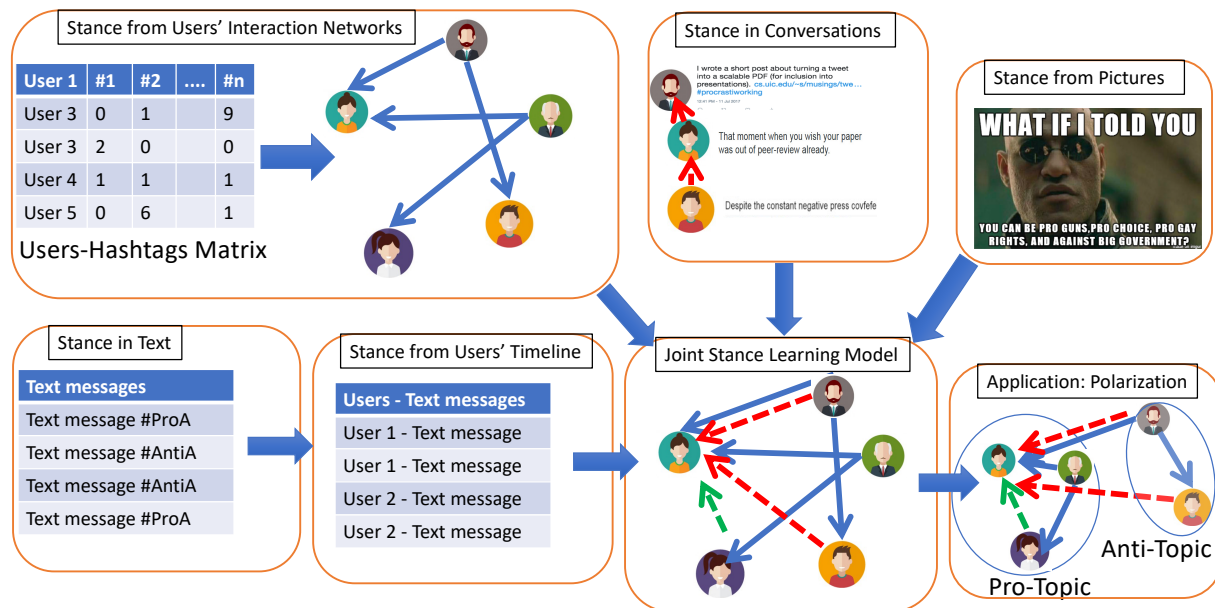


Figure 1.6: Summary of the research conducted in this thesis. The different colors of lines show different interactions. Solid lines indicate explicit relationships (like retweets) and dashed lines indicate inferred relationships. The goal of this thesis is to provide a framework to effectively utilize the multi-modal interactions on social-media platforms like Twitter for stance learning. Stance from pictures and application to community polarization are left for future work.

Chapter 2

A New Stance Dataset and Some Baseline Models

2.1 Introduction

People express their opinions on blogs and other social media platforms. Automated ways to understand the opinions of users in such user-generated corpus are of immense value. It is especially essential to understand the stance of users, which involves finding people’s opinions on controversial topics. Therefore, it’s not surprising that many researchers have explored automated ways to learn stance given a text [49]. While learning stance from users’ individual posts have been explored by several researchers [59, 91], there is an increased interest in learning stance from conversations. For example, as we show in Fig. 6.1, a user denies the claim made in the original tweet. This kind of stance learning has many applications, including insights into conversations on controversial topics [38] and finding potential rumor posts on social-media [6, 136, 137]. However, the existing datasets used for training and evaluating the stance learning models limit the broader application of stance in conversations.

The existing research on stance in conversations has three significant limitations: 1) The existing datasets are built around rumor events to determine the veracity of a rumor post based on stance taken in replies [137]. Though useful for rumor detection, this does not generalize to non-rumor events [17], 2) The existing datasets focus primarily in direct responses and do not take into account quotes. This is critical as quotes have been gaining prominence since their introduction by Twitter in 2015, especially in the context of political debates [40], 3) The existing datasets have uneven class distributions, i.e., only a small fraction of the examples in the dataset have supporting and denying stances, and most other examples have no clear stance. These unbalanced classes lead to poor learning of denying stance (class) [68]. The denying class is expected to be more useful for downstream tasks like finding an antagonistic relationship between users. Therefore there is a need to build a new dataset that has more denying stance examples.

To overcome the above limitations, in this research, we created a new dataset by labeling the stance in replies (and quotes) to posts on Twitter. To construct this dataset, we developed a new collection methodology that is skewed towards responses that are more likely to have a denial

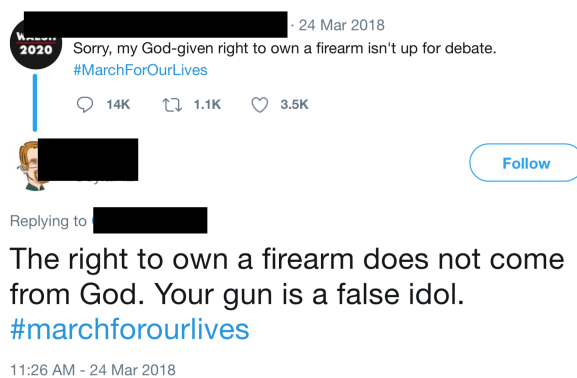


Figure 2.1: When we reply on Twitter, sometimes we also support or deny others claims. For example, in the conversation shown above, a user denies the claim made in the original tweet. In this research, we build a new dataset to learn the language pattern that users’ employ while taking a stance (support vs deny). This dataset could be used to develop automated methods to infer the stance in replies (and quotes).

stance. This methodology was applied across three different contentious events that transpired in the United States during 2018. We also collected an additional set of responses without regard to a specific event. We then labeled a representative sample of the response-target pairs for their stance. Focusing on the identification of denial in responses is an essential step for the identification of tweets that promote misinformation [137, 138] and also to estimate community polarization [38]. By leveraging these human-labeled examples, along with more unlabeled examples on social-media, we expect to build better systems for detecting misinformation and understanding of polarized communities.

To summarize, the contribution of this work is fourfold:

1. We created a stance dataset (target-response pairs) for three different contentious events (and many additional examples from unknown events). To the best of our knowledge, this is currently the largest human-labeled stance dataset on Twitter conversations with over 5200 stance labels.
2. To the best of our knowledge, this is the first dataset that provides stance labels for Quotes (others are based on replies). This provides a new opportunity to understand the use of quotes.
3. The denial class is the minority label in existing datasets built in a prior research [137] and is the most difficult to learn, but is also the most useful class for downstream tasks like rumor detection. Our method of selecting data for annotation results in a more balanced dataset with a large fraction of support/denial as compared to other stance classes.
4. We introduce two new stance categories by distinguishing between explicit and implicit non-neutral responses. This can help the error analysis of trained classifiers as the implicit class, for either support or denial, is more context dependent and harder to classify.

This paper is organized as follows. We first discuss the related work and then describe our approach to collect the potential tweets to label in ‘Dataset Collection Methodology’. As the sample that can be labeled is rather small (because of budget limitations) compared to the en-

tire available dataset, we discuss the sample construction procedure for annotation. Then, we describe the annotation process and the statistics of the dataset that obtained as a result of annotation in section ‘Annotation Procedure and Statistics’. Next, we present some baseline models for stance learning and present the result. Finally, we discuss our results and propose future directions.

2.2 Related Work

Topics on learning stance from data could be broadly categorized as having to do with: 1) Stance in posts on social media, and 2) Stance in Online Debates and Conversations. We next describe prior work on these topics.

2.2.1 Stance in Social-Media Posts

Mohammad et al. [91] built a stance dataset using Tweets of several different topics, and organized a SemEval competition in 2016 (Task 6). Many researchers [5, 79, 126] used this dataset and proposed algorithms to learn stance from data. However none of them exceeded the performance achieved by a simple algorithm [91] that uses word and character n-grams, sentiment, parts-of-speech (POS) and word embeddings as features. The authors used an SVM classifier to achieve 0.59 as the mean f1-macro score. While learning stance from posts is useful, the focus of this research is stance in conversations. Conversations allow a different way to express stance on social media in which a user supports or denies a post made by another user. Stance in a post is about authors’ stance on any topic of interest (pro/con), in contrast, stance in conversation is about stance taken when interacting (replying or quoting) with other authors (favor/deny). We describe this in detail in the next section.

2.2.2 Stance in Online Debates and Conversations

The idea of stance in conversations is very general and its research origin can be traced back to identifying stance in online debates [110]. Stance in online debates have been explored by many researchers recently [49, 108, 111]. Though stance-taking by users on social-media, especially on controversial topics, often mimic a debate, social-media posts are very short. An approach of stance mining that combines machine-learning to predict stance in replies – categorized as ‘supporting’, ‘denying’, ‘commenting’ and ‘querying’ – to a social media post is gaining popularity [135, 137]. Prior work has confirmed that replies to a ‘false’ (misleading) rumor are likely to have replies that deny the claim made in the source post [138]. Therefore, this approach is promising for misinformation identification [6]. However, the earlier stance dataset on conversations was collected around rumor posts [137], and contains only replies, and has relatively few denials. Our new dataset generalizes this approach and extends it to quotes-based interactions on controversial topics. As described, this new dataset is distinct as: 1) it distinguishes between ‘replies’ and ‘quotes’, the two very different types of interaction on Twitter, 2) it is collected in way to get more ‘denial’ stance examples, which was a minority label in [135], and 3) it is collected on general controversial topics and not on rumor posts.

2.3 Dataset Collection Methodology

Figure 2.2 summarizes the methodology developed to construct the datasets that skews towards more contentious conversation threads. We describe the steps in details next.

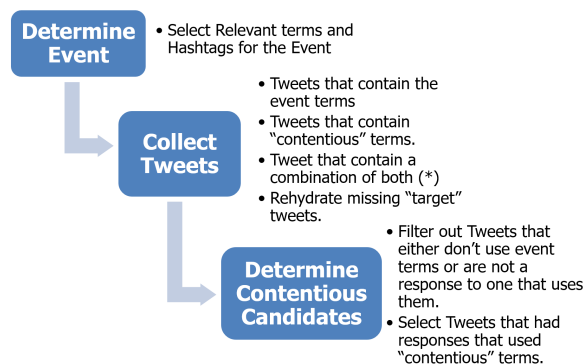


Figure 2.2: Methodology developed for the collection of contentious tweet candidates for a specific event.

The first step requires finding the event related terms that could be used to collect the source (also called target) tweets. Additionally, as the focus is on getting more replies that are denying the source tweet, we use a set of contentious terms used to filter the responses made to the source tweets.

2.3.1 Step 1: Determine Event

The collection process centered on the following events.

- **Student Marches:** This event is based on the ‘March for Our Lives’ student marches that occurred on the 24 of March of 2018 in the United States. Tweets were collected from March 24 to April 11 of 2018. The following terms were used as search queries: #MarchForOurLives, #GunControl, Gun Control, #NRA, NRA, Second Amendment, #SecondAmendment.
- **Iran Deal:** This event involves the prelude and aftermath of the United States announcement of its withdrawal from the Joint Comprehensive Plan of Action (JCPOA), also known as the "Iran nuclear deal" on May 8, 2018. Tweets were collected from April 15 to May 18 of 2018. The following terms were used as search queries: Iran, #Iran, #IranDeal, #IranNuclearDeal, #IranianNuclearDeal, #CancelIranDeal, #EndIranNuclearDeal, #EndIranDeal.
- **Santa Fe Shooting:** This event involves the prelude and aftermath of the Santa Fe School shooting that took place in Santa Fe, Texas, USA in May 18, 2018. Tweets were collected from May 18 to May 29 of 2018. For this event, the following terms were used as search queries: Gun Control, #GunControl, Second Amendment, #SecondAmendment, NRA, #NRA, School Shooting, Santa Fe shooting, Texas school shooting.

- **General Terms:** This defines a set of tweets collected that were not from any specific event, but are collected based on responses that contain the contentious terms described next. Tweets were collected from July 15 to July 30 of 2018.

The set of contentious terms used across all events are divided in 3 groups: hashtags, terms and fact-checking domains:

- **Hashtags:** #FakeNews, #gaslight, #bogus, #fakeclaim, #deception, #hoax, #disinformation, #gaslighting.
- **Terms:** FakeNews, bull**t, bs, false, lying, fake, there is no, lie, lies, wrong, there are no, untruthful, fallacious, disinformation, made up, unfounded, insincere, doesnt exist, misrepresenting, misrepresent, unverified, not true, debunked, deceiving, deceitful, unreliable, misinformed, doesn't exist, liar, unmasked, fabricated, inaccurate, gaslight, incorrect, misleading, deception, bogus, gaslighting, mistaken, mislead, phony, hoax, fiction, not exist.
- **URLs:** www.politifact.com, www.factcheck.org, www.opensecrets.org, www.snopes.com.

2.3.2 Step 2: Collect Tweets

Using Twitter's REST and the Streaming API we collected tweets that used either the event or contentious terms (as described earlier). If the target of a response is not included in the collection, we obtained it from Twitter using their API.

2.3.3 Step 3: Determine Contentious Candidates

A target-response pair is selected as potential candidate to label if the target contains any of the listed event terms and the response contains any of the contentious terms. If urls are in the tweet, they are matched at the domain level by using the *urllib* library in Python. For 'General Terms' event collected pairs based solely on the responses regardless of the terms used in the target.

To reduce the sample size, we filtered the tweets on some additional conditions. We only used the responses that were identified by Twitter to be in English and excluded responses from a user to herself (as this are used to form threads). In order to simplify the labeling context, we also excluded responses that included videos, or that had targets that included videos and limited our sample set to responses to original tweets. This effectively limits the dataset to the first level of the conversation tree.

The above steps resulted in a dataset which can potentially be labeled. We show the distribution of this dataset in Tab. 2.1. Because this set is large, we developed a method to a retrieve a smaller sample for labeling. We describe this sample construction method next.

2.4 Sample Construction for Annotation

We sought to design a sample that was representative of the semantic space observed on the responses across the different events. For this purpose we encoded the collected responses via Skip-Thought vectors [63], to obtain an a priori semantic representation. The Skip-Thought

Table 2.1: Distribution of relevant tweet pairs by response type that could be labeled.

Event	Replies	Quotes
Student Marches	23314	8321
Santa Fe Shooting	24494	11825
Iran Deal	21290	14939
General Terms	3756269	2540084

model is trained using a large text dataset such that the vector representation of the text encodes the meaning of the sentence. To generate vectors, we use the pre-trained model shared by the authors of Skipthought¹. The model uses a neural-network that takes text as input and generate a 4800 dimension embedding vector for each sentence. Thus, on our dataset, for each response in Twitter conversations, we get a 4800 dimension vector representing the semantic space.

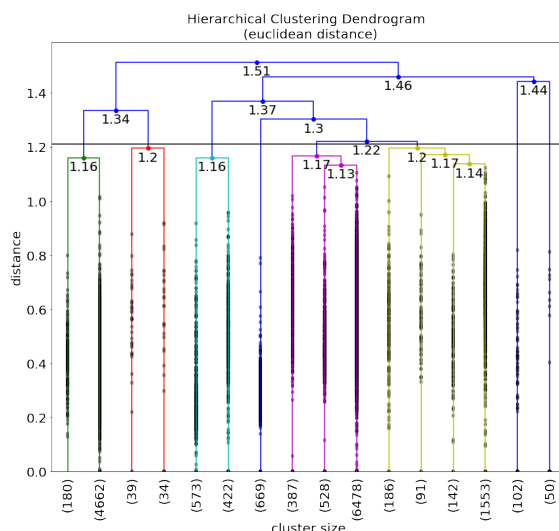


Figure 2.3: Dendrogram derived for the Student Marches event. Horizontal line describes the maximum cophenetic distance used when determining the final cluster labels. Further bifurcations of the dendrogram where replaced with dots in order to avoid clutter.

To obtain a representative sample of the semantic space, we applied a stratified sampling methodology². The strata were determined by clustering the space via hierarchical clustering methods using a 'average' linkage algorithm and a euclidean distance metric. It is important to note that given the difficulty of assessing clustering quality on such high-dimensional spaces (over 4k dimensions), we first reduced the space to 100 dimensions via Truncated Stochastic Value Decomposition [131]. Figure 2.3 presents the derived dendrogram and the optimal number

¹<https://github.com/ryankiros/skip-thoughts>

²Stratified Sampling is a sampling method that divides a population in exhaustive and mutually exclusive groups which can reduce the variance of estimated statistics.

of clusters selected for the Student Marches event, a similar analysis was done for the other events. The relevant hyper-parameters used were determined by evaluating the final clustering quality based on the resulting cophenetic correlation [105]. It is important to note that the number of clusters selected was higher than the optimal, as our main purpose is to get a thorough partition of the semantic space.

A two level stratified scheme was utilized, with the second level being the type of response. This means that the percentage of Quotes and Replies within each stratum were maintained. Finally, we decided to under-sample, by a factor of two, the responses to verified accounts so that our final sample has more interaction between regular Twitter users. The final sample distribution by response type is presented in table 2.2.

Table 2.2: Distribution of relevant tweet pairs by response type. Notice that these terms tend to be used more frequently in direct replies.

Event	Replies	Quotes
Student Marches (SM)	293	443
Santa Fe Shooting (SS)	609	609
Iran Deal (ID)	508	738
General Terms (GT)	1476	544

Figure 2.4 presents a 3-dimensional representation, obtained via Truncated Stochastic Value Decomposition, of the semantic space observed for the responses in the General Terms event and the derived sample. A similar clustering pattern is observed on other events as well. Notice that the sample covers fairly well the observed semantic distribution, especially when compared with simple random sampling.

2.5 Annotation Procedure and Statistics

Recent work on stance labeling in social media conversations has centered on identifying 4 different positions in responses: agreement, denial, comment, and queries for extra information [99, 138]. We introduced two extra categories, by distinguishing between explicit and implicit non-neutral responses. The former refers to responses that include terms that explicitly state that their target is wrong\right (e.g. ‘That is a blatant lie!’). The implicit category on the other hand, as its name implies, correspond to responses that do not explicitly mention the stance of the user, but that, given the context of the target, are understood as denials or agreements. These are much harder to classify, as they can include sarcastic responses.

The annotation process was handled internally by our group and for this purpose we developed a web interface for each type of response (see Fig. 2.5 for replies, and Fig. 2.6 for quotes). Each annotator was asked to go through a tutorial and a qualification test to participate in the the annotation exercise. The annotator is required to indicate the stance of the response (one of the six options in the list below) towards the target and also provide a level of confidence in the label provided. If the annotator was not confident in the label, then the task was passed to another

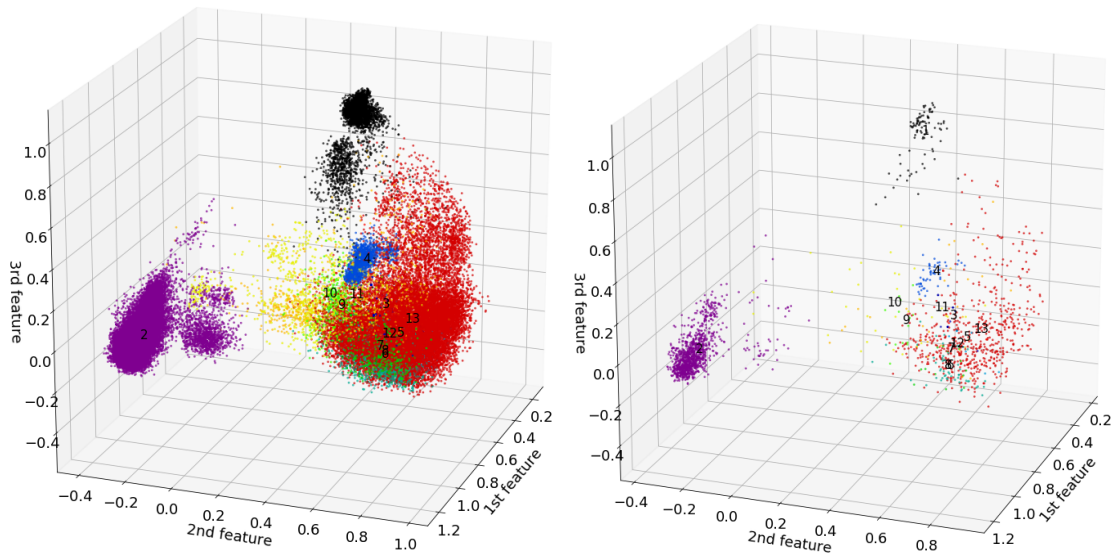


Figure 2.4: 3-dimensional representation, obtained via Truncated Stochastic Value Decomposition, of the skip-thought vector representation for the responses in the General Terms event. The top figure corresponds to the collected universe and the bottom to the derived sample. Similar distributions and clustering behavior is observed on other events.

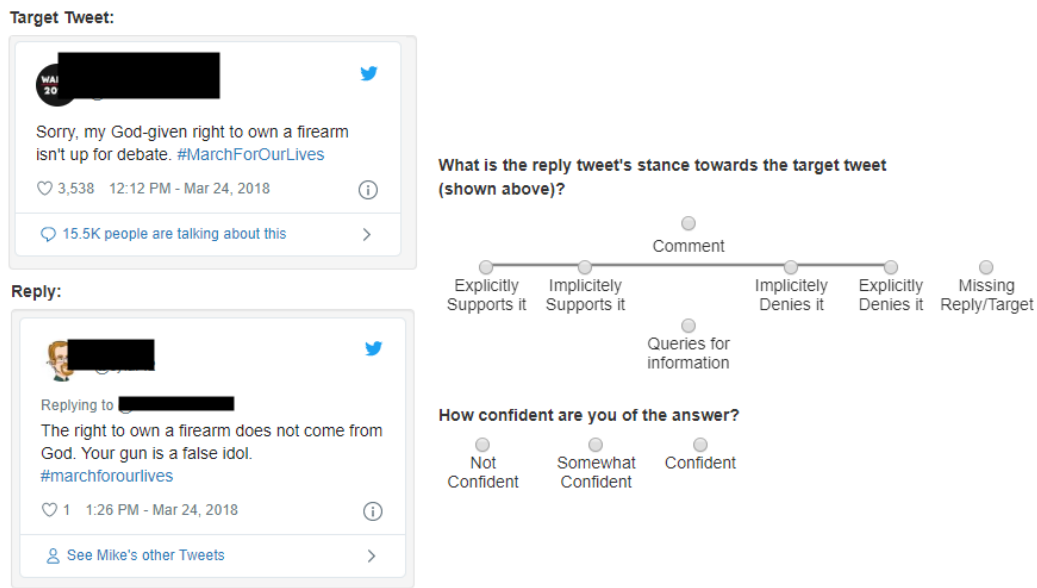


Figure 2.5: Snapshot of a web-form used for labeling replies.

annotator. If both labels agreed, the label was accepted and if not the task was passed to a third annotator. Then the majority label was assigned to the response, and in the few cases where disagreement persisted, the process was continued with a different annotator until a majority label was found.

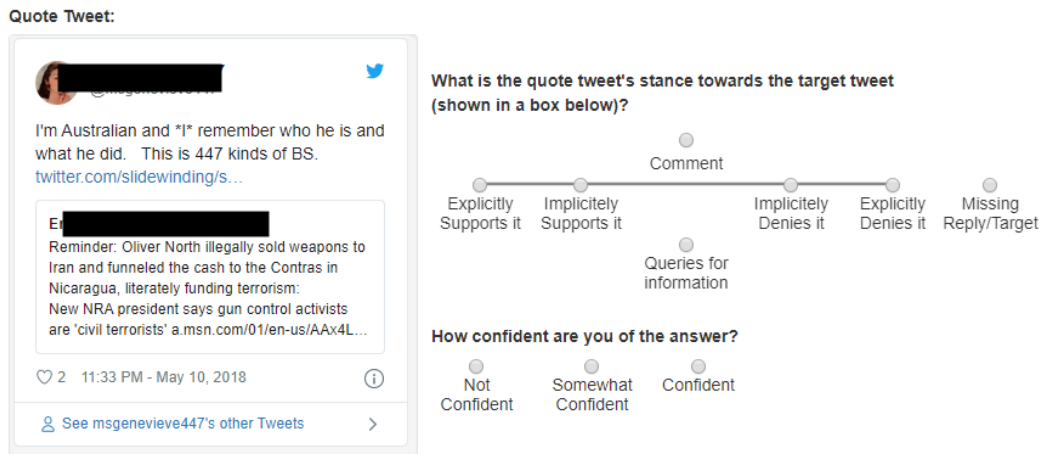


Figure 2.6: Snapshot of a web-form used for labeling quotes.

2.5.1 Definition of Classes

We define the stance classes as:

1. **Explicit Denial:** Explicitly Denies means that the quote/tweet outright states that what the target tweets says is false.
2. **Implicit Denial:** Implicitly Denies means that the quote/tweet implies that the tweeter believes that what the target tweet says is false.
3. **Implicitly Support:** Implicitly Supports means that the quote/tweet implies that the tweeter believes that what the target tweet says is true.
4. **Explicitly Support:** Explicitly Supports means that the quote/tweet outright states that what the target tweets says is true.
5. **Queries:** Indicates if the reply asks for additional information regarding the content presented in the target tweet.
6. **Comment:** Indicates if the reply is neutral regarding the content presented in the target tweet.

2.5.2 Inter Annotator Agreement

To validate the methodology, we selected 55% of the tweets that were initially confidently labeled to be annotated again by a different team member. Of this sample, 86.83% of the tweets matched the original label and the remainder required additional annotation to find a majority consensus. From the 13.17% of inconsistent tweets, a 61.86% were labeled confidently by the second annotator. This means that among the confident labels we validated, only 8.15% resulted in inconsistencies between two confident annotators, which we deemed an acceptable error margin.

Cohens kappa measures the agreement between two or more raters. If each rate labels N items into C categories, Cohen kappa is defined as:

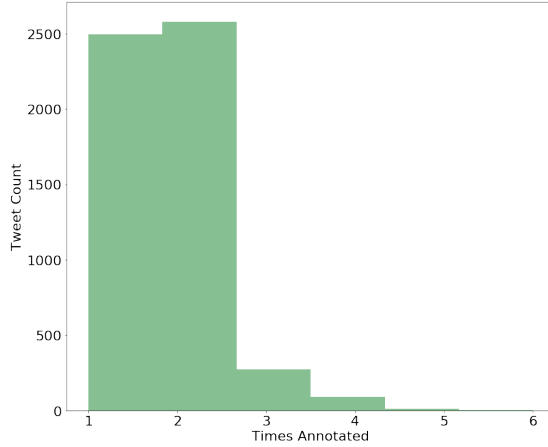


Figure 2.7: Histogram of number of times tweets were annotated. As we used confidence score in labeling, the labeled tweets for which the labler had low confidence were relabeled by more labelers. This process resulted in some tweets getting labeled up to 5 times to obtain confidence in the assigned class label.

$$\kappa = \frac{p_0 - p_e}{1 - p_e} = \frac{0.92 - 0.33}{1 - 0.33} = 0.89 \quad (2.1)$$

where p_0 is the relative observed agreement among raters and p_e is the estimate of possible agreement by chance. In our experiment, $p_0 = 0.92$ and agreement chance $p_e = 0.33$ as there are three class types. This leads to κ value of 0.89

Figure 2.7, shows the distribution of times the tweets were annotated. As shown, 45% of tweets were annotated only once, 47% were annotated twice, 5% were annotated three times and less than 2% required more than three annotations.

2.5.3 Distribution of Labeled Conversations

Table 2.3: Distribution of relevant tweet pairs by response type.

	General Terms	Iran Deal	Santa Fe Shooting	Student Marches
Comment	656 (32.5)	293 (23.5)	246 (20.2)	153 (20.8)
Explicit Denial	521 (25.8)	350 (28.1)	471 (38.7)	253 (34.4)
Implicit Denial	202 (10)	116 (9.3)	116 (9.5)	49 (6.7)
Explicit Support	138 (6.8)	118 (9.5)	85 (7)	47 (6.4)
Implicit Support	415 (20.5)	327 (26.2)	279 (22.9)	215 (29.2)
Queries	88 (4.4)	42 (3.4)	21 (1.7)	19 (2.6)

Table 2.3 presents the label distribution for the different events. As expected we observe that the labeled dataset is skewed towards denials as, when combining implicit and explicit types, they constitute the majority label for all events. Interestingly, when applied to a specific event, the "comment" category fall behind the two explicit non-neutral labels. This suggest that for contentious events, the proposed collection methodology is effective at recovering contentious conversations and more non-neutral threads.

In Figure 2.8, we show the distribution of the labels for each type of response. Note that among Quotes, the majority label becomes implicit support, which shows how these types of responses are more context dependent. As we show in the next section, this also translates on a more complex prediction task.

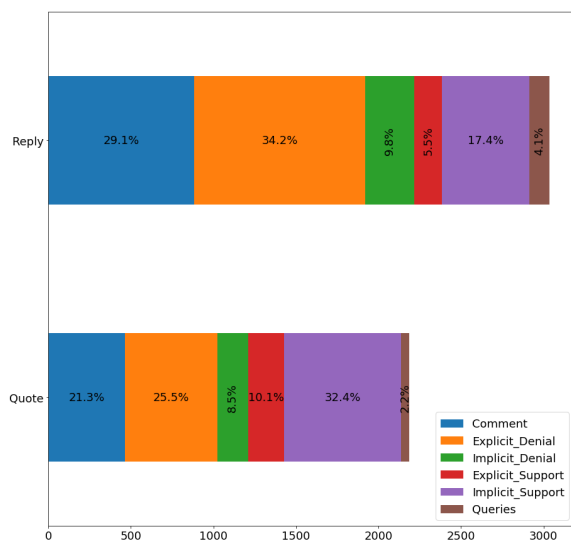


Figure 2.8: Distribution of the labels among the different response types. Note that among Quotes, the majority label is implicit support. This shows how these type of responses tend to be more context dependent and harder to label.

2.5.4 Distribution of Users' Stance

In addition to conversations, we also labeled a small set of users in the dataset for their stance. For 'Santa Fe Shooting' and 'Student Marches', the stance was labeled for 'Pro/Con' gun control. For 'Iran Deal', the stance was evaluated for pro and against the breaking of the Iran deal agreement. The labeled dataset is summarized in Tbl. 2.4.

2.6 Dataset Schema and FAIR principles

In adherence to the FAIR principles, the database was uploaded to Zenodo and is accessible with the following link <http://doi.org/10.5281/zenodo.3609277>. We also adhere

Table 2.4: Distribution of labeled users’ stance.

	Iran Deal	Santa Fe Shooting	Student Marches
Pro	137	188	129
Anti	122	64	154

to Twitter’s terms and conditions by not providing the full tweet JSON but provide the tweet ID so that it can be rehydrated. However, for the labeled tweets, we do provide the text of the tweets and other relevant metadata for the reproduction of the results. The annotated tweets are included in a JSON file with the following fields:

- *event*: Event to which the target-response pair corresponds to.
- *response_id*: Tweet ID of the response, which also served as the unique and eternally persistent identifier of the labeled database (in adherence to principle F1).
- *target_id*: Tweet ID of the target.
- *interaction_type*: Type of Response: Reply or Quote.
- *response_text*: Text of the response tweet.
- *target_text*: Text of the target tweet.
- *response_created_at*: Timestamp of the creation of the response tweet.
- *target_created_at*: Timestamp of the creation of the target tweet.
- *Stance*: Annotated Stance of the response tweet. The annotated categories are: Explicit Support, Implicit Support, Comment, Implicit Denial, Explicit Denial and Queries.
- *Times Labeled*: Number of times the target-response pair was annotated.

We also include a separate dataset that provides the universe of tweets from which the labeled dataset was selected. Because of the number of tweets involved, we do not include the text of the target-response pairs. These tweets are included in a JSON file with the following fields:

- *event*: Event to which the target-response pair corresponds to.
- *response_id*: Tweet ID of the response.
- *target_id*: Tweet ID of the target.
- *interaction_type*: Type of Response: Reply or Quote.
- *response_text*: Text of the response tweet.
- *terms_matched*: List of ‘contentious’ terms found on the text of the response tweet.

2.7 Baseline Models and Their Performance

We consider a number of classifiers including traditional text features based classifiers and neural-networks (or deep learning) based models. In this section, we describe the input features, the model architecture details, the training process and finally, discuss the results.

2.7.1 Input Features

As we have sentence pairs as input, we use features extracted from text to train the models. For each sentence pair, we extract text features from both the source and the response separately.

TF-IDF

Tf-Idf (Term frequency- inverse document frequency) [104] is very popular feature commonly used in many text based classifier. In our research, we use TF-IDF along with Support-Vector Machine (SVM) model that we describe later.

Glove (GLV)

In this kind of sentence encoding, word vectors are obtained for each word of a sentence, and the mean of these vectors are used as the sentence embedding. To get word vectors, we used Glove [96] which is one the most commonly used word vectors. Before extracting the Glove word vectors, we perform some basic text cleaning which involves removing any @mentions, any URLs and the Twitter artifact (like ‘RT’) which gets added before a re-tweet. Some tweets, after cleaning did not contain any text (e.g. a tweet that only contains a URL or an @mention). For such tweets, we generate an embedding vector that is an average of all sentence vectors of that type in the dataset. The same text cleaning step was performed before generating features for all embeddings described in the paper.

Skip-thoughts (SKP)

We use the pre-trained model shared by the authors of Skipthought ³. The model uses a neural-network that takes sentences as input and generate a 4800 dimension embedding for each sentence [63]. Thus, on our dataset, for each post in Twitter conversations, we get a 4800 dimension vector

DeepMoji (DMJ)

We use the DeepMoji pre-trained model ⁴ to generate deepmoji vectors [35]. Like skipthought, DeepMoji is a neural network model that takes sentences as input and outputs a 64 dimension feature vectors.

The process of training the LSTM model using DeepMoji vectors closely follows the training process for the semantic features. The only difference is that the input uses DeepMoji vectors, and hence the size of input vector changes.

2.7.2 Classifiers

As mentioned earlier, we tried two types of classifiers: 1) TF-IDF Text features based classifiers, and 2) neural-networks (deep learning) based classifiers. For the classification task, we only

³<https://github.com/ryankiros/skip-thoughts>

⁴<https://github.com/huggingface/torchMoji>

consider four class classification by merging ‘Explicit Denial’ and ‘Implicit Denial’ as Denial, and ‘Implicit Support’ and ‘Explicit Support’ as Support. We describe the details of the classifiers next.

SVM with TF-IDF features

Support Vector Machine (SVM) is a classifier of choice for many text classification tasks. The classifier is fast to train and performs reasonably well on wide-range of tasks. For the Text SVM classification, we only use the reply text to train the model. The classifier takes TF-IDF features as input and predicts the four class stance classes. We would expect that this simple model cannot effectively learn to compare the source and the reply text as is needed for good stance classification. However, we find that such models are still very competitive and therefore serves as a good baseline.

Deep Learning models with GLV, SKP, DMJ features

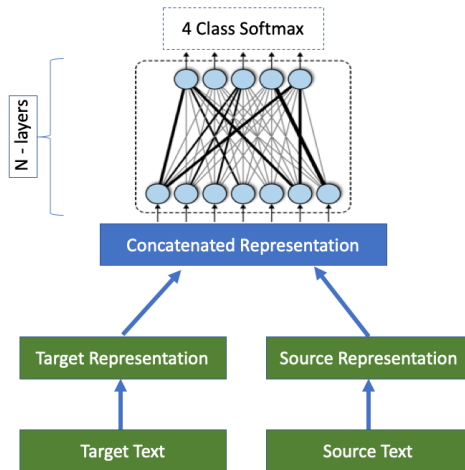


Figure 2.9: Deep learning model sample diagram

As opposed to traditional text classifiers, neural-network based models could be designed to effectively use text-reply pair as input. One such model is shown in Fig. 5.7. A neural network based architecture that uses both source and reply can effectively compare target and reply posts and we expect it to result in a better performance. This type of neural network can further be divided in two types based on inputs: 1) Word vectors (or embeddings) are used as input such as Glove (GLV), 2) Sentence vectors (or sentence representations) are used as input such as skip-thoughts, DeepMoji and a joint representation of skip-thought and deep-moji (SKPDMJ). The first model that takes word embeddings as input requires a recurrent layer that embeds the text and reply to a fixed vector representation (one for target and one for reply). One fully connected layer that uses the fixed vector representation input and a softmax layer on top to predict the final stance label. The second type of model that uses the text and reply representations only have one (or more) fully connected layer and a softmax layer on top to predict the final stance label.

Table 2.5: Classification results for Replies: F1-score (micro) and mean of F1 scores (Mean) for different events. QOT implies quotes, RPL implies replies and CMB implies combined quotes and replies.

Model↓ Event →	Iran Deal (ID)			General Terms (GT)			Student Marches (SM)			Santa Fe Shooting (SS)			Mean		
Data Type	QOT	RPL	CMB	QOT	RPL	CMB	QOT	RPL	CMB	QOT	RPL	CMB	QOT	RPL	CMB
Baseline Models															
Majority	0.46	0.47	0.37	0.37	0.36	0.36	0.53	0.50	0.41	0.40	0.56	0.48	0.44	0.47	0.41
Text SVM	0.44	0.44	0.43	0.46	0.41	0.41	0.45	0.51	0.45	0.44	0.55	0.48	0.45	0.48	0.44
Deep Learning Models															
Glove	0.41	0.46	0.40	0.42	0.41	0.42	0.49	0.48	0.47	0.47	0.56	0.49	0.45	0.48	0.45
SKP	0.46	0.42	0.39	0.38	0.37	0.37	0.48	0.50	0.42	0.38	0.53	0.46	0.43	0.45	0.41
DMJ	0.46	0.46	0.40	0.40	0.39	0.41	0.54	0.51	0.44	0.41	0.56	0.48	0.45	0.48	0.43
SKPDMJ	0.45	0.41	0.39	0.39	0.39	0.36	0.46	0.49	0.42	0.46	0.51	0.44	0.44	0.45	0.40

2.7.3 Classifiers Training

Our neural-network based models are built using Keras library ⁵. The models used feature vectors (Glove, SKP, DMJ) as input. Because Glove is a word vector embeddings, we use a recurrent layer right above the input to create a fixed size sentence embeddings vector. For SKP, DMJ and SKPDMJ, the concatenated sentence representation is used as the input to the next fully connected layer. The fully connected layer is composed of relu activation unit followed by a dropout (20 %) and batch normalization. For all models, a final softmax layer is used to predict the output. The training of SKPDMJ model also followed the same pattern except the concatenation of SKP and DMJ features which is used as the input. The models are trained using ‘RMSProp’ optimizer using a categorical cross-entropy loss function. The number of fully connected layers and the learning rate were used as hyper-parameter. The learning rate we tried were in range 10^{-5} to 10^{-1} . The fully-connected layer size we tried varied from 1 to 3. Once we find the best value for these hyper parameters by initial experiments, they remain unchanged during training and testing the performance of the model for all four events. For all models we find that a single fully connected layer performs better than multi-layered fully connected networks, so we use single layer network for all the results discussed next.

2.7.4 Results and Discussion

We summarize the performance of the models in Tab. 2.5 in which we show the f1 score (micro) for all models for each dataset. As we can observe, if we consider the mean values across events, the replies-based models perform better. The performance is better not just when compared with quotes but also when compared with combined quotes and replies data. In fact, in all but one case, the model trained on combined data performs worse than both the replies based model and quotes based model. This confirms our earlier suspicion that people use quotes and replies in different ways on Twitter, and it is better to train separate models for inferring stance in quotes and replies.

⁵<https://keras.io/>

If we compare the input features (Glove, SKP, DMJ, SKPDMJ), we can observe that most models are only slightly better than the majority (class) based model, which means that this problem is very challenging. The SVM model that used TF-IDF text features is the simplest yet performs as good as the deep learning models. Only on the combined data, the SVM is .01 worse than the Glove based model. This is not completely unexpected, as we know that most deep learning models require a lot of data to train, and in our case, we barely have a few thousand examples. What is more interesting is that even among the deep learning models, the Glove features based model that is the simplest to train, performs better than all other feature-based models. This is also unexpected given that earlier work, e.g., [68], has indicated the benefit of using sentence vectors (SKP, DMJ and SKPDMJ) in comparison to word vectors based models (GLOve). This phenomenon could partially be because of the difference in the models used in the earlier work.

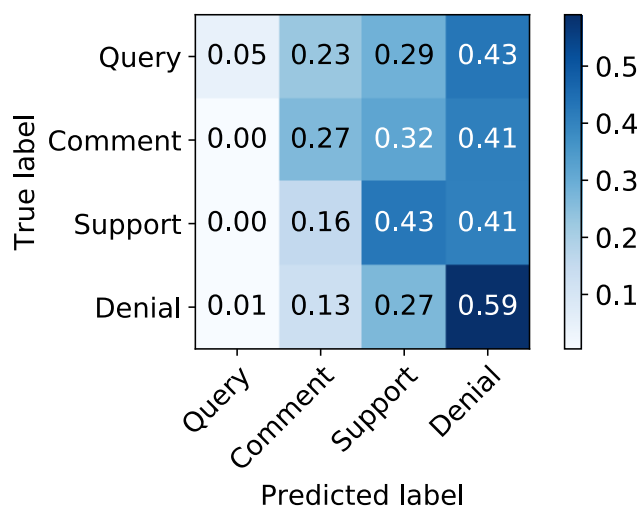


Figure 2.10: Confusion Matrix for Glove feature based deep-learning model for combined quotes and replies data.

If we consider the confusion matrix as shown in Fig. 2.10, we can observe that the ‘Denial’ class is the best performing class followed by ‘support’ class. This is aligned with the overall objective of this research to improve the denial class performance. In future work, we would like to combine the dataset prepared in earlier research [137] where ‘comment’ is the majority class and and this new dataset that has more ‘Denial’ and ‘Support’ labels.

2.8 Conclusion and Future Work

In this research, we created a new dataset that has stance labels for replies (and quotes) on Twitter posts on three controversial issues and on additional examples which do not belong to any specific topic. To overcome the limitations of prior research, we developed a collection methodology that is skewed toward non-neutral responses, and therefore has a more balanced class distribution

as compared with prior datasets that have ‘Comment’ as the majority class. We find that, when applied to contentious events, our methodology is effective at recovering contentious conversations and more non-neutral threads. Finally, our dataset also separates quotes and replies and is the first dataset to have stance labels for quotes. We envision that this dataset will allow other researchers to train and test models to automatically learn the stance taken by social-media users while replying to (or quoting) posts on social media.

We also experimented with few machine learning models and evaluated their performance. We find that learning stance in conversations is still a challenging problem. Yet stance mining is important as conversations are the only way to infer negative links between users of many platforms, and therefore inferring stance in conversations could be very valuable. We expect that our new dataset will allow the development of better stance learning models and enable a better understanding of community polarization and the detection of potential rumors.

Chapter 3

Learning Users' Stance by Combining Users' Networks and Users' Posts by Creating Virtual Connections

3.1 Introduction

Online social media platforms are popular for sharing information and allowing users to network with each other. Analyzing such social networks is an active area of research. Significant problems in this field are finding communities, predicting interests of users, and recommending friends and content. Typical end goals are to identify groups, infer links and make network predictions. To achieve these goals, it is first necessary to determine for two social-media users, how socially proximate they are. Two individuals who are connected by a friendship tie or a follower-followee tie are more socially proximate than are those not connected. However, just examining the binary friendship or follower-followee ties is insufficient for assessing their overall social proximity. Users are more socially proximate vis-a-vis a topic, the more they have in common. Learned representations using random walks over the network links provide a better feature to find the social proximity of users, but they still miss important cues such as what they say or feel about a topic. We argue, that in addition to the explicit ties among users, the activities and preferences of the social media users could be used to find weighted *virtual links* that can be leveraged to learn more useful node representations.

In this paper we present People2vec, a latent space representation of the user in context as a vector. This representation supports generalization and the identification of similar actors and so groups. This representation is learned from the data and captures the complexities of the situation in which the user is embedded. Our approach is inspired by an analogous problem in language technology which is to learn a representation of a word from the common context of the word in sentences. To that end we draw on the word2vec approach for identifying the context of the node by random traversals of the network similar to [45, 97]. However, we move beyond this approach by bringing in user activities as node attributes, which enables the inference of additional links and solves the sparse network problem. Existing network analytic tools, like ORA, uses both the network and the attributes on the nodes. By exploiting attribute information to infer the network,

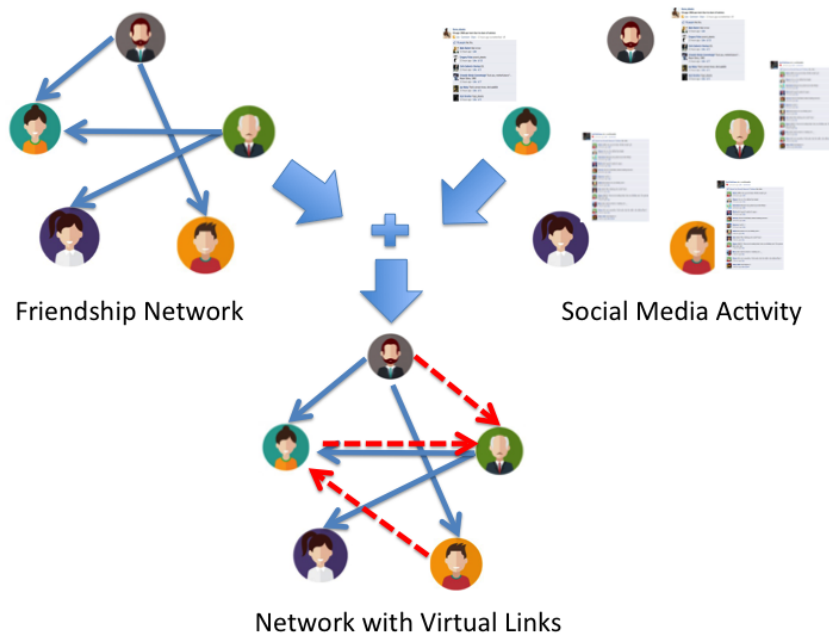


Figure 3.1: Using just the binary friendship network, the only similarity between the orange and green person is that they are both friends of the brown person. Each person also has social media activity, e.g. a set of messages that they send. Combining these two types of data can generate new weighted virtual links (the red dashed lines) between nodes and reveal hidden connections.

People2vec supports a more detailed analysis.

The important contributions of this paper are:

- We propose People2vec, a model to learn node representations that captures similarity in users’ attributes and activities, in addition to their friendship links.
- Our model extends the popular random walk approach of learning node representation and brings valuable improvements, yet preserves the simplicity of the approach.

The paper is outlined as follows. First, we discuss prior work in section 3.2. Then, we introduce our ‘People2Vec Model’ in section 3.3. In section 3.4, we present our experiments on two different datasets, along with a discussion of the results. We finally conclude and discuss the future work. Code to reproduce the results is available on Github ¹.

3.2 Related Work

Some recent advancements in learning node representations are inspired from the improvement in natural language processing. Bengio et al. [11] proposed the distributed representations of words aka ‘Word embeddings’ which was later used by Collobert and Weston [24] to demonstrate their usefulness in many NLP tasks. Mikolov et al. [88] proposed Word2vec, a Skip-gram model for learning high-quality distributed vector representations using skip-gram model. Such representations capture many syntactic and semantic word relationships e.g. they predict:

¹<https://github.com/CASOS-IDeaS-CMU/People2Vec>

$\text{vec}(\text{'Berlin'}) - \text{vec}(\text{'Germany'}) + \text{vec}(\text{'France'}) = \text{vec}(\text{'Paris'})$. The concept of learning representations using skip-gram could be applied to networks as well. These latent representations can be learned in a number of ways; e.g. a) factorization of social network's adjacency matrix b) learning functions to find better features. Recently, researchers have tried to train neural-networks to find efficient nonlinear transformations for learning node embeddings. However, unlike words in a language that has plenty of examples to get related contextual words, often networks are sparse. Besides, there is no clear notion of social-context in networks. To incorporate context in networks, researchers have tried random walks [45, 97]. Random walks on links in a network help to generate contexts that consist of proximity nodes. Like in language models, such context is then used to build embedding vectors (representations) often by training a shallow neural network. Learning low dimensional representation of nodes in networks allow mapping local structural characteristics to a continuous space representation. Learning these representations of nodes have helped to improve performance in many tasks including node classification and link prediction. Though models proposed in [45, 97, 115] learn good representation on simple graphs, they mostly explore binary edges, so are best suited for social-networks that have clear links as in friendship and follower-follower relationship. They do not exploit node attributes and preferences which are often very relevant and strong indicators in social networks. This gap is the focus of this research.

3.3 People2Vec Model

We consider the problem of learning node representation in social-networks that captures users' preferences, in addition to their explicit links in the form of friendship or follower-follower relationship. We expect a good solution to have the following two properties:

a) Users with direct links in a network should be closer in latent representation space. Many existing models exhibit this property.

b) Users with similar preferences should be closer in latent representation space. The way to measure similarity in preferences should be flexible to allow the model to adapt to different formulations of preferences. For example, in one situation, two users discussing a topic could be similar, and in another situation, two users using same tag could be similar.

We formulate the problem as follows: Let $G(V, E)$ be a network where $v \in V$ are nodes (or users) and $e \in E$ are edges. Let $F : v \rightarrow \mathcal{R}^d$ be the function that learns a d dimensional representation (z) of a node. Let Y be a matrix of user preferences that contains a set of preferences for each user, where Y_i^j indicates preferences of node v_i towards j th item. The j th 'item' could be stance towards a topic or the count of a tag (as in hashtag used by a user). The goal of the algorithm is to learn z_i , a low-dimensional representation of user v_i that considers the explicit links in E and also the similarity in Y space.

As in Deepwalk[97], we follow the language modeling technique of generating latent representation of words from sentences. In language modeling, given some text corpus $W = (w_0, w_1, \dots, w_n)$, the goal is to maximize the likelihood $Pr(w_i | w_0, w_1, \dots, w_{i-1}, w_{i+1}, \dots, w_n)$ over the entire corpus. By analogy, in social networks, we define the likelihood of observing a node v_i given other nodes by $Pr(v_i | v_0, v_1, \dots, v_{i-1}, v_{i+1}, \dots, v_n)$. In the latent space F of node representations, this can be formulated as maximizing the likelihood of

$$Pr\left(v_i | (F(v_{i-k}), F(v_{i-k+1}), \dots, F(v_{i-1}), F(v_{i+1}), F(v_{i+k-1}), F(v_{i+k}))\right) \quad (3.1)$$

where we only consider $2k$ immediate neighbors on node v_i . To efficiently solve such a formulation, we use the skip-gram [88] approach. Taking log the optimization problem can be formulated as :

$$\min_F - \log Pr\left((v_{i-k}, v_{i-k+1}, \dots, v_{i+k-1}, v_{i+k}) | F(v_i)\right) \quad (3.2)$$

Since a node and its latent vector have symmetry in latent space, the conditional likelihood of a neighbor node v_j given by $Pr(v_j | F(v_i))$ can be approximated as similarity in latent space. As in Deepwalk [97], we use the stochastic gradient descent over neighbor nodes collection generated by random walk to optimize the final objective function. To speed up the training we used hierarchical soft-max [?]. In the proposed model, the neighborhood of a node v_i ($v_0, v_1, \dots, v_{i-1}, v_{i+1}, \dots, v_n$) is not limited to nodes reachable by explicit links, and is extended to nodes having similar attributes. We call such links ‘*virtual links*’ (explained next).

3.3.1 Virtual Links from Users’ Activities

We define *virtual links* as edges in social-networks that connect users with similar preferences as evident by their involvement on these platforms. These virtual links (like real links) can be used in random-walks to explore node neighborhood, thus enhancing the network information present in the original networks. Moreover, these virtual edges, based on nodes similarities, can also strength the existing linkage. Hence, virtual links enable to learn more meaningful node representations.

There are two possible formulations of virtual-links. A rigid link that is either present or absent, and a weighted link that that gives a probabilistic score of links being present. We go with the probabilistic version of virtual-links as it enables a more flexible learning framework.

The probability of having a virtual link between two users is based on similarity between their activity on social-media platform. As discussed earlier, let’s model the activity profile of users as a matrix Y_i^j , where Y_i^j indicates preferences of node v_i towards j th item. Similarity between users is obtained by measuring similarity between vectors representing users preferences.

The similarity between two users is defined as:

$$Similarity(v_k, v_i) = Sim(Y_k, Y_i) \quad (3.3)$$

There are a number of ways to measure such similarity. We tried three such similarity measures: a) Cosine Similarity b) Hamming Distance c) Euclidean Distance. On our datasets, we find that ‘Cosine Similarity’ (CS) performs better than other measures. Hence, we use CS as the similarity measure in rest of the paper.

Random Walks

We use random walks to sample neighborhood nodes. The random walks approach provides a more flexible approach over known 'Depth First' and 'Breadth First' sampling as explained in [45]. A random walk starts from a node, say v_0 , and uses node links to find the next node. In prior studies, the probability of transition from node v_0 to v_i is given by:

$$P(v_i|v_0) = \begin{cases} 0, & \text{if } (v_0, v_i) \notin E \\ \frac{1}{\sum_k 1}, & \text{otherwise} \end{cases} \quad (3.4)$$

where k = Number of links of v_0 .

Random Walks over Virtual Links

In our model, we extend the random walk over nodes to include random walk over virtual links. Figure 3.2 explains the idea.

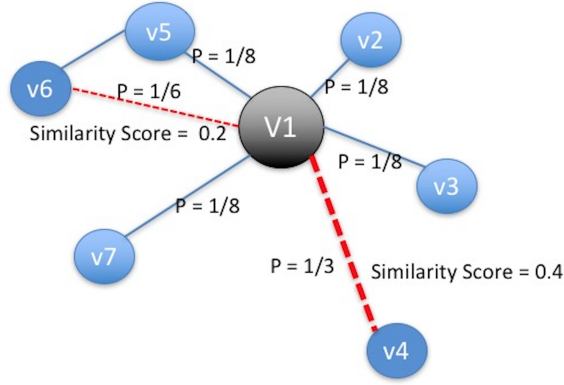


Figure 3.2: An example of a random walk transition in People2vec. Real links are shown in blue and virtual links are shown in red. The walk originates from node v_1 . The probability of transition to other nodes is shown in text. Similarity scores for virtual links are obtained using *Sim* function. Here we consider the weighing factor $\alpha = 0.5$, i.e. we weigh the real links and the virtual links equally. For clarity, we have not shown virtual links for nodes already connected via real-links.

To weigh the relative importance of virtual links and to real links, let's introduce a hyper-parameter α , a weighing factor. This parameter is tuned based on characteristics on the network under consideration. For random walks over virtual links (see Fig. 3.2), we use the probability of transition from node v_0 to v_i as:

$$P(v_i|v_0) = \begin{cases} \alpha * Sim'(Y(v_0), Y(v_i)), & \text{if } (v_0, v_i) \notin E \\ (1 - \alpha) * \frac{1}{\sum_k 1} + \alpha * Sim'(Y(v_0), Y(v_i)), & \text{if } (v_0, v_i) \in E \end{cases} \quad (3.5)$$

where k = Number of real links of v_0 , and Sim' is the normalized similarity score defined as $Sim' = \frac{Sim(Y(v_0), Y(v_i))}{\sum_{v_i} Sim(Y(v_0), Y(v_i))}$.

3.3.2 People2Vec Algorithm

People2Vec extends the original Deepwalk algorithm [97] by including virtual links. Like in Deepwalk, our algorithm uses random walk to learn node representation. However, in the process of generating walks, the algorithm uses ‘virtual links’ in addition to the real links, thus considers neighbors that were not accessible in plain random walks. The steps as described in Algorithm 1. Again, similar to Deepwalk, we use skip-gram [88] algorithm to efficiently learn the representations for each node.

Algorithm 1 The People2Vec algorithm.

```
1: learnRepresentations
2: Graph G, VirtualGraph  $G_v$ 
3: RepresentationDimension d
4: walks per node r
5: walk length l
6: window size w
7:  $walks = []$ 
8: for  $iter \in \{1, \dots, r\}$  do
9:   for  $node \in V$  do
10:      $walk = \text{randomWalk}(G, G_v, node, l)$ 
11:      $walks.append(walk)$ 
12:   end for
13: end for
14: SkipGram(F,  $walks$ , w) (see Ref. [88])
15: _____
16: randomWalk(Graph G, VirtualGraph  $G_v$ , Start node u, Length l)
17:  $walk = [u]$ 
18: for  $walk\_iter \in \{1, \dots, l\}$  do
19:    $currentNode = walk[-1]$ 
20:    $newNode = \text{getNeighbor}(currentNode, G, G_v)$ 
21:    $walk.append(newNode)$ 
22: end for
23: _____
24: getNeighbour( Start node  $v_0$ , Graph G, VirtualGraph  $G_v$ )
25: start at v
26:  $N_r = \text{getRealNeighbors}(G, v_0)$ 
27:  $N_v = \text{getVirtualNeighbors}(G_v, v_0)$ 
28: pick neighbor  $v_1$  from  $[N_v + N_r]$  using  $P(v_1|v_0)$  (See Eqn. 3.5 )
```

3.4 Experiments and Results

In this section, we present the experimental evaluation of our model on two different datasets. The first dataset is a set of blogs, second is a sample from Flickr website and the third is a stance dataset based on Twitter data. We also evaluate the impact of using different latent-space dimensions.

3.4.1 Datasets

We use two existing datasets (BlogCatalog and Flickr) to evaluate our algorithm. These datasets are publicly available and were used in earlier studies [54]. In addition, we also experiment with stance detection where users are labeled as ‘Pro’/‘Con’. For stance detection, the proposed algorithm is used on a Twitter dataset vis-a-vis three controversial topics namely: 1: ‘Gun Control’, 2) ‘Abortion’, and 3) ‘Obamacare’. The dataset was built in a prior study on bias and is described in [81]. We summarize the users in Table 4.1 and their labeled stance in Table 4.2.

Table 3.1: Dataset Description

Dataset	Nodes	Edges	Labels	Attributes
BlogCatalog	5,196	171,743	6	8,189
Flickr	7,575	239,738	9	12,047

BlogCatalog Dataset

BlogCatalog is an online community of bloggers. The dataset is created by including keywords used in blog description as attributes [54]. Using those keywords, we generate users preference matrix. In this dataset, the labels used for predicting the classification performance represent bloggers’ interests.

Flickr Dataset

Flickr is popular website that hosts videos and images. Users can follow each other, thus, forming a network. They can join different groups which is used as labels. To get the users’ preference matrix, tags by users are used [54].

Stance Dataset

We used the dataset created in [80] to find if our approach generalizes to stance learning. We summarize the users in Table 4.1 and their labeled stance in Table 4.2.

Table 4.1 summarizes Biaswatch topics. In the table, RT users mean the number of users that were retweeted in data and the information is used to create the user-retweet graph. Similarly, endtags show the number of unique hashtags used at the end of tweets, and are used to build the

Table 3.2: Topics summary

Events	Users	Tweets	RTUsers	Endtags
Guncontrol	70387	117679	15635	5505
Obamacare	67937	123320	14807	7376
Abortion	111463	173236	26818	9784

user hashtags networks. We use endtags as prior research has shown that hashtags that appear at the end convey stance signal [34].

Table 3.3: Labeled users summary

Events	Neutral	Pro	Anti	Total
Gun-control	60	156	288	504
Obamacare	33	108	363	504
Abortion	55	169	280	504

3.4.2 Baseline Algorithms And Model Optimization

We measure the performance of People2Vec against several state-of-the-art algorithms [45, 97, 115]. DeepWalk [97] uses uniform random walks on networks to learn embeddings as in language modeling techniques like word2vec. LINE [115] algorithm preserves both local and global network structures and uses edge sampling approach for optimization. Node2Vec [45] extends Deepwalk by combining depth-first and breadth-first search in their sampling strategy.

We use social-media tags to create a nodes’ preference vectors. Preference vectors of different nodes are then compared using cosine similarity measures to create weighted virtual links which are later used to learn node representations. Because similarity between nodes generates $O(n^2)$ possible edges, which could result in a very dense graph so we use a threshold (10 closest neighbours unless otherwise stated) to reduce the number of virtual-edges used in learning embeddings. The exact threshold is of lesser importance as People2vec random-walk prefers node transitions to higher similarity nodes, and thus ignores less similar nodes more often. The dimensions of representations used are 64 and 128. For a fair comparison, all algorithms used $walklength = 10$ and $walkcount = 40$. For all models, we use one-vs-rest logistic regression as used in [97]. We trained all the models on an Ubuntu Linux machine with 64 GB ram and eight core Intel i7 processor with 4.00 GHz processing speed.

3.4.3 Experimental Results

We present the results of the experiments. Table: 3.4 shows the top classification performance for the two datasets. Figure 3.3 shows the trend of F1-score (macro) for different train and test ratio. The plot show that most algorithms have a reasonable performance right from the

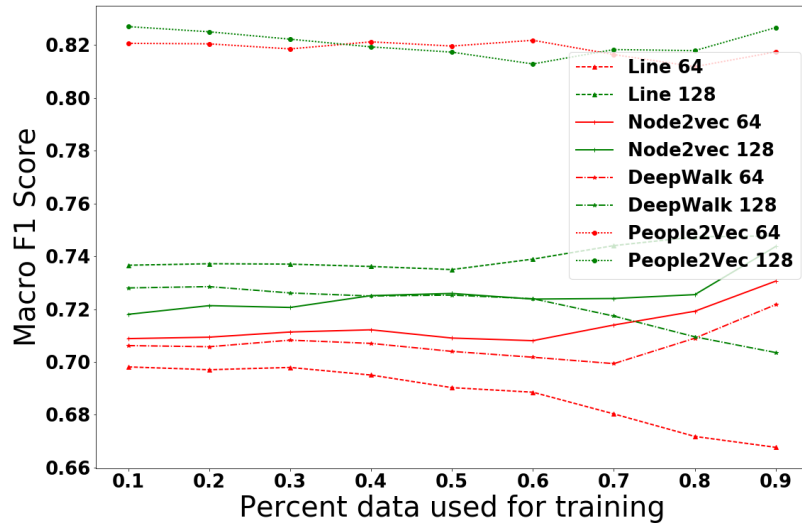


Figure 3.3: We tried different algorithms to learn the class labels (a proxy of community) of nodes in BlogCatalog graph. In this plot, we compare mean F1 score for different algorithms.

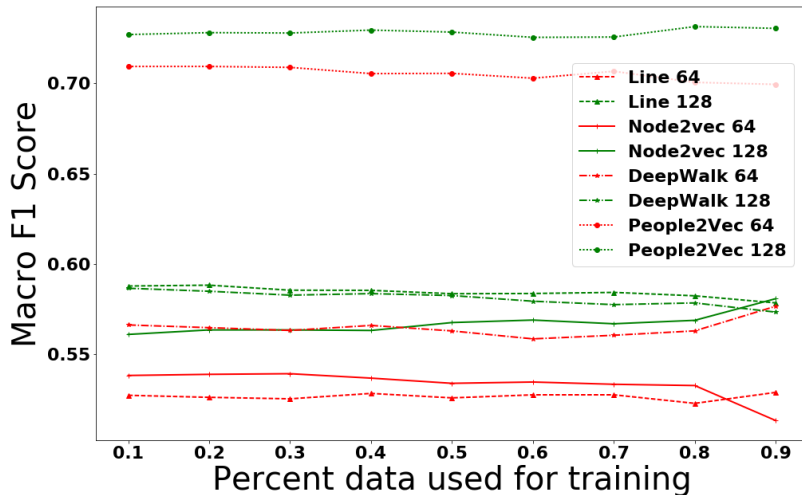


Figure 3.4: We tried different algorithms to learn the class labels (a proxy of community) of nodes in Flickr graph. In this plot, we compare mean F1 score for different algorithms.

smallest training percentage (10%). There are small improvements as we increase the training data percentages. Peopl2vec performed better than rest of the algorithms. In general, embeddings with 128 dimension perform better than 64 dimension embeddings. People2Vec is an exception for which scores were very similar. Figure 3.4 shows the results for the Flickr dataset. Again for most algorithms 128 dimensional embedding performed better than 64 dimensional. Like before, Peopl2vec performed better than all other algorithms.

Table 3.4: Best F1 Macro Score

Method	BlogCatalog	Flickr
DeepWalk	0.73	0.58
Node2vec	0.72	0.58
LINE	0.73	0.58
People2Vec	0.83	0.75

We don't compare our results with matrix factorization based approaches (like LANE [54], BlogCatalog: 0.90 best F1 score, Flickr: 0.90 best F1 score) as walk based approaches allows to generalization and extension to other modalities as we show in the next chapters. However, we want to note that we observe similar performance gains over the baselines e.g. on BlogCatalog dataset, LANE improved from 0.81 (Deepwalk) to 0.90 (LANE). The gain is similar to our work which improved from 0.73 (Deepwalk) to 0.83 (People2vec).

3.4.4 Parameter Sensitivity

We consider the following parameters in evaluating People2Vec: a) Different measure of similarity for creating virtual links, b) Effect of parameter α that allows to weigh real links vs virtual links c) Effect of *embedding dimensions*.

Effect of different similarity measures:

We tried three similarity measures to find the similarity between any two users. a) Cosine similarity: b) Hamming distance c) Euclidean distance. The cosine similarity gives better performance on the two datasets we used. Fig. 3.5 shows the performance of People2Vec if Euclidean distance is used to measure similarity. The best accuracy is observed for $\alpha = 0.08$, and a higher value of α , in general, reduced the performance.

Fig. 3.6 shows the performance on People2vec when Hamming distance is used as the measure of similarity. For Hamming distance, the best performance is obtained for $\alpha = 0.2$ for lower training percentage, but for higher training percentage $\alpha = 0.6$ performs better.

All other plots use cosine similarity metrics, unless specified otherwise.

Effect of weighing parameter α :

Hyper-parameter α needs to be optimized for the particular dataset, as the best parameter depends on the characteristics of networks like size, sparsity, and variance in node attributes. For BlogCatalog, we tried different values of α for embedding dimension 128. We find that $\alpha = 0.5$ gives the best classification performance. This means for the optimum performance, equal weight should be given to virtual and real links in the random walks.

Effect of embedding dimensions:

For comparing People2vec with other approaches, we consider embeddings of two different dimensions of 64 and 128 size. Fig. 3.4 and Fig. 3.3 shows the results. We observe that a higher

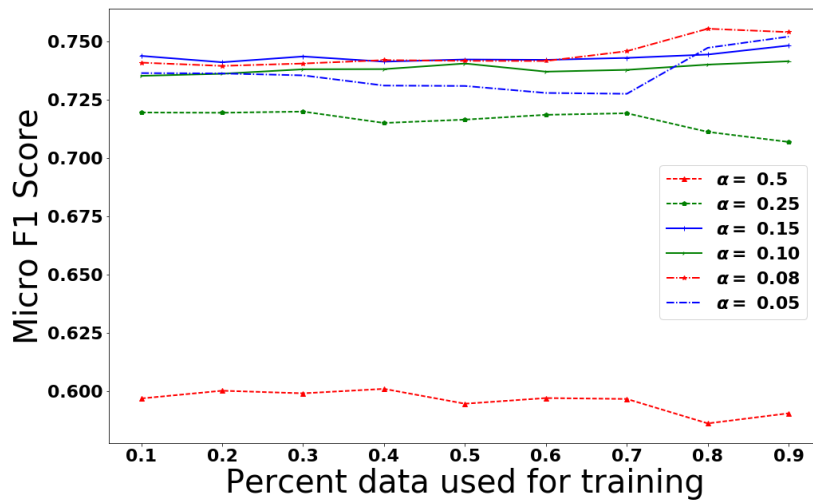


Figure 3.5: F1-micro score for different α values using Euclidean distance as the measure of similarity.

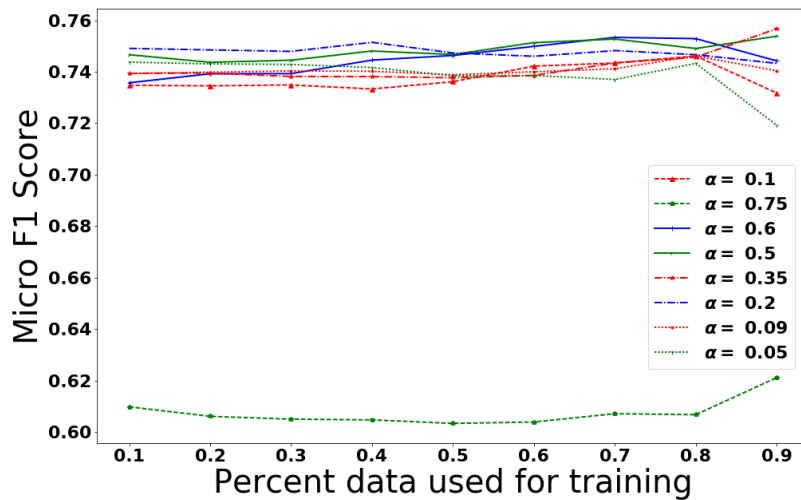


Figure 3.6: F1-micro score for different α values using Hamming distance as the measure of similarity.

dimension leads to a better performance in most cases. We also evaluate the performance of Peopl2vec for many different dimensions. Figure 3.8 shows the trend of F1 score for embedding of dimension 8 to 254, in multiples of 32. The plot shows a general improvement in performance when dimension is increased, but the benefit saturates around size 192, and then the performance decreases.

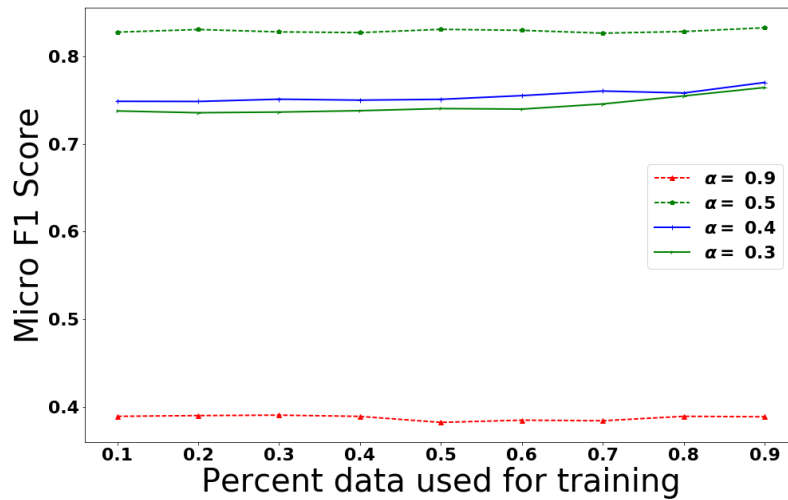


Figure 3.7: F1-micro score for different α values. α indicates the weighing parameter, and a higher alpha means more virtual links are considered in the random walks.

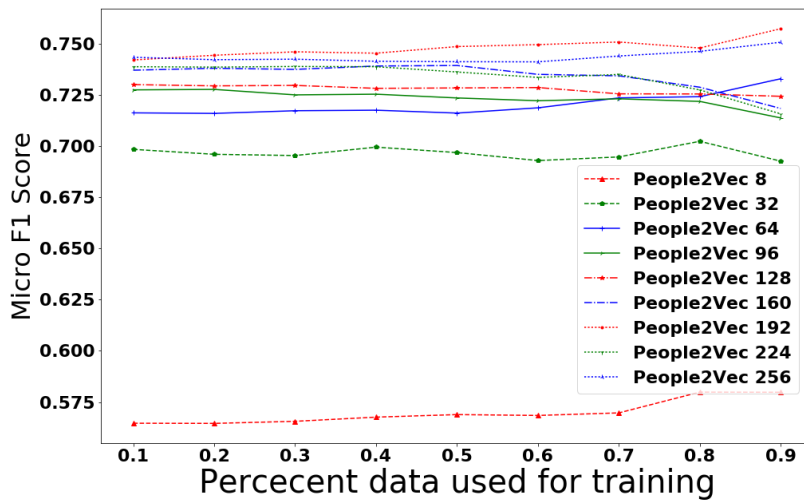


Figure 3.8: F1-micro score for embeddings of different dimensions on Flickr dataset.

3.4.5 Results on Stance Dataset

Next we present results when we apply the Peopl2Vec algorithm to another dataset used to predict stance on users on Twitter. The dataset was described earlier in the Dataset section. Here we elaborate on the training process and the results. The training process used the same methods as we describes earlier for other datasets. For node similarity, we use TF-IDF features extracted for the text of all users and used the co-sine similarity as the measure of similarity. We pick an $\alpha = 0.5$, as that have worked well for other datasets and an embedding dimension of 128. Also, for

computational efficiency we only considered the top 10 similar nodes to each node to create the virtual connections.

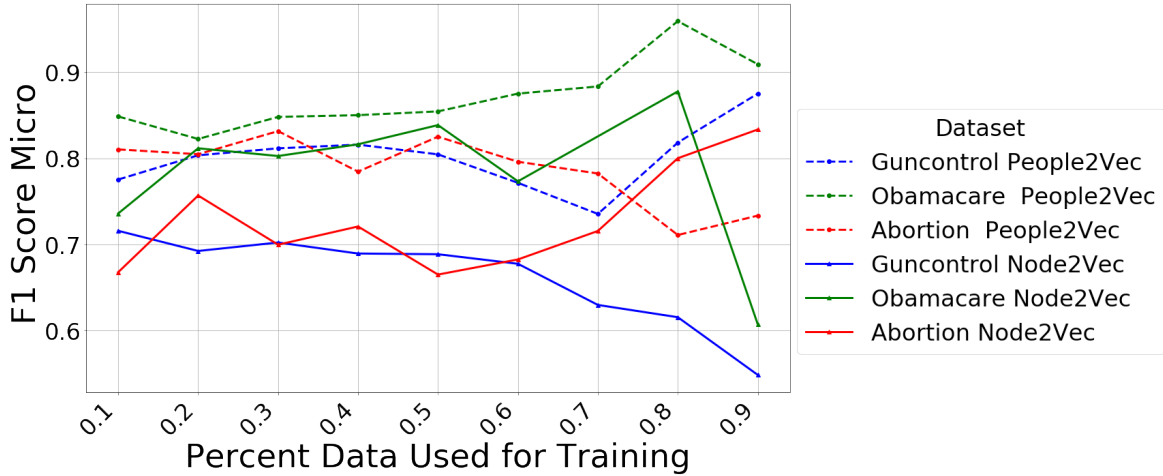


Figure 3.9: F1-micro score for different topics in the Biaswatch dataset. Retweets graph was used as original network and text similarity was used to generate the virtual connections. 128 vector sized embedding was used for both people2vec and node2vec.

As we observe in Fig. 3.9, for almost all training set fraction, the people2vec model exceeds the performance of the Node2vec model. The trend is only reversed for ‘Abortion’ dataset for which, when 0.8 and 0.9 fraction of the entire dataset is used for training, the test performance is better for Node2Vec model. At least for this topic, it implies that adding similarity based virtual connections decreases the test performance. However, looking at other plots, we don’t expect the pattern to generalize.

Overall, we can confidently say that the performance the best performance obtained by People2Vec model is better than the performance obtained by the Node2Vec model, indicating that bringing virtual connection using mode similarity adds useful information for stance detection.

3.5 Conclusions and Future Work

We proposed People2Vec, a model to learn the representation of nodes in complex networks. We used nodes similarity to construct virtual links between nodes. Virtual links enhance the original graph with additional information that uses users’ attributes and preferences. People2Vec considers these virtual edges while building random-walk paths, thus, exploits similarity of nodes in addition to real links. Experiments on two real-world datasets reveal that using People2Vec to learn representations substantially improves node classification performance. On the BlogCataog dataset, we observe an improvement of 13% (F1 score) over other state-of-the-art algorithms to learn embeddings. On the Flickr dataset, the performance improved by 25% on F1-score.

People2Vec is straightforward and intuitive, yet learns better node representations. The approach is also very general, hence can easily be extended to other types of data on users. Peopl2vec significantly improves over the baselines as it can use additional node features (at-

tributes) that are not available to other models. The baseline models only make use of network connections without able to exploit the node attributes. Peopl2vec allows adding new node connections based on the users attributes, making the attributes useful for learning better node embeddings. Better node embeddings, in turn, lead to improved performance for node classification. In future, we plan to investigate the usage of sentiment and emotions in social media posts, to learn node representations that consider the stance of users towards topics. We would also like to explore confidence levels on the results for sparse graphs.

Chapter 4

Co-Training on Social Networks: A Joint Network Label Propagation and Text Classification Approach for Stance Mining

4.1 Introduction

People express their opinions on blogs and other social media platforms. Automated ways to understand users opinion in such large human-generated corpora is of great value. For example, social media data can be used to find the users' perception of a product [56], the spread of diseases [71], the stock market trend [14], and more. A particular sub-field of opinion mining is stance mining – which focuses on finding automated ways to infer users' opinions on controversial topics.

Stance mining is of growing importance because it enables a better understanding of the stance taken in social media posts [91], the polarization of online communities [38] and the spread of misinformation [82]. For learning the stance of social media users, the conventional approach is to use a human-labeled dataset to train a supervised machine-learning classifier and then use the trained model to predict the stance of unlabeled users [76, 87, 91]. Though immensely valuable, this approach has two significant limitations.

First, the supervised learning approach works well for tasks where both train and test data are from the same distribution. That is usually the case with topics that change slowly over time. However, many topics evolve quickly. For example, take the topic of immigration reforms in the US. The topic has evolved from arrests on the border, to the new border bill, to the construction of a border wall, to new rules for asylum seekers, to perhaps something new as you read this paper. Because such topics evolve fast, the traditional approach to train a stance classifier using human-labeled examples from the past (which are only available for a few topics) is of limited value.

Second, social-media allows multiple interactions, for example, on Twitter, users can 'Tweet', 'Retweet', 'use hashtags' etc (see Fig. 4.1). Most existing stance learning approaches use text in user's tweets. However, simple text-based classifiers barely perform well, as evident by the SemEval 2016 competition scoreboard, in which the top team received a 0.67 F1-score (average)

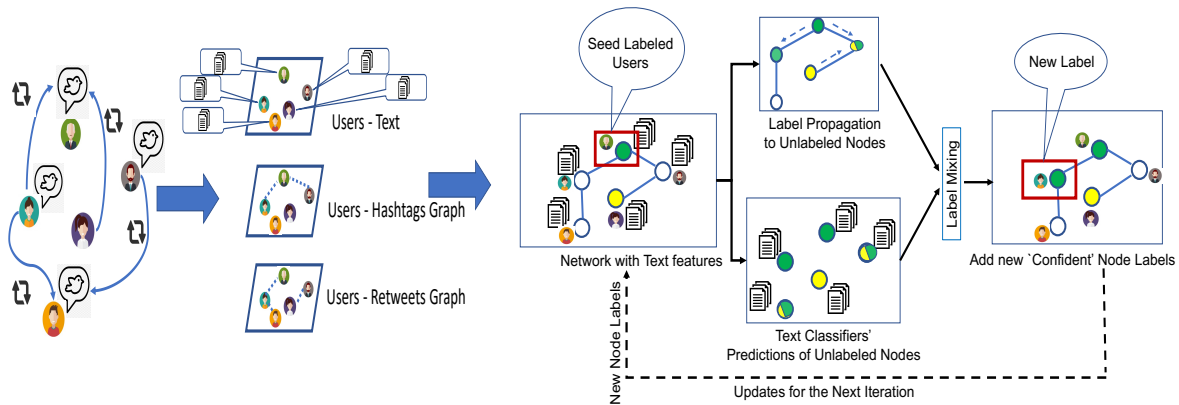


Figure 4.1: Twitter allows multiple interactions that we can broadly group into users’ text messages and users’ networks (e.g., retweets graph). Given data on a topic, a stance classifier (as in pro/con about the topic) can learn from any of these interactions. Though the model trained using a single interaction is useful, this approach of training fails to take advantage of the uncorrelated information in different interactions. In this research, we show the benefits of co-training that leverages information in multiple interactions to train better stance classifiers.

[90] in the two-class stance classification problem. While using multiple interactions makes it harder to train models, prior research on semi-supervised learning has shown that by using multiple interactions, we can exploit additional patterns in unlabeled data to train better models [13]. In this research, we tackle both limitations of stance learning using a new semi-supervised approach that resembles the co-training setting [93] and combines the idea of co-training with another semi-supervised approach, label propagation. Unlike the original co-training approach [13] that used two text-based views, we use a co-training setting in which we train two models using data from two or more types of interactions. By co-training these models, our method allows us to effectively learn from multiple interaction features.

The other advantage of our proposed approach is that we use weak supervision to train models, by utilizing the stance signal given by a few seed hashtags (e.g., #ActOnClimate as pro and #ClimateChangeHoax as anti). Our approach effectively utilizes the stance signal given by such seed hashtags to obtain the initial set of seed users, who are then used to co-train the classification models. Co-training these models improves their performance, not only because sharing the more confident labeled examples expands the training set, but also because one model can label examples that which are otherwise not possible for the other model to label (e.g., label propagation cannot classify nodes in disconnected components without any labeled nodes).

To validate the benefits of our proposed approach, we predict the stance of users on a Twitter dataset that contains manually verified stance labels of 504 users and additional unlabeled users (100,000 users) on three topics. By following the proposed co-training approach, which uses stance given by two to four labeled hashtags as input stance signal, we improve the text based stance classifier by more than 17% on all three topics.

Our main contributions are as follows:

- We extend the original co-training approach by using two different types of models, i.e.,

a label propagation and a text classifier. This combination can learn from both users' networks and users' text features data (sec: 4.3.3). Co-training results in trained models that perform better than the self-trained models (sec. 5.4).

- We use a few hashtags (e.g., #ProChoice:+1,#StandForLife:-1) as a way to obtain the seed training examples. We show that the co-training approach effectively handles such noisy training examples (sec: 4.4.8). Because it's easy to label a few hashtags manually, this makes our approach very flexible.
- The trained text-classifier can be used to predict the stance of new tweets without requiring networks (as in inductive learning) (sec: 4.4.6). This resolves concerns with prior network based approaches that used the transductive learning setup.

The remainder of the paper is organized as follows: We formulate the problem in sec. A.3. We discuss the label propagation algorithm in sec. 4.3.1 and self-trained text classification in sec. 4.3.2. In section 4.3.3, we elaborate on our proposed co-training based approach. We explain our dataset, training methodology, baseline modes, and results in sec. 5.4. In section A.7, we describe other prior work that are relevant to this research. Finally, in sec. 4.6 we conclude our findings and propose future work. Code to reproduce the experiments is available on Github ¹.

4.2 Background and Problem Formulation

Our goal is to propose an approach that can be used to train stance classifiers that predict the stance of users provided their social media data. Moreover, we aim only to use the stance given by a few labeled hashtags as the input signal. Utilizing weak signals to train models is an overarching desire that has been explored in the past, but can be said to have achieved only partial success at best. For instance, in Mohammad et al. [91], the research (and related SemEval competitions) that could be credited to popularize the field of stance mining, briefly discussed the idea of using unlabeled tweets for improving stance classification performance. The authors explored using hashtags based dataset for training. They found that using data from specific hashtags as additional training data could improve the f-score by up to 4 percentage points (on one topic, but minimal on others). Even by including a large datasets built using hashtags, they fail to improve the models' performance partly because people use hashtags not only to explain their stance but for many other purposes [34].

To pick the correct signal from hashtags, we position the stance classification task as a user-level task rather than a text-level task. As discussed earlier, we use the tweets along with their metadata to 1) aggregate text in users' tweets, 2) construct users' retweets network, and 3) construct users' hashtags network. We use these three views to co-train a network-based model and a text-based model. One may ask, are the views conditionally independent (one condition for co-training to work), and if they are not, is there any benefit of using such views. To justify our choice, note that co-training does not require *completely* independent views [30]. Moreover, we have multiple steps that filter the noisy signal. First, each classifier uses only the more confident examples for training. Even then, if we train these classifiers independently, the test performance is on the lower side. To improve this result, we co-train the models (4.3.3), which results in

¹https://github.com/CASOS-IDeaS-CMU/stance_analyzer

improving the performance of these models.

4.2.1 Problem Statement

Given a topic, let's assume we retrieve a set of tweets $T = \{t_1, t_2, t_3, \dots, t_m\}$ which were tweeted by a set of users $U = \{u_1, u_2, u_3, \dots, u_n\}$ where $n \leq m$. The tweets can be grouped by users resulting in a new set of groups of tweets, which we name $D = \{d_1, d_2, d_3, \dots, d_n\}$ where d_i is the collection of tweets tweeted by user u_i . As mentioned earlier, tweets also have additional metadata that are used to build user-hashtags network H and user-retweet network R . Let's assume H is a weighted matrix created from k most used hashtags in the dataset. Similarly, R is a weighted matrix created using p most popular retweets in the dataset. Therefore, $H \in \mathbb{R}^{n \times k}$ matrix and $R \in \mathbb{R}^{n \times p}$ matrix. When the distinction between H and R is not critical, we use I to represent the user interaction matrix (H or R) with edge weights $w_{i,j}$.

Furthermore, because the topics studied in this research are controversial, we assume that users have either pro-stance (+1) or con-stance (-1) (same as anti-stance). Users can also have an unknown stance (0) when the stance of the user is not known. The goal of this research is to correctly assign a stance label $\{+1, -1\}$ to as many users as possible in the set U based on D, H, R . This assignment results in a user-stance matrix $S = \{s_1, s_2, s_3, \dots, s_n\}$ where $s_i \in \{+1, 0, -1\}$.

4.3 Methodology

We divide the proposed method in three parts. We first describe: 1) Label Propagation on Bi-Partite Networks, 2) Self-Trained Text Classifier, and then we explain the 3) Co-training process.

4.3.1 Label Propagation on Bi-Partite Networks with Linear Threshold

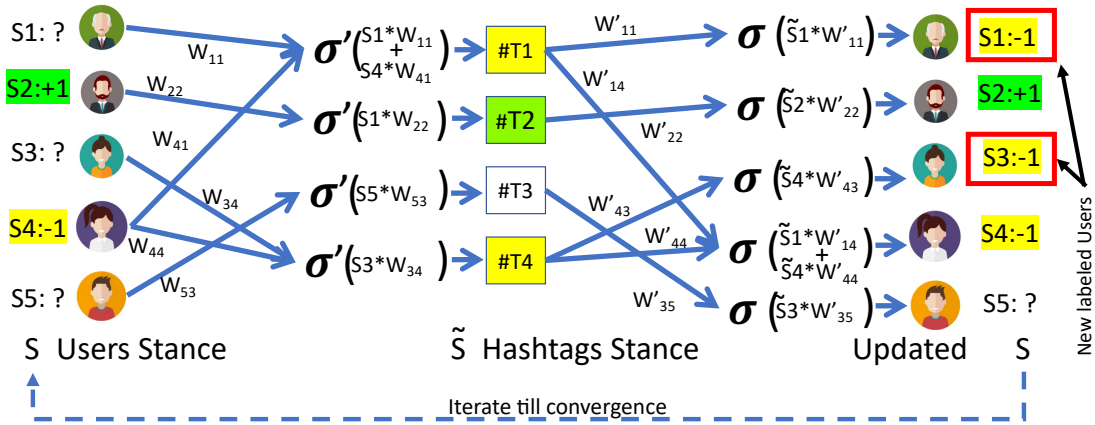


Figure 4.2: Label propagation on bipartite graphs with influence functions σ and σ' , and W is a matrix representing usage of hashtags by users.

In the bi-partite label-propagation, there are two types of nodes, and information flows from both types of nodes. For example, if we consider user-hashtag networks where ‘user’ is one node type, and ‘hashtag’ is another node type, label propagates from users to hashtags. and then from hashtags back to users (see Fig. 4.2). In addition to the propagation of labels through edges in a typical label propagation approach, we incorporate some ideas from influence propagation research [15] to improve our models.

In influence propagation, influence functions are used to model the spread of a disease [74] or a belief [21]. A node (user) is influenced by another node (user), only when certain conditions are met. The condition to get influenced could be as simple as – if a user gets higher than a certain level of influence from the influencers, the user gets influenced. For instance, let’s assume users can have two possible states +1, 0, where +1 implies a user has been influenced, and 0 implies a user has not been influenced. A user gets influenced i.e., ‘+1’, if the ratio of ‘+1’ influence over the sum of all incoming influence (‘+1’ and ‘0’) is above a threshold value. This simple influence propagation model is called the ‘Linear Threshold Model’ (LTM). We modify this LTM model to suit our requirements of three types of stance states.

Let’s assume σ and σ' are influence functions. For a bipartite network I , in step: 1) influence propagates from users to hashtags, and in step 2) influence spreads from influenced hashtags to users. We assume θ_h to be a parameter that acts as a threshold for propagating influence from users to hashtags, and θ_u is another parameter that is another threshold for spreading influence from hashtags to users. The label propagation model could be represented as:

$$\tilde{S} \leftarrow \sigma'_{\theta_h}(I' \cdot S^I) \quad (4.1)$$

$$S^I \leftarrow \sigma_{\theta_u}(I \cdot \tilde{S}) \quad (4.2)$$

where \cdot is dot product, and I' is the transpose of the matrix I . For influence functions, we propose a new model that extends the idea in the LTM model and better suits our requirements.

Linear threshold model (LTM) with decreasing threshold (LTMDT)

As we want the stance to propagate to as many nodes as possible, we propose to use a decreasing threshold function. We name this model LTMDT in which the threshold condition linearly decreases after every iteration that allows propagating the stance to nodes that are connected even by relatively small edge weights. LTMDT model is described as:

$$\tilde{s}_j = \begin{cases} 1, & \text{if } \sum_{k=1}^n w'_{jk} * s_k^I > f_t(\theta_h) \\ -1, & \text{if } \sum_{k=1}^n w'_{jk} * s_k^I < -f_t(\theta_h) \\ 0, & \text{otherwise} \end{cases} \quad (4.3)$$

$$s_j^I = \begin{cases} 1, & \text{if } \sum_{k=1}^n w_{jk} * \tilde{s}_k > f_t(\theta_u) \\ -1, & \text{if } \sum_{k=1}^n w_{jk} * \tilde{s}_k < -f_t(\theta_u) \\ 0, & \text{otherwise} \end{cases} \quad (4.4)$$

where f_t is a uniformly decreasing function of the iteration number t , and s_k^I is users’ stance based on network data. Though confidence estimate is not used in bi-partite label propagation,

we define it here as it will be needed while describing the co-training of the model (discussed later). Confidence estimate is defined as the ratio of weight of the edges leading to the stance of a user, divided by the sum of all edge-weights for that user:

$$c_j^I = \begin{cases} \frac{\sum_{k=1}^n w_{jk} * \mathbf{1}_{s_k = s_j^I}}{\sum_{k=1}^n w_{jk}}, & \text{if } s_j^I \neq 0 \\ 0, & \text{otherwise} \end{cases} \quad (4.5)$$

where c_j^I is the confidence in the estimated stance s_j of user j .

4.3.2 Self-Trained Text Classifier

As each user in our dataset has text features (from tweets), it is possible to infer the stance of users based on their tweets. Many text classifiers can be used for this task. However, given that we have plenty of unlabeled data, it is natural to ask if we can pick a classifier that uses unlabeled data in training. Nigam et al. explored this idea and proposed self-training [94].

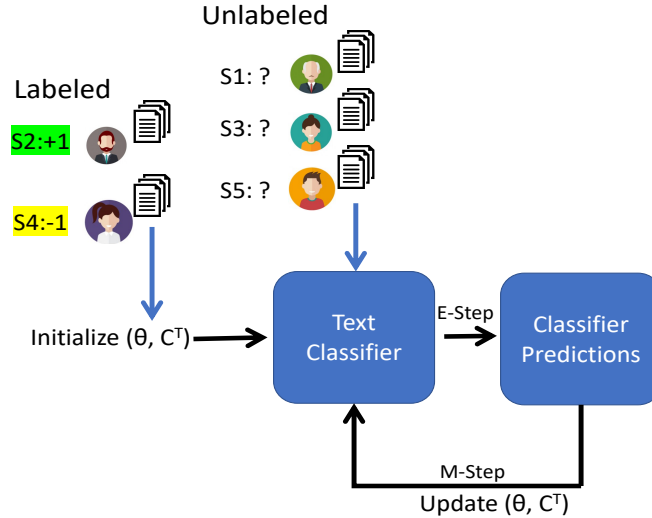


Figure 4.3: Self-trained Text Classifier. The model uses Expectation Maximization (EM) steps but is incremental. Only the users that are predicted with high-confidence (u_k s.t. $C_k^T > \theta^T$) are used in the next iteration.

In self-training, a model is first trained using an initial set of labeled examples (text from seed users) to predict the class labels of unlabeled users (E-Step). We then use the predictions of the unlabeled examples to retrain the text classifier (M-Step). Then, the trained model is again used to label the unlabeled data, and this process repeats until convergence. For a better performance, we again use a confidence threshold function that only allows to use the predictions above a certain threshold to be used as training examples.

We show the steps of training the text classifier in algorithm 2. Though the text classifier only predicts one label for a text sample (obtained from a tweet), because users can have multiple tweets, multiple predictions (one for the text from each tweet) can be used to quantify the

Algorithm 2 Self-Training Text Classifier

Require: X_l labeled tweets and X_u unlabeled tweets

```
1: function SELF-TRAIN-ITERATE( $X_l, X_u$ )
2:   Train classifier  $f_{text}(\theta)$  using  $X_l$ 
3:   for until convergence do
4:     E-Step
5:     Estimate text stance  $f_{text}(s_j|x_j; \theta)$ 
6:     Estimate users stance  $S^T$  ▷ see Eqn. 4.6
7:     Estimate confidence  $C^T$  ▷ see Eqn. 4.7
8:     Expand  $X_l \leftarrow X_l + \delta X_l$  ▷ see Eqn. 4.8
9:     M-Step
10:    Update  $\theta$  using new  $X_l$ 
11:  end for
12:  return ( $S^T, C^T$ )
13: end function
```

confidence in the stance estimation of a user. In each iteration, the number of ‘pro’ text and ‘con’ text a user has, determines the stance S^T and prediction confidence C^T . This step can be formulated as:

$$s_j^T = \begin{cases} 1, & \text{if } \frac{\sum_{k=1}^m \mathbf{1}_{s_k > 0}}{\sum_{k=1}^m 1} > f_t(\theta^T) \\ -1, & \text{elif } \frac{\sum_{k=1}^m \mathbf{1}_{s_k < 0}}{\sum_{k=1}^m 1} > f_t(\theta^T) \\ 0, & \text{otherwise} \end{cases} \quad (4.6)$$

$$c_j^T = \begin{cases} \frac{\sum_{k=1}^m \mathbf{1}_{s_k > 0}}{\sum_{k=1}^m 1}, & \text{if } s_j^T > 0 \\ \frac{\sum_{k=1}^m \mathbf{1}_{s_k < 0}}{\sum_{k=1}^m 1}, & \text{elif } s_j^T < 0 \\ 0, & \text{otherwise} \end{cases} \quad (4.7)$$

where s_k is the stance of the k^{th} tweet of the user j who has a total of m tweets. To expand the training set, we only use the users who pass the threshold criterion and for whom stance is not zero:

$$\delta X_l \leftarrow D_j \in s_j^T \neq 0 \quad (4.8)$$

where D_j is the set of tweets associated with user u_j for whom stance $s_j \neq 0$.

4.3.3 Co-Training of Label Propagation Model and Text Classifier

In our co-training setting, the labeled seed hashtags are used to obtain the initial set of labeled users. The labeled users and the unlabeled users are the input to the algorithm, which iterates till it converges, i.e., no change in user labels. We explain the steps using user-hashtag network (see Alg. 3), but the steps transfer well to user-retweet or combined user-hashtag and user-retweet network.

Algorithm 3 Co-Training: Joint training of hashtag based label-propagation model and text classifier

Require: T is the set of tweets collected

```
1: function CO-TRAIN SOCIAL NETWORKS( $T$ )
2:   Extract User-Text  $D$ 
3:   Extract User-Hashtag Network  $H$ 
4:   Label Seed hashtags ▷ e.g, #Prochoice +1
5:   Get Seed users ( $UL$ )
6:   Get Unlabeled users ( $UU$ )
7:   while until convergence do
8:     STEP 1: Label Propagation
9:     Label hashtags ( $H$ ) using  $UL$ 
10:    Predict the Stance  $S^I$  of  $UU$  using  $H$ 
11:    Estimate stance confidence ( $C^I$ )
12:    STEP 2: Text based classification
13:    Train a text classifier  $f_{text}(\theta)$  using  $UL$ 
14:    Use  $f_{text}(\theta)$  to Predict  $UU$  stance  $S^T$ 
15:    Estimate stance confidence ( $C^T$ )
16:    STEP 3: Update  $UL$ 
17:     $L=LabelMixing(S^I, C^I, S^T, C^T)$ 
18:     $UL= UL + L$ 
19:  end while
20:  return  $UL$ 
21: end function
```

In each iteration, the label propagation algorithm uses the users-hashtag network and propagates users' stance-labels to unlabeled hashtags, and then propagates the hashtags' stance-labels to unlabeled users. Therefore, in an iteration, the label propagation model uses the user-hashtags network to predict labels of the unlabeled users. Similarly, the text classifier uses the stance-labels of the labeled users and their tweets to retrieve a set of labeled text examples. The text examples and their labels are then used to train the text-classifier model. The trained text classifier then predicts the stance labels of text used by unlabeled users, which is then used to get the text-based stance and the confidence (in stance prediction) of the unlabeled users.

At the end of each iteration is a label-mixing step. In the label-mixing step, the predictions and the confidence in predictions of the models are used to expand the training set. For this step, we use a common-set approach in which a common-set of labeled examples (say $u_1, u_2, \dots, u_k \in UL$) is used to train classifiers. In every iteration, the top 5 % confident predictions of unlabeled data, which exceed the confidence thresholds, are used to expand UL . The newly labeled users and labeled users from prior iterations are used as the new training set.

For the final stance prediction, we create a joint model that combines the predictions of different models using the confidence scores to create a classifier (as described below):

$$s_j = \begin{cases} 0, & \text{if } C_j^T = C_j^I = 0 \\ s_j^T, & \text{elif } C_j^T \geq C_j^I \\ s_j^I, & \text{Otherwise} \end{cases} \quad (4.9)$$

4.4 Experiments and Results

In this section, we evaluate the proposed method on a dataset with three topics. We describe the dataset, the data preprocessing steps, seeds hashtags and seed users, details on training and hyper-parameter optimization, baseline models to compare the performance of the proposed methods, and finally discuss and visualize the results. Towards the end, we also discuss the effect of different seed hashtags and the effect of hyper-parameter selection.

4.4.1 Dataset

In Bias-watch [81], a study to understand opinion bias on social media, the authors built a human-labeled dataset on three topics, namely ‘gun-control’, ‘abortion’, and ‘obamacare’. Because opinion bias and stance are used in similar context, we use this dataset for evaluating our algorithms. We only use the pro/con labels for evaluation, ignoring the neutral users. We summarize the users in Table 4.1 and their labeled stance in Table 4.2.

Table 4.1: Topics summary

Events	Users	Tweets	RTUsers	Endtags
Guncontrol	70387	117679	15635	5505
Obamacare	67937	123320	14807	7376
Abortion	111463	173236	26818	9784

RT users shows the number of users that were retweeted in data and the information is used create the user-retweet graph. Similarly, endtags show the number of unique hashtags used at the end of tweets, and are used to build the user hashtags networks. We use endtags as prior research has shown that hashtags that appear at the end convey stance signal [34].

Table 4.2: Labeled users summary

Events	Neutral	Pro	Anti	Total
Guncontrol	60	156	288	504
Obamacare	33	108	363	504
Abortion	55	169	280	504

4.4.2 Data Pre-processing

Most prior research on users’ bias and community polarization have used user-user networks. Working with graphs of single entity type (i.e. user) makes the problem formulation simpler. However, converting bipartite graphs to single mode graphs also results in the loss of information. For example, for understanding user’s bias in the gun-control conversation, converting a user-hashtag network to user-user network with edges as common hashtags results in losing the discriminating signal of various hashtags. This is evident as the hashtag ‘#GunControlNow’ and ‘#GunControl’ would be treated equally even though the first gives a much stronger stance signal. To improve this, our problem formulation includes bi-partite graphs.

We categorize the two types of dataset that can be built from users tweets.

1. Users-text: Here we only include the text in users’ tweets after removing any hashtags and retweets.
2. User-interaction networks: Tweets could also be used to extract bi-partite graphs e.g. the graph created by users and hashtags with edges as the count of usage.

Prior research and our experiments show that hashtags at the end are more likely to carry stance information [34], therefore, our experiments only used user-endtag networks. In the paper, when we refer to hashtags, we only mean hashtags that appear at the end of the tweets.

4.4.3 Seed Hashtags and Seed Users

In semi-supervised learning, a small fraction of data points (seed users) are labeled in the beginning. To get seed user labels, we label a few (two to four) popular hashtags which we call as seed hashtags (details in Tab. 4.3). For example, for the topic ‘abortion’, we labeled $\{\#prolife : -1, \#standAlife : -1, \#prochoice : +1, \#reprorights : +1\}$. The labels given by these seed-hashtags are propagated to users using the label propagation algorithm. These seed labeled users are noisy labels (not ground truth) and these labeled users are often only a small fraction of all users.

We describe the hashtags used as seed hashtags to train the models in Tab. 4.3. Using seed hashtag, one can get the seed users by using label propagation model on the user-hashtag networks. The number of seed users available for the three datasets is shown in the right column. The number varies across dataset, and as we can observe, ‘Obamacare’ has the most number of seed users ‘Gun-control’ dataset has the least number of seed users.

4.4.4 Training and Hyper-parameter optimization

We have five parameters to optimize $\{k, p, \theta^I, \theta^U, \theta^T\}$ where k is the the count of hashtags to use, p is the count of retweets to use and others are model parameters. To identify the right value for these, we tried these parameters on the ‘guncontrol’ dataset and then use the best parameter on other datasets and in the rest of the experiments. We show the performance of the models for different values of model parameters in Figures 4.4, 4.5, and 4.6.

Table 4.3: Seed hashtags, their labels, and the count of seed users obtained using seed hashtags

Dataset	Seed Hashtags	No. of Seed Users
Guncontrol	#guncontrolnow: Pro, #endgunviolence: Pro, #2ndamendment: Anti, #secondamendment: Anti	Pro:782, Anti:321
Obamacare	#uniteblue: Pro, #ilikeobamacare: Pro, #defundobamacare: Anti, #dontfundit: Anti	Pro:1342, Anti:3883
Abortion	#prochoice: Pro, #reprorights: Pro, #prolife: Anti, #stand4life: Anti	Pro:499, Anti:2183

As we can observe in the plot (see Fig. 4.4), θ^U has an optimal value for 0.0 and θ^I is good at 0.1. When we use θ^I that decreases over iterations, most values do well near the end of the iterations. The text classifier only has one parameter θ^T . Like for the label-propagation models, we again tested the text classifier for different values of the parameter, and when the parameter values decrease over iterations on one dataset, and used the best parameter on other datasets. As shown in Fig. 4.6, in the plot on the left, we find the $\theta^T = 0.6$ works well. However, when decreasing parameter is used over iterations, values in range $[0.0, 0.6]$ perform well.

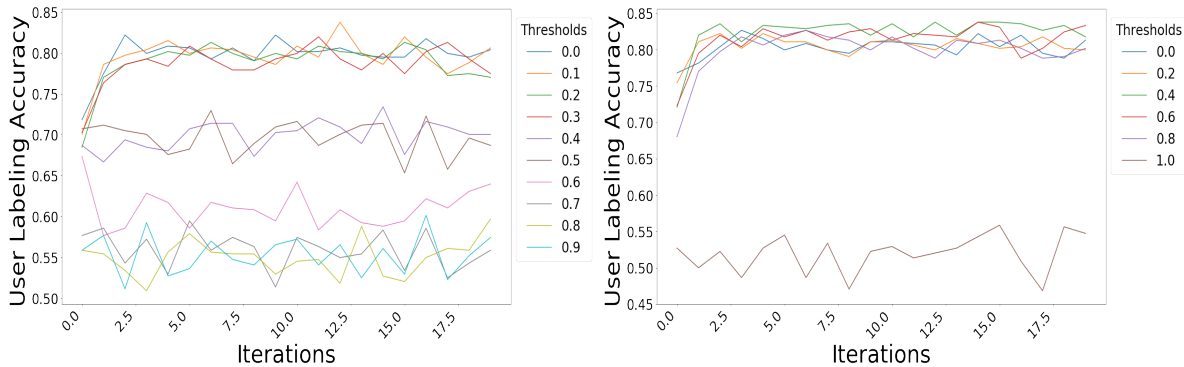


Figure 4.4: Accuracy of the label propagation model for different values of θ^I (left) and θ^U (right). The trends are for the 'gun-control' dataset.

Note that with LTMWDT which uses a uniformly decreasing threshold over iterations, there is a range of the parameter values that work well. We pick the optimal parameter values as: $\{k = 250, p = 5000, \theta^I = 0.1, \theta^U = 0.0, \theta^T = 0.6\}$.

4.4.5 Baseline Models

We pick some classifiers that are known to perform well on other NLP tasks namely, 'Support Vector Machine (SVM)', 'Neural Networks (NN) with varying size of hidden layers' and 'XG Boost (XGB)' as the baseline models. We provide more details on these models next. Details on performance of the baseline models is described in the first part of Tbl. 4.4. For co-training we use SVM because it is fast to train and also performs well.

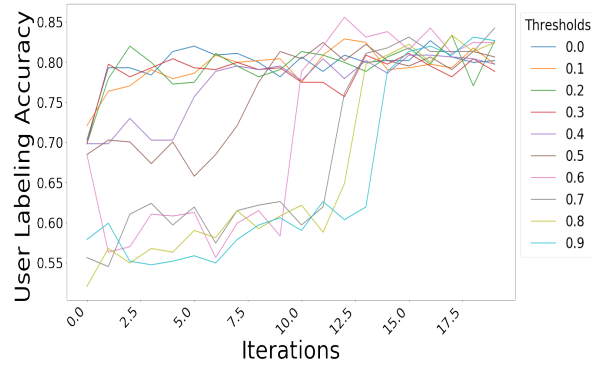


Figure 4.5: Accuracy of the label propagation for different values of θ^I when θ^I decreases over iterations. The trend is for the 'gun-control' dataset.

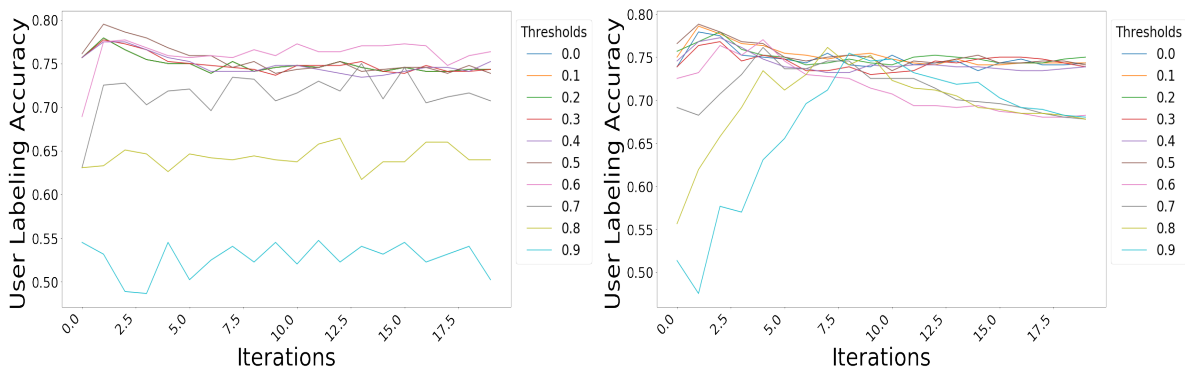


Figure 4.6: Accuracy of the text-classifier model at different threshold values θ^T , and when the threshold θ^T decreases over the iterations. The trend is for the 'gun-control' dataset.

We consider some of commonly used NLP classifiers namely 'Support Vector Machine (SVM)', 'Neural Networks (NN) with varying size of hidden layers' and 'XG Boost (XGB)' as the baseline models. We do not consider network based classifiers as baselines as we are interested in the inductive training setup that could generalize to new data (even when networks are not available). For a fair comparison, all classifiers use the same set of input. The input is composed of TF-IDF features extracted after vectorizing the count of words as uni-grams and bi-grams obtained from the pre-processed text. All text classifiers were built and trained using 'sklearn'² python package with 'CountVectorizer' and 'TfidfTransformer' and the classifier in a pipeline.

Support Vector Machine: The support vector machine classifier (SVM) was built using the 'SGDClassifier'³ with a 'hinge' loss that leads to a linear SVM and was trained for the maximum of 15 iterations. Alpha value which is used as the regularizer was used as a parameter with values in $\{1e - 2, 1e - 3\}$.

²<https://scikit-learn.org/stable/>

³[1 + /.../sklearn.linear_model.SGDClassifier.html](https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.SGDClassifier.html)

We also used an SVM model tri-grams as inputs, in addition to uni-grams and bi-grams. We name this model ‘svm123’. The other parameters of the model stay the same as the SVM as described above.

Neural Networks: We used two neural networks with slight different configurations to understand not just the general performance of neural networks but also how the performance varies by changing the number of hidden units and the number of layers. We call them ‘NN88’, and ‘NN8168’ where ‘NN’ indicates multi-layer perceptron classifier ⁴ with ‘relu’ activation function.

NN88 is constituted of two hidden layers each with eight neurons each. NN8168 is comprised of three hidden layer, first with eight units, second with sixteen units and third with eight units. All NNs were trained to optimize the log loss using the ‘lbfgs’ weight optimizer with an alpha value of $1e - 5$. The rectified linear unit function (‘relu’) was used as the activation function in all the models.

XG Boost: In addition to SVM and MLP, we also tried the XGBoost (XGB) classifier [20], a classifier commonly used in data mining competitions and have shown performance benefits over other classifiers. We used the sklearn based version of XGB ⁵ that takes the same input as the other two classifier and therefore, fits our training pipeline. For XGB, we used ‘gbtree’ as booster with a maximum depth of 2 and the learning rate of 0.1.

4.4.6 Results and Discussion

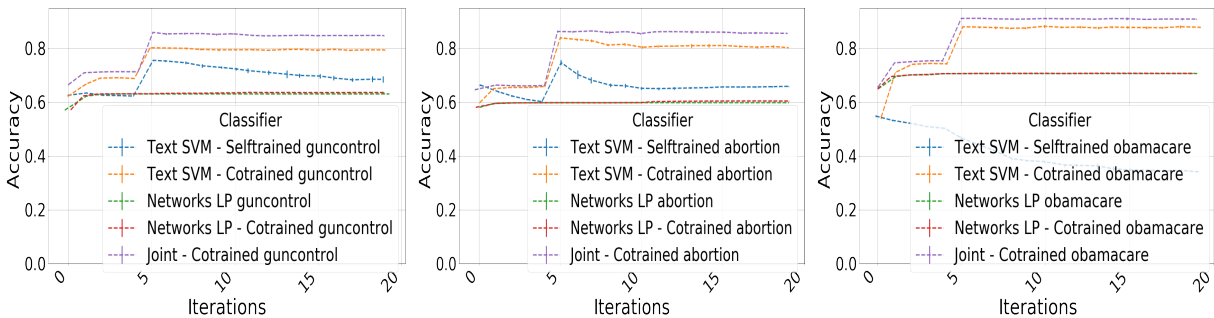


Figure 4.7: Classifiers performance over iterations. Networks LP implies label propagation model that uses both hashtags and retweets networks.

We use accuracy as the metric to compare models. However, as not all classifiers are able to label (reach) all users (e.g. a label propagation model cannot label users that are not connected in the graph i.e. have not used any hashtags or have not retweeted.), and also the models’ reach changes with iterations, so we use a modified definition of accuracy. This new accuracy assigns

⁴[1 + ...neural_network.MLPClassifier.html](#)

⁵<https://xgboost.readthedocs.io/.../module-xgboost.sklearn>

a random accuracy score of 0.5 for the users that are unreachable (not labeled) by the classifier. We define:

$$Accuracy = ULA * R + 0.5 * (1 - R) \quad (4.10)$$

where R is the fraction of users that the model is able to label and ‘User Labeling Accuracy’ (ULA) is the conventional accuracy metric for labeled users. We show the models’ accuracy over iterations in Fig. 4.7 and compare the performance of different models in Tbl. 4.4.

As we can observe in the figure, the iterative training process improves the accuracy of almost all classifiers. However, most improvement is observed for the text classifier and the joint model. Self-trained text classifier (for gun-control and abortion) improves till the 6-7 iterations after which the performance degrades. On further inspection, we find that the text classifier and the networks based label propagation models have complementary strengths. The network based models have low reach but their precision is rather high. In contrast, the text classifiers are able to label all users (as the dataset is collected using text keywords), but have lower accuracy as we only use the signal given by the hashtags.

We now discuss the results shown in the Tbl. 4.4. As we can observe, in almost all cases, co-training results in better performance. For ‘Guncontrol’ dataset, the joint models of ‘Hashtags + Retweets + Text’ has the highest accuracy of 0.85. The ‘Retweets +Text’ model also has the same accuracy, which is slightly better than ‘Hashtags + Text’ based model. If we compare the text classifiers, the self-trained text classifier (SVM) has an accuracy of 0.68. The accuracy improves to 0.80 when co-trained. This is a significant improvement especially considering that the only training signal that is available to the models are the hashtags. If we compare the hashtags based LP models, the accuracy improved from 0.49 to 0.54, a rather minor improvement. If we look at the results for ‘Abortion’ dataset, the best performer is again the joint model of ‘Hashtags + Retweets and Text’ with an accuracy of 0.86. The best text classifier is the one that is co-trained with ‘hashtag +retweets’. The accuracy of this model improves from 0.60 to 0.80. If we look at the LP models, the performance improvement is rather small (0.54 from 0.49 for hashtags and 0.60 from 0.53 for Retweets).

If we consider the results for ‘Obamacare’ dataset, the best performance is shared by ‘Retweets and Text’ cotrained and ‘Retweets, hashtags and text’ cotrained. The text model significantly improves from 0.54 (self-trained) to 0.82 (co-trained). Again, the improvement in LP based models is rather low (0.55 from 0.49 for hashtags, and 0.63 from 0.55 for retweets).

To summarize, it is clear that the co-training models have clear advantage over self trained model. The joint model is the best performer for all three datasets. The co-trained text classifier shows significant improvement (17.6% on Guncontrol, 31.6% on Abortion, 161.7% on Obamacare) in their performance over the self trained models. For Obamacare, the higher increase is partially because of the decrease in the base accuracy as a result of self-training process. For other two datasets, self-training improved the performance.

4.4.7 Visualization of Results

In this section, we try to visualize the outcome of the models. We use only the gun-control dataset for this discussion. In Fig. 4.8, we show the bipartite network of users on the left, and on

Table 4.4: Performance of models on different datasets. LP implies label propagation, and bold font indicates the best for a dataset.

Classifier Type ↓ Dataset →	Guncontrol (Accuracy)	Abortion (Accuracy)	Obamacare (Accuracy)	Mean (Accuracy)
<i>Self-trained Text Classifiers</i>				
Text SVM	0.68	0.60	0.34	0.54
Text SVM123	0.68	0.59	0.33	0.53
NN88	0.54	0.62	0.29	0.48
NN8168	0.53	0.38	0.35	0.42
XGB	0.66	0.60	0.54	0.60
<i>Network Label-Propagation Models</i>				
Hashtags LP	0.49	0.49	0.49	0.49
Retweets LP	0.53	0.53	0.60	0.55
Hashtags and Retweets LP	0.56	0.54	0.63	0.58
<i>Hashtags + Text Cotrained Models</i>				
Text SVM Cotrained	0.81	0.78	0.85	0.81
Hashtags Cotrained	0.54	0.54	0.57	0.55
Hashtags Text Joint	0.83	0.82	0.84	0.83
<i>Retweets + Text Cotrained Models</i>				
Text SVM Cotrained	0.80	0.79	0.89	0.82
Retweets Cotrained	0.62	0.60	0.69	0.63
Retweets Text Joint	0.85	0.85	0.91	0.87
<i>Hashtags + Retweets + Text Cotrained Models</i>				
Text SVM Cotrained	0.79	0.80	0.88	0.82
Hashtags and Retweets Co- trained	0.64	0.60	0.71	0.65
Hashtags and Retweets Text Joint	0.85	0.86	0.91	0.87

the right, we show the user-user network based on co-retweets. There are links connecting the dots which will be more clear after zooming. As we can see in the plot, there are many small groups of closely connected users. These are the groups formed by users using the same set of hashtags or user retweeting a similar set of users. There also many users away from the central region that are not connected with other users. These are possibly the users that only tweeted once or twice, and did not use any popular hashtags. Note that the plot does not show users who have not used hashtags or have not retweeted, and this set of users in the dataset is larger than the set shown in the plot. Therefore, this visual analysis of the plots just gives a partial picture.

Next we show the stance of users predicted by the models. In Fig. 4.9, we show users color based on the stance predicted by the text-based stance classifier (self-trained) on the left, and joint hashtag + retweet+ text based stance prediction on the right. Red colored nodes are users having pro-stance and green colored nodes have anti-stance. As we can observe in the plots, the

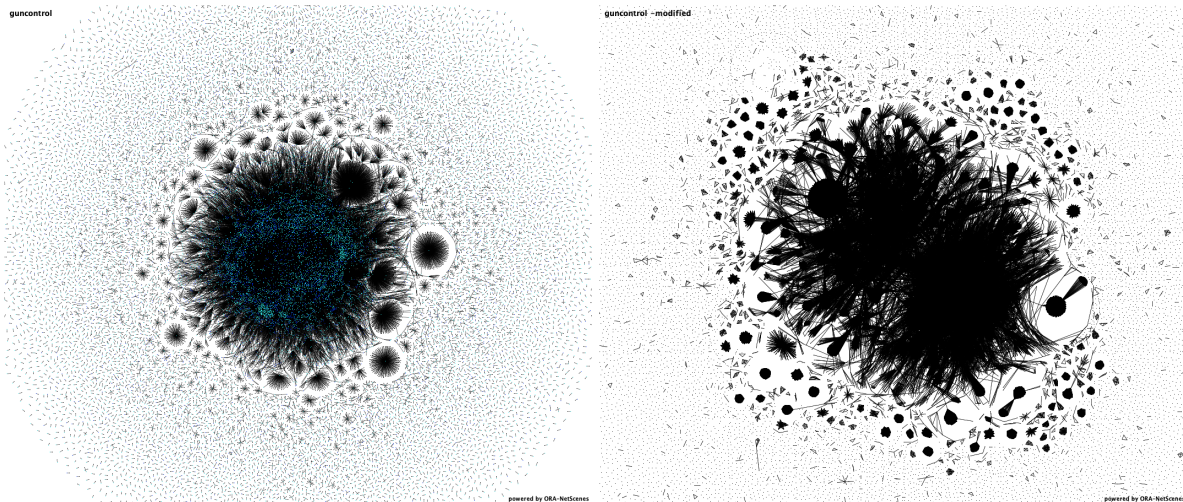


Figure 4.8: Forced directed layout user-endtag + user-retweet bipartite network (on left) using guncontrol data. Black dots represent users, blue dots represent endtags and light blue represents retweets. Links show the usage of endtags and retweets by users. On the right is the user-user network based on co-retweets. Co-retweets implies users who are retweeting the same set of users are connected by a link. Best if seen on a computer after zooming-in.

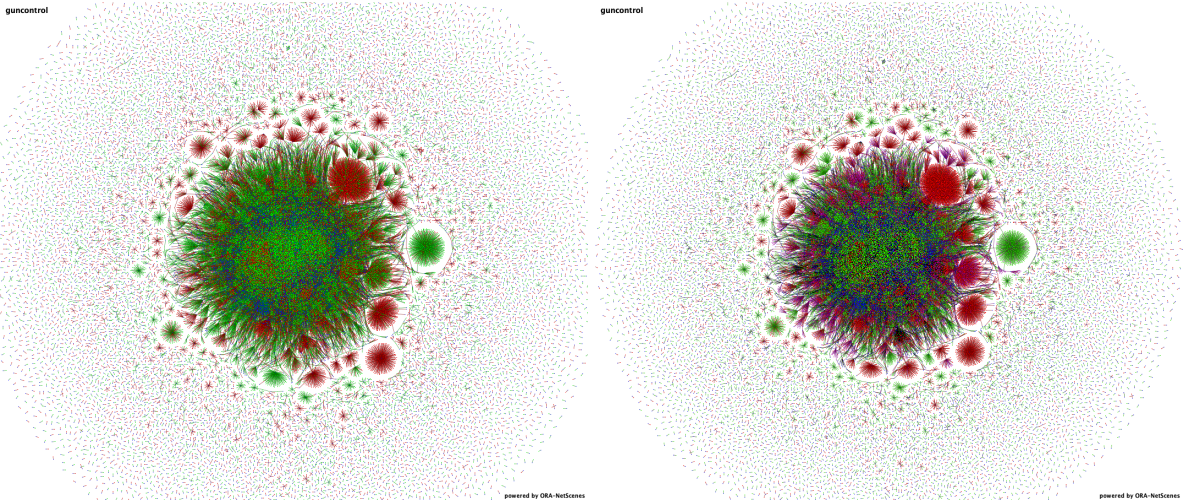


Figure 4.9: Forced directed layout for text based stance prediction (left) and joint hashtag + retweet+ text based stance prediction (right) on the guncontrol dataset. Red colored nodes are users having pro-stance and green colored nodes have anti-stance. Blue nodes represent endtags and retweets. Pink nodes shows that users for whom the joint model has no clear stance prediction. Best if seen on a computer screen after zooming-in.

joint model results in a more homogeneous plot where clusters are labeled by one color. This is possibly because, co-training the models, results in more coherent stance labels for the groups.

We visualize the output of stance prediction for guncontrol in Fig. 4.10 using the user-user network in the guncontrol dataset created using common retweets. The stance prediction for

self-trained text based classifier is shown on the left, and the co-trained joint model (hashtag + retweet+ text) based stance prediction is shown on the right. Red colored nodes are users having pro-stance, green colored nodes have anti-stance, and yellow nodes are the users for whom the joint model has no clear stance prediction. As we know from the results table, the self-trained SVM classifier is 68% accurate and the joint model is 80% accurate. The 12% improvement has lead to significant difference in the visualization, and as we can observe in the figure, the joint model is able to shown the polarized nature of the conversation which is not visible using the self-trained text classifier result.

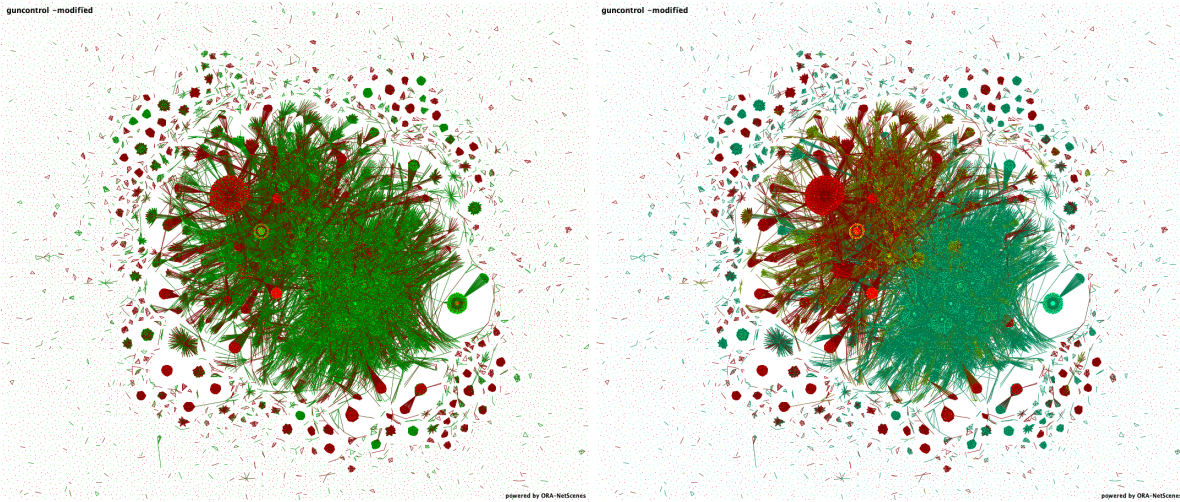


Figure 4.10: Forced directed layout network visualization for the text based (self-trained) stance prediction model (left), and the co-trained joint model (hashtag + retweet+ text) based stance prediction (right). We used the user-user network based on common retweets in the guncontrol dataset for this visualization. Red colored nodes are users having pro-stance, green colored nodes are users having anti-stance, and yellow nodes are users for whom the joint model has neutral stance prediction. As described in the paper, we only used the training signal given by four labeled hashtags to learn the stance of users. Best if seen on a computer screen after zooming-in.

To summarize the visualizations, as observed in prior research, the retweets based networks better reflect the polarized nature of social-media conversation on controversial topics. Also it is evident that the joint models are much better at dividing a controversial discussion into two groups. Given the our approach only used weak hashtag labels as input signal, being able to identify the polarized groups could be useful for many tasks e.g. estimating the polarization in a discussion.

4.4.8 Effect of Different Seed Examples

Given that we only use the stance signal given by a few seed hashtags, it is natural to ask how generalizable are the results obtained in the previous sections. To understand the differences because of choosing different seed hashtags, we conducted experiments by selecting a subset of the labeled hashtags used. We show the selected hashtags and the resulting accuracy of classifiers in Tbl. 4.5.

Table 4.5: Performance of co-trained joint model on bias-watch datasets with different seed hashtags. LP implies label propagation.

Seed Hashtags	Classifier Type	Metric→ Dataset↓	Accuracy	Precision	Recall	F1- Score	Labeled Users (Fraction)
#stand4life:+1, #prochoice:-1	Hashtags + Retweets + Text Joint	abortion	0.38	0.38	0.98	0.54	0.98
#prolife:+1, #prochoice:-1	Hashtags + Retweets + Text Joint	abortion	0.83	0.77	0.78	0.78	0.94
#reprorights:+1, #stand4life:-1	Hashtags + Retweets + Text Joint	abortion	0.83	0.76	0.80	0.78	0.94
#endgunviolence:+1, #secondamendment:-1	Hashtags + Retweets + Text Joint	guncontrol	0.68	1.00	0.09	0.16	0.96
#2ndamendment:+1, #guncontrolnow:-1	Hashtags + Retweets + Text Joint	guncontrol	0.81	0.87	0.55	0.67	0.92
#2ndamendment:+1, #endgunviolence:-1	Hashtags + Retweets + Text Joint	guncontrol	0.68	1.00	0.10	0.18	0.96
#ilikeobamacare:+1, #defundobamacare:-1	Hashtags + Retweets + Text Joint	obamacare	0.80	0.81	0.16	0.26	0.95
#uniteblue:+1, #defundobamacare:-1	Hashtags + Retweets + Text Joint	obamacare	0.90	0.87	0.67	0.75	0.92
#uniteblue:+1, #dontfundit:-1	Hashtags + Retweets + Text Joint	obamacare	0.91	0.84	0.75	0.79	0.90

As we can see in the table, the classifiers accuracy are not entirely independent of the seed hashtags. For example, for abortion, if we select only ‘#stand4life’ and ‘#prochoice’, the models accuracy decreases drastically to 0.38. In contrast, other selections like ‘#prolife’ and ‘#pro-choice’ does not make much of a difference in the trained classifiers’ accuracy (0.83). A similar pattern is visible on other datasets as well. For guncontrol, the accuracy varies from 0.68 to 0.81, and for obamacare, the accuracy varies from 0.80 to 0.91. One possible reason for such difference is the popularity of the chosen hashtags. For example, if one selects hashtags that are not frequently used in the dataset, it is less likely to result in good classifiers. The extreme of this is when one selects a hashtag that has never been used in the dataset, and therefore, the selected seed hashtags leads to no useful signal in the model resulting in random classifiers. On the other extreme, if one uses a very popular hashtag e.g., ‘#guncontrol’ that gives no clear stance signal, this would again fail, as there is not enough signal to differentiate the two sides of the conversation.

On the brighter side, it is evident that as long as we have a few (two or more) hashtags for the both sides, the classifiers are reasonably good. On further inspection, we find that as long as the seed hashtags are able to generate a reasonable number of labeled users (say over 500), the classifiers perform reasonably well. So our suggestion would be to look at the top few hashtags in the dataset and pick a few representing both sides of the conversation. It would be good it could get an estimate of the trained models’ accuracy based on the selected seed hashtags. There has been some work on using unlabeled data and multiple classifiers to get the estimate of classifiers’ accuracy (see [98]), but we leave it for the future work.

4.4.9 Effect of Hyper-Parameter Selection

We again re-look at the parameter values and their impact on the performance of the models. As the parameters decrease over iterations, as we have seen earlier, this reduces the need for a through parameter tuning. However, it is still important to understand the effect of using different parameter values. As we mentioned earlier, we have three parameters: user threshold θ^U , interaction threshold θ^I , and text threshold θ^T . Note that we picked a uniformly decreasing threshold function that has shown to minimize the impact of selecting a particular threshold but still results in good accuracies. We test the trend of the ‘hashtag+retweet+text’ joint model for different values of these parameters. We show the trend of performance of the joint models for different values of user thresholds and interaction thresholds in Fig. 4.11.

In the Fig. 4.11, on the left, we show the models’ accuracy trend for different user-threshold values θ^U . As we can observe, the performance is rather stable only changing between 0-2% points in accuracy. The trend is similar for all three datasets showing that the parameter selection had minimal impact. The plot on the right in the same figure, shows the trend for different interaction thresholds θ^I . Again we observe that the accuracy is mostly stables but degrades around the value of 1.0. The stability of parameters is a good news as the parameters need not be tuned independently for different datasets. Also, even for a selected parameter, there is not much of difference in the trend across different datasets.

To summarize, it appears that using a decreasing threshold function reduces the impact of parameter selection, and therefore, our approach should have a similar performance on other datasets as well.

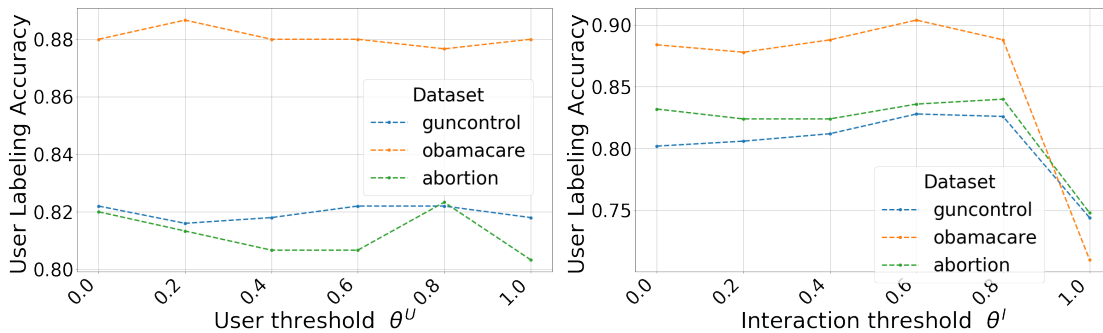


Figure 4.11: Trend of performance of the joint models for different values of user thresholds and interaction thresholds.

4.5 Related Work

4.5.1 Stance Learning

Stance learning – which aims to predict opinion of social-media users about a proposition or a topic on interest – is a sub-field of opinion mining. Researcher have explored stance in many different contexts e.g., stance mining has been actively used in online debates [95, 110]. Research on stance has also appeared in the context of political positioning e.g. liberal vs conservative.

[1, 26, 129]. However, in most of these work, stance learning has been attempted as a supervised learning problem (as proposed in [91]). Like other supervised learning problems, if labeled data is not available on the topic on interest, finding users' stance about the topic is difficult. In the research, we extend the stance learning to the semi-supervised domain using weak labels.

4.5.2 Weakly Supervised Machine Learning for Stance Mining

Though weak labels are common in many areas of text mining (sub-event discovery [130], aspect-based opinion mining [77] and sentiment mining[44]), for opinion mining tasks, most work use labeled datasets [59, 91]. Misra et al. used a set of hashtags to build a topic-specific training corpus [89] which was created in a semi-supervised manner by hand-selecting a set of seed hashtags. In another related work, Lu et al. proposed a opinion bias propagation framework to find the strong partisan members and use them to find partisanship of other members [81]. Our work tries to improve the text-based stance classifier (which removes the need for networks) by splitting the tweets information into 'text' and 'networks', and therefore can be used in an inductive fashion (unlike pure networks based optimization approaches like [81]). Strategies to improve a text classifier is likely to be more useful to newer data that does not have networks, which is common.

4.5.3 Co-Training

Blum and Mitchell [13] introduced the idea of co-training to improve the performance of a learning model when some labeled data and a large amount of unlabeled data is available with two distinct views. A number of researchers extended this idea, e.g., Han et al. [47] proposed Co-teaching which uses noisy labels to train deep neural networks. Very recently, [132] used co-training to identify users with disabilities. We follow a similar idea, however, 1) in our problem, even the label data are noisy (hashtgas), 2) we apply co-training to an entirely new task of stance learning.

4.6 Conclusion and Future Work

In this research, we proposed a new semi-supervised learning approach that uses two to four labeled hashtags, and plenty of unlabeled social-media data to train two stance classifiers. The proposed co-training approach uses different interactions (users' text, user-hashtags, and users-retweets) extracted from users' tweets, and learns from the complementary information in these views, to train better classifiers. Based on experiments on three human-labeled datasets, we estimate that with 2-4 hashtags as weak labels, the text classifiers trained using our proposed approach could reach an accuracy of over 80%. Co-training results in improving the text classifier accuracy by a margin of 17% across all three datasets. As our models are trained using the stance signal given by the labeled hashtags, the approach is very flexible and, therefore, can easily extend to new topics.

We get significant improvement using the co-training method because social-networks data appear to have the same characteristics (two or more views of the same information) that is

needed for co-training to work. For example, co-training requires two or more conditionally independent views. This requirement partially translates to complementary information in views i.e., if the information available in view one is not available to view two, the co-training helps. This requirement is also true for social-networks data. In social networks, users have their preferences. For example, some users commonly use hashtags; some use retweets, and others use text messages. Therefore, retweets only base view cannot help to identify the stance of all users. In this sense, the different views provide complementary information and hence, if we use data from all views to train models in co-trained fashion, the approach can label users with varied preferences, thus improving the overall performance.

In this work, we only considered semantic networks. However, the simplicity of our approach allows extending the method to other interactions (e.g., ‘Following’, ‘Likes’). We plan to use these interactions in the future work.

Chapter 5

A Joint Network and Text based Model for Learning Stance in Conversations

5.1 Introduction

Automated ways to learn Stance – which aims to predict the stance of social-media users on controversial topics – has been broadly explored as two separate threads of research: 1) learning stance of users based on their social media posts (as in Pro/Con about a topic), and 2) learning stance taken in conversations while replying (as in favoring and denying a post) to social-media posts. Though these threads represent different aspects of stacetaking behaviour on social-media forums, it is natural to ask if they are two faces of the same phenomenon, and if so, what would be a unifying approach.



Figure 5.1: Stance in Conversations: While users have stance about topics, they also exhibit their stance while in conversation with other users. For example, in this illustrative example above, a user while replying to another users, reveals his stance by ‘denying’ to the original post.

Using a new human-labeled dataset with labels for both ‘stance of users’ from their posts’ and ‘stance in conversations’ (as described in Chapter 2), we provide empirical evidence that

these two stancetaking behaviors are indeed related (sec. 5.2). As we would expect, given a discussion on a controversial topic, a user is more likely to deny a post (while replying) to a post of another user who has an opposing stance. This pattern can in fact be used to train conversations based stance classifiers. The benefit of learning stance from conversations is that, conversations are the only modality of interaction on Twitter that allows to exhibit negative relationship. Most other interaction modalities such as ‘Retweeting’, ‘Liking’ or ‘Following’ allow to infer positive interactions. In contrast, as mentioned earlier, conversations as in ‘replies’ and ‘quotes’ (and sometimes tweeting with mentions) allow to show one-to-one antagonistic relationship. However, inferring relationship directly from text messages is challenging (as discussed in chapter 2). A part of the challenge is that labeled examples are few, so training models with a large number of parameters is difficult. Instead, we propose a weakly supervised approach that uses training labels generated using weak supervision. To learn from the pattern in conversations, we propose a multi-modal stance classifier (MMSC) that jointly learns stance from : text in users’ conversations, and users’ networks. MMSC brings two unique ideas: 1) confidence in node class (users’ stance) prediction, and 2) edge polarity (stance in conversations) prediction (as in signed networks) to improve on prior approaches to stance classification. To summarize, we propose a holistic approach of learning stance from social-media data, which in turn also leads to a better stance classifier for conversations.

This chapter is organized as follows. Before we move ahead with the idea of using users’ stance to train stance learning models on conversations, it is necessary to empirically validate the proposed correlation i.e., between ‘Stance in conversations’ and ‘Stance of Users’. We do it in the next section 5.2. We describe our joint learning model in sec. 5.3. We discuss the experiments and the results in sec. 5.4. Finally, we conclude and suggest directions for future research. Code to reproduce the experiments is available on Github ¹.

5.2 Empirical Evidence of Correlation Between ‘Stance in Conversations’ and ‘Stance of Users’

Stance of users as in ‘pro’ or ‘con’ on a topic indicates users’ preference of taking one side over other. This preference of one side over other also influences the stance that the user takes while engaging in conversations with other users. Intuitively, when engaged in a conversation on a topic, a person who has pro stance on a topic (say pro ‘gun control’) is likely to give favorable replies to other users who share the stance. In contrast, while engaging in a conversations with a user who is against gun-control, the user is more likely to have a ‘denial’ type response. As this idea is only a hypothesis, we try to validate it empirically in this section.

To validate the hypothesis empirically, we use the dataset that we built in the first chapter. The dataset has the stance of users (as in pro/con) as well as stance labels for conversations (as in favor/deny). To validate the hypothesis, we first create two groups using the conversations which we found in our dataset. The first group is composed of Twitter users who have the ‘pro’ stance, and the second group is composed of users who have ‘anti’ stance on a topic. Then, for both these groups, we get the conversations within the group and across the groups. Then

¹https://github.com/CASOS-IDeaS-CMU/Stance_in_Conversation

for these two groups, we find the number of users that have ‘favoring’ and ‘denying’ stances in conversations. We present the result of this analysis as a confusion matrices below:

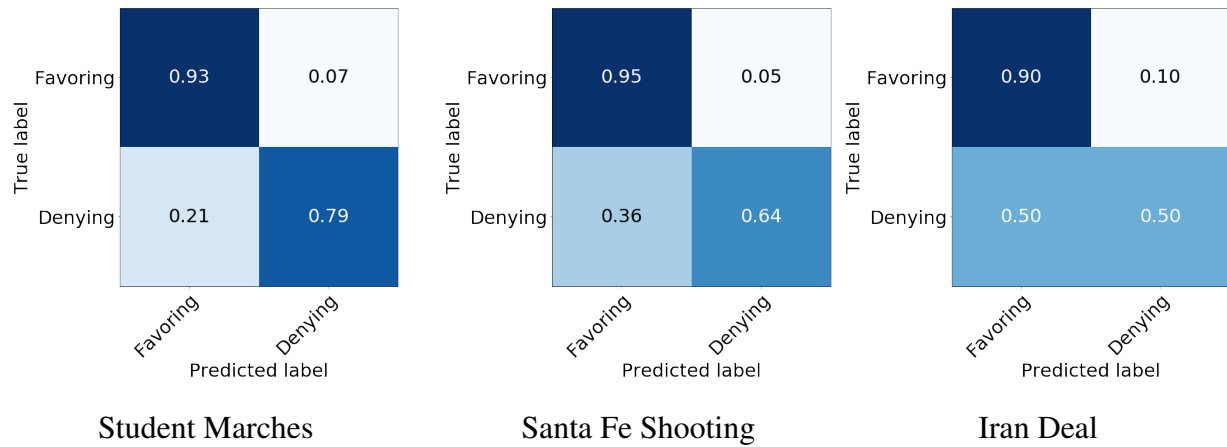


Figure 5.2: Confusion Matrix for stance in conversations (True label as Y axis) and predicted label obtained from the users’ stance on the topic (on X axis). For prediction, if the two users engaged in the conversation have the same stance, the predicted label is ‘Favoring’, otherwise it is ‘Denying’. For the plots above, we ignore the conversations that have ‘comments’ and ‘query’ as ground truth labels.

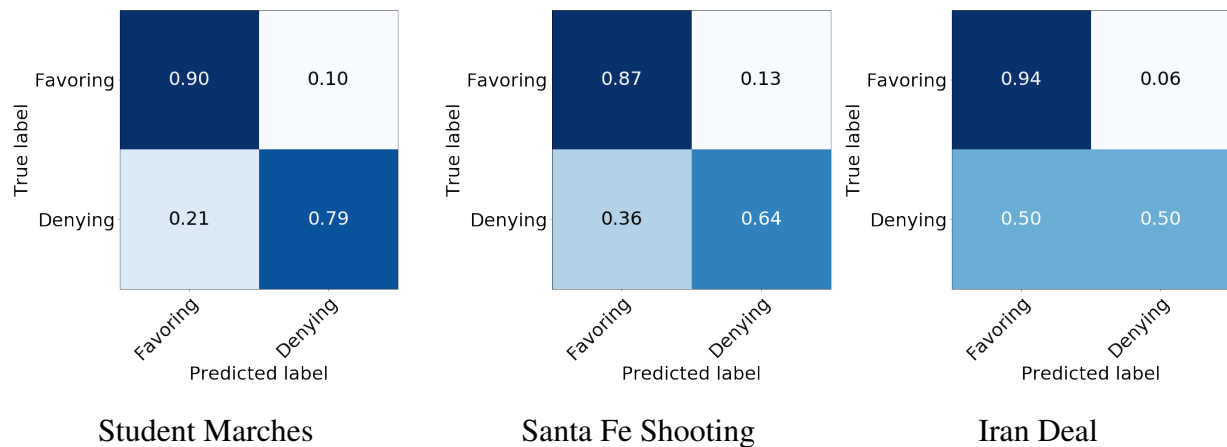


Figure 5.3: Confusion Matrix for stance in conversations (True label) and label obtained by users’ stance on the topic when ‘comment’ labels are used as ‘Favoring’. For prediction, if the two users engaged in the conversation have the same stance, the predicted label is ‘Favoring’ otherwise it is ‘Denying’.

We present the result of this analysis as confusion matrices in Fig. 5.2. As we can observe, based on the stance of the users, we are able to predict the stance in conversations (as in denying vs favoring) with a high probability for ‘Student Marches’ and ‘Santa Fe Shooting’. If two users had ‘favoring’ conversations in our dataset, with over 90% of probability the users have the same stance on the topic. For denying conversations, the two users are more likely to have opposing

stance (except for the ‘Iran deal’ topic). For ‘Iran Deal’, though we are able to get ‘Favoring’ type conversations with a high accuracy, getting ‘denying’ conversations based on users’ stance appears challenging.

The confusion matrix (Fig. 5.2) ignores ‘Comments’ and ‘Query’ type true labels. In Fig. 5.3 We show a similar trend, but in this case we use ‘comment’ type true labels as ‘favoring’. The idea behind using comments as favoring response is that two users are more likely to have a neutral conversation if they have similar stance on the topic of discussion. As we can observe, this results in a slight decrease of the accuracy. Overall, the correlation between ‘users’ stance’ and ‘stance the users take in conversations’ is strong enough to try to use users’ stance to train a model for learning stance in conversations.

5.3 Methodology

In the last chapter of this thesis (chapter 4), we showed that the stance of users can be learned using weak supervision. We want to extend the idea of weak supervision to train models to learn ‘stance in conversations’ in this chapter. The goal is to first use weak supervision to get the stance of a few seed users, and then use the seed users to train the stance classification models. Figure 5.4 shows the steps we use to train the classifiers but before we discuss the steps, let’s formulate the problem.

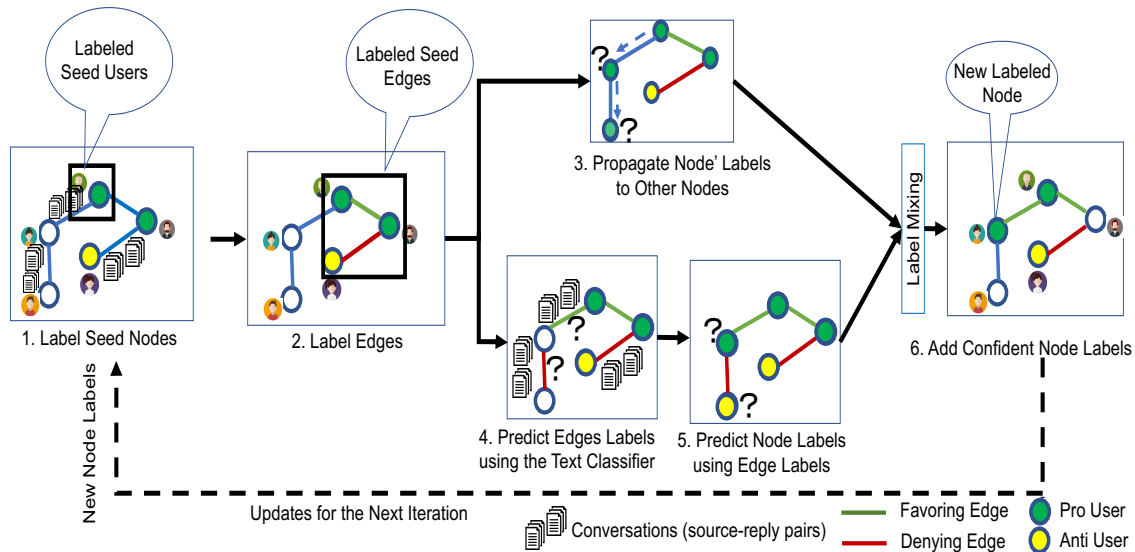


Figure 5.4: A high level illustration of the methodology. As shown in the diagram, the entire process could be divided in six smaller steps.

5.3.1 Problem Statement

Given a topic, let’s assume we retrieve a set of tweets $T = \{t_1, t_2, t_3, \dots, t_m\}$ which were tweeted by a set of users $U = \{u_1, u_2, u_3, \dots, u_n\}$ where $n \leq m$. Tweets also have additional metadata

that are used to build user-hashtags network H and user-retweet network R . Let's assume H is a weighted matrix created from k most used hashtags in the dataset. Similarly, R is a weighted matrix created using p most popular retweets in the dataset. Therefore, $H \in \mathbb{R}^{n \times k}$ matrix and $R \in \mathbb{R}^{n \times p}$ matrix. When the distinction between H and R is not critical, we use I to represent the user interaction matrix (H or R) with edge weights w_{ij} .

Moreover, two users can have conversations (one user tweets and another user replies to his/her tweet) between them. Let's define the conversations between two users u_i and u_j as C_{ij} where $C_{ij} = \{C_{ij}^1, C_{ij}^2, \dots, C_{ij}^k\}$ are k conversations (source tweet, reply tweets pair). Therefore, $C_{ij}^k = (t_s, t_r)$ where t_s is source text and t_r is reply text. These source reply pairs could have stance labels as 'Favoring' (+1) and 'Denying' (-1). We ignore 'Comments' and 'Queries' (the other two classes in the labeled dataset) in this chapter. The goal of this research is to correctly assign a conversation stance label $\{+1, -1\}$ to as many conversations as possible in the set C .

Because we use users' stance as a signal to learn stance in conversations, an intermediate step in the process of learning stance of C_{ij}^k is to learn stance of users u_i and u_j on the topic. Because the topics studied in this research are controversial, let's assume that users have either pro-stance (+1) or con-stance (-1) (same as anti-stance). Users can also have an unknown stance (0) when the stance of the user is not known. An intermediate goal of this research is to correctly assign a stance label $\{+1, -1\}$ to as many users as possible in the set U based on D, H, R, C . This assignment results in a user-stance matrix $S = \{s_1, s_2, s_3, \dots, s_n\}$ where $s_i \in \{+1, 0, -1\}$.

Fig. 5.4 show the step of training the models. Note that these steps extends the approach which was described in details in the last chapter. The main difference from the last chapter is that the approach is modified to train stance models on conversations. For this, we change the text classifier (that predicted users' stance) to predict stance in conversations (sour-reply text pairs). This lead to new steps like step 2 and step 5 for transforming between users' stance and stance in conversations.

In step 1, we label two to four hashtags to get seed labeled users. As the labeled users don't directly result in labeled conversations, we first use labeled users to derive seed labeled edges (step 2). In step 3, a bi-partied label propagation is used to get labels for other users. In step 4, we use whatever edges are already labeled to train a text classifier which takes source-reply text pairs as input, and predict the edge labels for the rest of the conversations. Then in step 5, the predicted labels of the edges are used to predict node labels. In the last step, node labels from both the classifiers are used to predict the stance of unlabeled nodes (step 6 and 7).

As in the last chapter, the proposed process uses two classifiers, 1) a node label propagation, and a 2) text classifier that uses source, reply pairs from users' conversations. Both theses classifiers are trained in a co-training fashion to reduce noise from weak labels. We elaborate on these steps next.

5.3.2 Step 1: Label Seed Nodes

We use two to four hashtags (that appear at the end of the text in tweets) as the weak signal to label a few initial seed users. This step uses the label propagation algorithm on the user-hashtag bipartite network.

$$S^I = H \cdot \tilde{h} \tag{5.1}$$

where S^I is the resulting labeled users and \tilde{h} indicates the hashtags vector with a few labeled tags as $\{+1, -1\}$.

5.3.3 Step 2: Label Edges from Seed Nodes

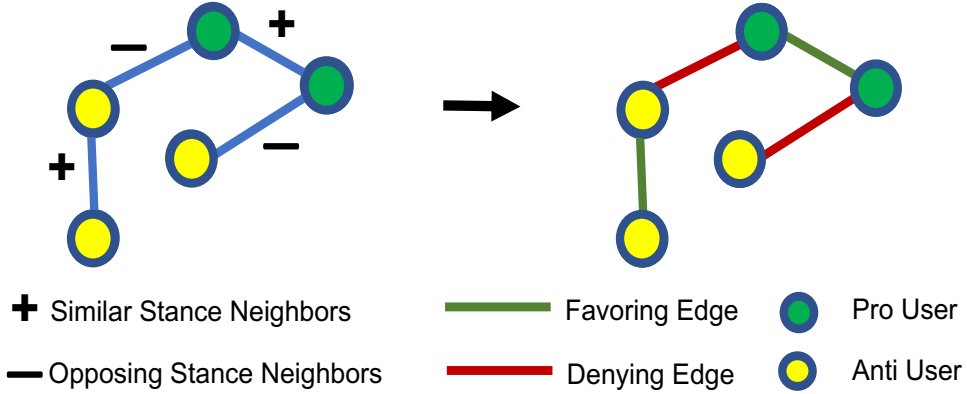


Figure 5.5: Model of predicting stance of edges (conversations) as in favoring/denying from stance of users (pro/anti)

We use the stance of users (node labels) to derive the stance on potential edges between the users (see Eqn. 5.2). For this we use a simple heuristics which is based on our empirical validation in the last section (see Fig. 5.5). If two users have the same stance (i.e. either both of them are pro or both of them are anti), the conversations between them (as in edges in the network) are assumed to be ‘favoring’. In contrast, if if two users have opposing stance, all conversations between them are assumed to be ‘denying’. This way, we get edge labels for all conversations (source, reply pair text) between any two users with non-neutral stances.

$$C_{ij}^k = \begin{cases} 0, & \text{if } s_i = 0 \text{ or } s_j = 0 \\ -1, & \text{elif } s_i \neq s_j \\ 1, & \text{otherwise} \end{cases} \quad (5.2)$$

It is possible that any of the two users have unknown ($\{0\}$) stance , in which case, their edges are ignored. However, as described in the next step, we also use a label propagation model to propagate stance over the users’ networks. This way, the stance labels of users expand over the network, and more conversations could be labeled.

5.3.4 Step 3: Propagate Node Stance Labels to Other Nodes

This step uses a bi-partied label propagation algorithm to propagate labels to other nodes. Let’s assume σ and σ' are influence functions. For a bipartite network I , the stance propagates in two steps: 1) influence propagates from users to hashtags, and in step 2) influence

spreads from influenced hashtags to users. We assume θ_u to be a parameter that acts as a threshold for spreading the influence from hashtags to users. The label propagation model could be represented as:

$$\tilde{S} \leftarrow \sigma'_{\theta_h}(I' \cdot S^I) \quad (5.3)$$

$$S^I \leftarrow \sigma_{\theta_u}(I \cdot \tilde{S}) \quad (5.4)$$

where \cdot is dot product, and I' is the transpose of the matrix I . As we want the stance to propagate to as many nodes as possible, for influence functions, we reuse the model proposed in the last chapter i.e., Linear threshold model (LTM) with decreasing threshold (LTMDT). In LTMDT the threshold condition linearly decreases after every iteration that allows propagating the stance to nodes that are connected even by relatively small edge weights. LTMDT model is described as:

$$\sigma_{\theta_{uj}} = \begin{cases} 1, & \text{if } \sum_{k=1}^n w_{jk} * \tilde{s}_k > f_t(\theta_u) \\ -1, & \text{if } \sum_{k=1}^n w_{jk} * \tilde{s}_k < -f_t(\theta_u) \\ 0, & \text{otherwise} \end{cases} \quad (5.5)$$

where f_t is a uniformly decreasing function of the iteration number t , and s_k^I is users' stance based on network data.

Though the confidence estimate is not used in bi-partite label propagation, we define it here as it will be needed while describing the joint model (discussed later). Confidence estimate is defined as the ratio of weight of the edges leading to the stance of a user, divided by the sum of all edge-weights for that user:

$$R_j^N = \begin{cases} \frac{\sum_{k=1}^n w_{jk} * \mathbf{1}_{\tilde{s}_k = s_j^I}}{\sum_{k=1}^n w_{jk}}, & \text{if } s_j^I \neq 0 \\ 0, & \text{otherwise} \end{cases} \quad (5.6)$$

where R_j^N is the confidence in the estimating stance s_j of user j .

5.3.5 Step 4: Predict Labels of Conversations using the Text Classifier

The text classifier uses conversations between any two users that are already labeled (last step). All source reply pairs such that $C_{ij}^k = 0$ are used as the training set for a text classifier that takes source and reply pairs as input. This classifier after training is used to predict the stance labels for rest of the conversations i.e., $C_{ij}^k = 0$. The predictions are used to predict labels for unlabeled nodes in the next step.

For a better performance, we again use a confidence threshold function that only allows to use the predictions above a certain threshold to be used as training examples. Though the text classifier only predicts one label for a source-reply text pair (obtained from a conversation), because users can have multiple conversations, multiple predictions (one for each source-reply pair) can be used to quantify the confidence in the stance estimation between two users C_{ij} . In each iteration, the number of 'favoring' text and 'denying' the users u_i, u_j have, determines the stance C_{ij} and prediction confidence R_{ij}^T . This step can be formulated as:

$$C_{ij} = \begin{cases} 1, & \text{if } \frac{\sum_{k=1}^m \mathbf{1}_{C_{ij}^k > 0}}{\sum_{k=1}^m \mathbf{1}} > f_t(\theta^T) \\ -1, & \text{elif } \frac{\sum_{k=1}^m \mathbf{1}_{C_{ij}^k < 0}}{\sum_{k=1}^m \mathbf{1}} > f_t(\theta^T) \\ 0, & \text{otherwise} \end{cases} \quad (5.7)$$

$$R_{ij}^T = \begin{cases} \frac{\sum_{k=1}^m \mathbf{1}_{C_{ij}^k > 0}}{\sum_{k=1}^m \mathbf{1}}, & \text{if } C_{ij}^T > 0 \\ \frac{\sum_{k=1}^m \mathbf{1}_{C_{ij}^k < 0}}{\sum_{k=1}^m \mathbf{1}}, & \text{elif } C_{ij}^T < 0 \\ 0, & \text{otherwise} \end{cases} \quad (5.8)$$

where C_{ij}^k is the stance of the k^{th} source-reply pair of the users i, j who has a total of m conversations. For a better performance, we expand the training set over iterations such that in early iterations more confident examples are only used for expanding the training set. Therefore, we use the conversation with confidence higher than threshold criterion $f_t(\theta^T)$. Note that f_t is regularly decreasing function over iterations t as in the last chapter.

5.3.6 Step 5: Predict Node Labels using Edge Labels

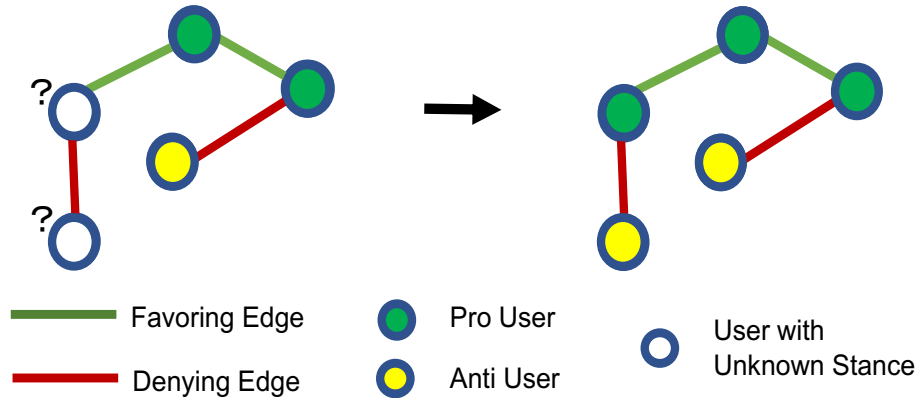


Figure 5.6: Step of predicting stance of nodes (pro/anti users) from stance of edges (as in favoring/denying in conversations)

In this step, given a few unlabeled nodes and a few labeled edges, we predict the labels of unlabeled nodes. This step is the reverse of step 2 and is illustrated in Fig. 5.6. For all the labeled edges, if they are connecting two nodes such that one of the two nodes is unlabeled, the label of the labeled node and the label of the edge, is used to predict the label of the unlabeled node.

Stance of user s_i could be derived from the stance of another user s_j with whom he had conversations. The equation could be written as:

$$s_i = \begin{cases} 0, & \text{if } s_j = 0 \\ -1, & \text{elif } C_{ij} = -1 \text{ and } s_j = 1 \\ -1, & \text{elif } C_{ij} = +1 \text{ and } s_j = -1 \\ +1, & \text{elif } C_{ij} = +1 \text{ and } s_j = 1 \\ +1, & \text{otherwise} \end{cases} \quad (5.9)$$

where C_{ij} is derived using 5.7. This could be thought as the majority label of all k conversations between user i and user j .

5.3.7 Step 6: Label Mixing and Adding Confident Node Labels as New Labeled Data

At the end of an iteration, we use the K percentage top confident labels of both classifiers to be added as new training examples. K is a hyper-parameter and its value is determined using experiments.

5.3.8 Joint Model

For the final stance prediction, we create a joint model that combines the predictions of different models using the confidence scores to create a classifier (as described below). The classifier uses the stance given by the more confident model to assign the stance of a conversation J_{ij}^k . If the network based classifier is more confident, then the similarity and difference in the stance of users i, j is used to predict the joint stance.

$$J_{ij}^k = \begin{cases} 0, & \text{if } C_{ij}^k = 0 \text{ and } s_i = 0 \text{ and } s_j = 0 \\ C_{ij}^k, & \text{elif } R_{ij}^T \geq R_{ij}^N \\ +1, & \text{elif } R_{ij}^T \leq R_{ij}^N \text{ and } s_i = s_j \\ -1, & \text{elif } R_{ij}^T \leq R_{ij}^N \text{ and } s_i \neq s_j \end{cases} \quad (5.10)$$

where R_{ij}^T is the confidence score of predicting the the edge stance by the text classifier, and R_{ij}^N is the confidence in predicting the stance of the edge by the network classifier, s_i and s_j are stance of users i and j and C_{ij}^k is the stance predicted by the text classifier. R_{ij}^N is the mean of confidence in estimating the stance s_i, s_j of users using network label propagation, and is defined as:

$$R_{ij}^N = \frac{R_i^I + R_j^I}{2} \quad (5.11)$$

To summarize the steps, we propose an approach to train a conversation (source text, reply text) based stance classifier, with only weak supervision from two to four labeled hashtags. Note that we still use the labeled examples to show the performance of the models.

5.4 Experiments and Results

5.4.1 Dataset Statistics

For the experiments in this chapter, we use a labeled dataset built in chapter 2. We describe the general statistics of the dataset in Tbl. 5.1, and then summarize the labeled part of the dataset in Tbl. 5.2.

Table 5.1: Dataset summary

Events	Users	Tweets	RTUsers	Endtags	Replies
Student Marches	410785	1039778	391127	17821	48277
Santa Fe Shooting	973506	3043731	910967	45057	83293
Iran Deal	685058	3304519	580180	86653	71808

In this chapter, we are interested in support and denial, so we first group implicit and explicit types labels together, and we regroup the stance labels in two classes ignoring comments and queries.

Class labels

1. Denial: Denial means that the reply tweet outright states that what the target tweets says is false or that the tweeter implicitly believes that what the target tweet says is false.
2. Support: Support means that the reply tweet implies that the tweeter believes that what the target tweet says is true or it means that the quote/tweet outright states that what the target tweets says is true.

Also, for the experiments in this chapter, we only take ‘replies’ as the mode of interaction of interest, ignoring ‘quotes’. This lead to the dataset described in Tbl. 5.2.

Table 5.2: Distribution of labeled replies across different events.

Stance Category	Student Marches	Santa Fe Shooting	Iran Deal
Denial	220	304	198
Support	212	225	202

5.4.2 Expanding the Dataset by Including Users’ Timeline Data

As our stance in conversation prediction model is based on ‘stance of users’, we would like to improve our users’ stance prediction accuracy. In chapter 5, we showed that a text classifier

and a network classifier could be jointly trained to improve stance prediction accuracy of users' in a stance dataset. In this chapter, instead of using the same approach, we use an alternative approach. As the stance based model uses network data in a semi-supervised approach, one way to improve a semi-supervised classifier is by getting more unlabeled data to the network. For example, for our problem on stance learning, one way to add additional data is by collecting more data for each user in the network. This is easy for Twitter users as a Twitter API allows to collect users' timeline data. Timeline data allows to have more retweets as well as more hashtags for each user, thereby, extending the user's data by almost a factor. We summarize this enhanced dataset in Tab 5.3. Note we use this enhanced dataset for training the model in the rest of the chapter.

Table 5.3: Summary of the enhanced dataset that includes users' timeline data

Events	Users	Tweets	RTUsers	Endtags	Replies
Student Marches	578669	9712379	486897	761105	1780261
Santa Fe Shooting	1157176	14673240	1007408	778000	1815277
Iran Deal	870495	18244243	680612	809967	1803792

5.4.3 Seed Hashtags and Seed Users

As in the last chapter, a small fraction of data points (seed users) are needed in the beginning. To get seed user labels, we label a few (two to four) popular hashtags which we call as seed hashtags (details in Tab. 5.4). The labels given by these seed-hashtags are propagated to users using the label propagation algorithm. These seed labeled users are noisy labels (not ground truth) and these labeled users are often only a small fraction of all users.

We describe the hashtags used as seed hashtags to train the models in Tab. 5.4. Using seed hashtag, one can get the seed users by using label propagation model on the user-hashtag networks.

Table 5.4: Seed hashtags and their labels

Dataset	Seed Hashtags
Iran Deal	#thankyoutrump: Pro, #iranuprising: Anti, #freeiran: Anti
Student Marches	#defendthesecond: Pro, #2ashallnotbeinfringed: Pro, #2adefenders: Pro, #2ndamendment: Pro, #guncontrolnow: Anti, #marchforourlives: Anti
Santa Fe Shooting	#defendthesecond: Pro, #2ashallnotbeinfringed: Pro, #2adefenders: Pro, #2ndamendment: Pro, #guncontrolnow: Anti, #marchforourlives: Anti

5.4.4 Baseline Models and Their Performance

We consider a number of classifiers including traditional text features based classifiers and neural-networks based models. In this section, we describe the input features, the model architecture details, the training process and finally, discuss the results.

5.4.5 Input Features

As we have sentence pairs as input, we use features extracted from text to train the models. For each sentence pair, we extract text features from both the source and the response separately.

TF-IDF

Tf-Idf (Term frequency- inverse document frequency) [104] is very popular feature commonly used in many text based classifier. In our research, we use TF-IDF along with Support-Vector Machine (SVM) model that we describe later.

Glove (GLV)

To get word vectors, we used Glove [96] which is one the most commonly used word vectors. Before extracting the Glove word vectors, we perform some basic text cleaning which involves removing any @mentions, any URLs and the Twitter artifact (like ‘RT’) which gets added before a re-tweet. Some tweets, after cleaning did not contain any text (e.g. a tweet that only contains a URL or an @mention). For such tweets, we generate an embedding vector that is an average of all sentence vectors of that type in the dataset. The same text cleaning step was performed before generating features for all embeddings described in the paper.

5.4.6 Classifiers

A text classifier is used in step 4 of the training process. Additionally, it is also used to determine the baseline results. As mentioned earlier, we tried two types of classifiers: 1) TF-IDF Text features based SVM classifier that uses only text in replies, and 2) neural-networks (deep learning) based classifiers that input source-reply text pairs. We describe the details of the classifiers next.

SVM with TF-IDF features

Support Vector Machine (SVM) is a classifier of choice for many text classification tasks. The classifier is fast to train and performs reasonably well on wide-range of tasks. For the Text SVM classification, we only use the reply text to train the model. The classifier takes TF-IDF features as input and predicts the two class stance classes. We would expect that this simple model cannot effectively learn to compare the source and the reply text as is needed for good stance classification. However, we find that this model is still able to get reasonable accuracy on the training set and therefore serves as a good baseline.

Deep Learning models with GLV features

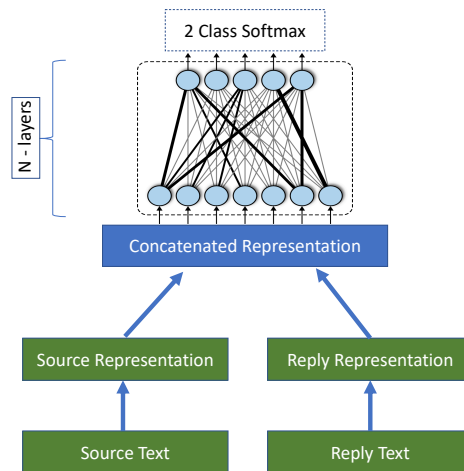


Figure 5.7: Deep learning model diagram which was used in step 4 of the proposed system.

As opposed to SVM text classifiers, a neural-network based models could be designed to use text-reply pair as input. One such model is shown in Fig. 5.7. A neural network based architecture that uses both source and reply can effectively compare target and reply posts and we expect it to result in a better performance. This type of neural network can further be divided in two types based on inputs: 1) Word vectors (or embeddings) are used as input such as Glove (GLV), 2) Sentence vectors (or sentence representations) are used as input such as skip-thoughts, DeepMoji and a joint representation of skip-thought and deep-moji (SKPDMJ). The first model that takes word embeddings as input requires a recurrent layer that embeds the text and reply to a fixed vector representation (one for target and one for reply). One fully connected layer that uses the fixed vector representation input and a softmax layer on top to predict the final stance label. The second type of model that uses the text and reply representations only have one (or more) fully connected layer and a softmax layer on top to predict the final stance label. In chapter 2, we found that word vectors based models are as good as sentence representation based models. As sentence representation based models require an additional step of getting sentence vectors, for this experiment, we stick with word-vectors based neural networks only.

5.4.7 Training Process and Parameters

Our neural-network based models are built using Keras library ². The models used feature Glove word vectors as input. Because Glove is a word vector embeddings, we use an addition recurrent neural network to embed an entire sentence to a fixed size representation. This recurrent layer is an addition recurrent network layer right above the input to create a fixed size sentence embeddings vector. The fully connected layer is composed of relu activation unit followed by a

²<https://keras.io/>

dropout (20 %) and batch normalization. For all models, a final softmax layer is used to predict the output.

The model is trained using ‘RMSProp’ optimizer using a categorical cross-entropy loss function. The number of fully connected layers and the learning rate were used as hyper-parameter. The learning rate we tried were in range 10^{-5} to 10^{-1} . The fully-connected layer size we tried varied from 1 to 3. Once we find the best value for these hyper parameters by initial experiments, they remain unchanged during training and testing the performance of the model for all four events. For all models we find that a single fully connected layer performs better than multi-layered fully connected networks, so we use single layer network for all the results discussed next.

We also have three hyper parameters that are used in the model training steps namely: 1) θ_u , 2) θ^T and 3) K (mixing parameter). The values of these parameters are determined by experimenting on one of the datasets (Student Marches). We find the following values that work well: $\theta_u = 0.7, \theta^T = 0.7, k = 0.2$.

5.4.8 Results and Discussion

We provide the training plots in Fig. 5.8 (with only weak supervision) and Fig. 5.9 in which weak labels and labeled conversations data from other events was used for training. As we can observe, in both cases, performance of the classifiers improve over iterations. This is as expected as more and more users in the network are labeled, the classifiers have more data for training. For both the plots, ‘Iran Deal’ performed on the worse side. This is also not entirely unexpected, as we had observed in the empirical correlation section that the users’ stance and their stance in conversations are less correlated for Iran Deal. If we compare, the two plots, we see that for the network LP classifier performed better when labeled text data was not used in the training. Surprisingly, the text classifiers achieve the same performance with and without additional labeled data.

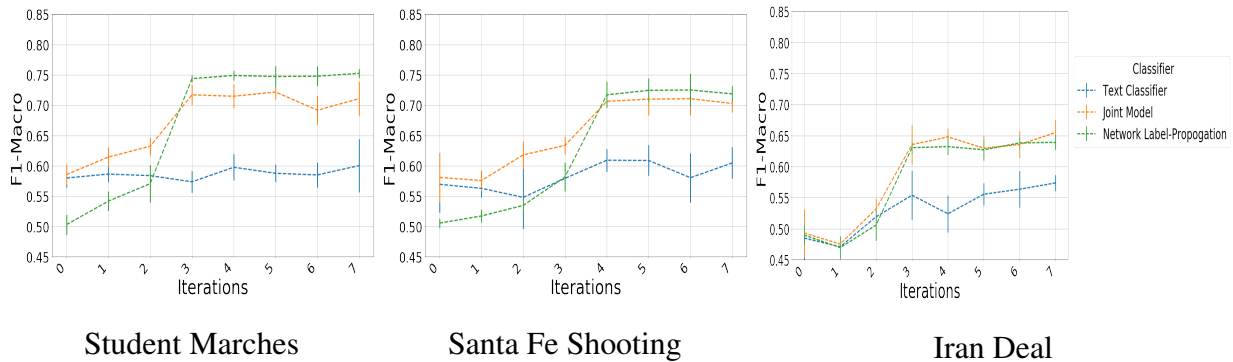


Figure 5.8: Models Performance with training iterations when only labeled data from weak-supervision is used for training.

We summarize the performance of the models in Tab. 5.5 in which we show the f1 score (macro) for all models for each dataset. As we can observe, if we consider the mean values across events, the jointly trained models perform better. However, the text based models, even

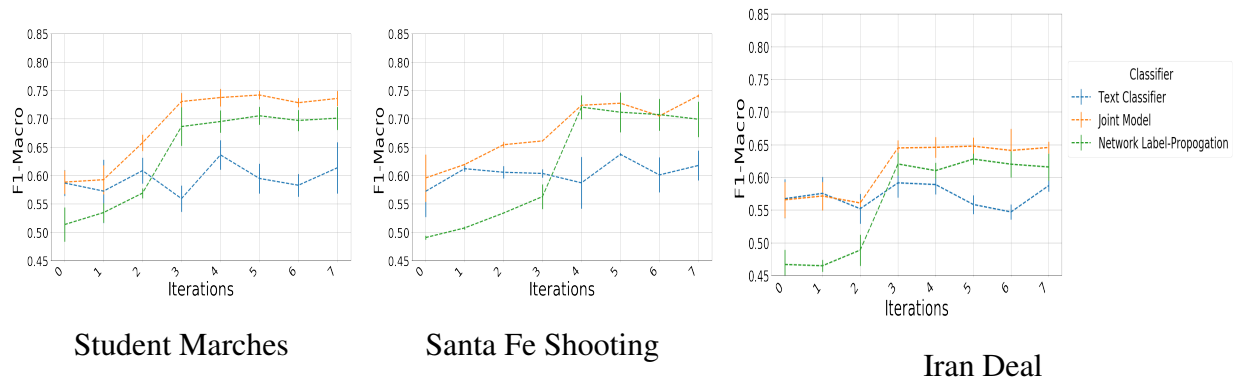


Figure 5.9: Models Performance with training iterations when other events text data is used while training the neural network text classifier.

when trained jointly, have minimal improvement in performing indicating that learning stance in conversations just from text is a challenging task. Surprisingly, the networks based models performs reasonably well which confirms out earlier find that users' stance and stance is conversations are correlated. The best mean performance is achieved when weak supervision is augmented with additional labeled data from other events.

For the final predictions of the joint model, we show the confusion matrix as shown in Fig. 5.10. As we can observe the model is fairly good at predicting both the favoring and denying type conversations. For the event 'Iran Deal' the performance is worse (for denying) but this is also as we expected from our earlier observation on empirical correlation.

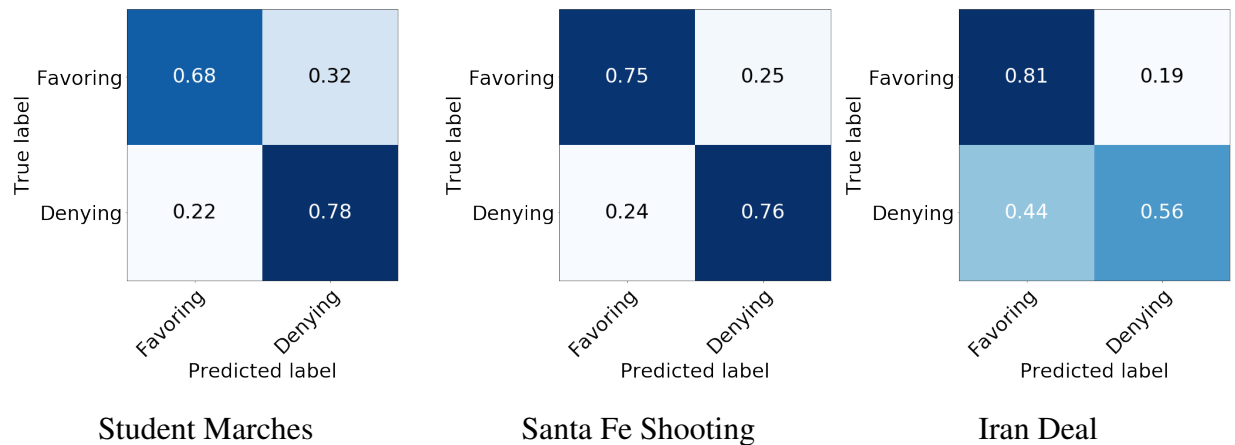


Figure 5.10: Confusion Matrix for stance in conversations (True label) and stance predicted by the joint model. For the performance shown above, no labeled examples were used in the training set.

Table 5.5: Performance of models on different datasets. Bold font indicates the best for a dataset. F1-score Macro is used for comparing the models.

Classifier Type ↓ Dataset →	Student Marches (F1-Macro)	Santa Fe Shooting (F1-Macro)	Iran Deal (F1-Macro)	Mean (F1-Macro)
Baselines				
Random	0.50	0.46	0.52	0.49
Majority	0.40	0.42	0.41	0.41
Leave-one-out Event Based Supervised Text Classifiers				
SVM using Reply Text	0.52	0.56	0.56	0.55
Text based Neural Network using Source-Reply pair	0.62	0.63	0.58	0.61
Weakly Supervised Network Label-Propagation Models				
Hashtags LP	0.51	0.60	0.47	0.53
Retweets LP	0.72	0.71	0.43	0.62
Hashtags and Retweets LP	0.71	0.70	0.55	0.65
Weakly Supervised (Hashtags + Retweets + Text) Jointly-trained Models				
Text Based Neural-Network	0.58	0.56	0.56	0.57
Hashtags and Retweets Label Propagation	0.75	0.73	0.63	0.70
Hashtags, Retweets and Text Joint	0.72	0.73	0.63	0.69
Weakly Supervised (Hashtags + Retweets + Text) Jointly-trained Models with Leave-one-out Text Data				
Text Based Neural-Network	0.62	0.61	0.62	0.62
Hashtags and Retweets Label Propagation	0.72	0.74	0.64	0.70
Hashtags, Retweets and Text Joint	0.73	0.73	0.66	0.71

5.5 Related Work

5.5.1 Stance in Conversations

Stance in conversations was earlier explored as identifying stance in online debates [49, 108, 110, 111]. Though stance-taking by users on social-media, especially on controversial topics, often mimic a debate, social-media posts are very short because of which many of the earlier developed methods don't directly transfer to conversations on platforms like Twitter. More recently, stance in conversations on social media post is gaining popularity [135, 137] with applications to predicting veracity of rumors. In this chapter, we extend the ideas explored in [135] for rumor detection to a more general formulation where stance in conversation are explored on controversial topics. Moreover, we employ ideas from stance of users' (as in pro/con) based on their network position to improve the prediction accuracy of stance the users take in conversations (as in favoring/denying).

5.5.2 Signed Link Prediction

Signed link prediction has remained an active area of research for a while. Leskovec et al. [73] used three data platforms (epinions, slashdot and Wikipedia) to show that the classical theory of structural balance tends common patterns of interaction, and extended the theory to directed graphs. The authors also observed that when two nodes have multiple neighbors in common, then the link is significantly more likely to be positive. In a related, more recent work, Beigi et al. [10] uses the theories of ‘Emotional Information’, ‘Diffusion of Innovations’ and ‘Individual Personality’, for link analysis in signed networks. In most of these work on signed-link prediction, ground-truth for links was known at least partially. In contrast, in this chapter, we try to predict the links without any labeled data while training the models.

5.5.3 Joint Models for Text and Network Data

Graph representation (also called embedding) involves learning low dimensional vector encoding of nodes that maps local network structural characteristics to a continuous space representation. Because these representations are useful in many tasks including node classification and link prediction, learning representation of nodes in networks is an active area of research. Though many algorithms have been proposed to learn representation of nodes on simple graphs [45, 97, 115], most research have only explored explicit unsigned edges. Very few work have explored node representation for signed networks [73], e.g. Kim et al. [61] used sign and direction information in edges to learn node representations that encodes structural information. However, even in these work signs of edges that are explicit are known a priori. In contrast, in our research on Twitter users, we do not have the direct edges between users. Edges like following relationships only provide partial information as often it’s the interaction (e.g. liking, quoting, commenting and retweeting) that convey similarity in users’ thoughts. Thus, a crucial aspect of our work is the way we extract users-networks from Twitter data. Moreover, it is perhaps the first work, where we extract and used both positive and negative edges based on users’ conversations. These signed edges are then used with a network based model to better predict the edges polarity.

5.6 Conclusion and Future Work

In this research, we studied the correlation in ‘stance of users’ (as in pro/con) and ‘stance in conversations’ (as in favoring/denying) when users communicate by tweeting and replying to tweets. Our empirical study using a labeled dataset showed that two users are more likely to have a favoring conversation when they share the same stance on the topic of conversation. Conversely, two users are more likely to have denying stance in their conversations if they have opposing stance on the topic. We used this empirical evidence to propose a weakly supervised method to label stance in conversations. A joint model that uses input from both conversation (source-reply text pair) based classifier and users’ stance based classifier outperforms both the models. By evaluating the models on three human labeled dataset that we created in chapter 2, we estimate that the improvement is over 18% on average. This is a significant improvement and shows the potential of our proposed approach.

We also note that the performance of the co-trained text-classifier (0.57 mean F1-macro) is substantially lower than the supervised text classifier which uses leave-one-out event data (0.61 mean F1-macro). This indicates the text classifier is not gaining much from the weak supervision. This is not entirely surprising as using the stance signal of users to predict the stance in conversations, which in then use to train the conversation models add noise at all steps. Our co-training process is not very effective in reducing the noise as it was for users' stance prediction as shown in the last chapter. Therefore, in future, we would like to try other methods that could perhaps keep the noise to the minimal. The other direction that we would like to pursue is more on the applied side of this research e.g., could we use stance in conversations to find rumors. This is something that is the focus of the next chapter.

Chapter 6

Models of Tree Structured Conversations for Predicting Stance and Rumor Veracity

6.1 Introduction

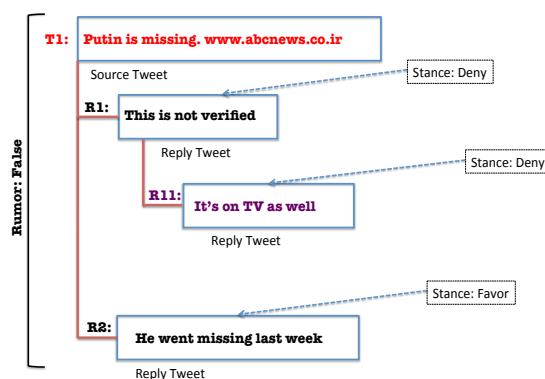


Figure 6.1: Twitter threads with stance and rumor-veracity labels. The conversation tree shown above has two branches a) T1–R1–R11 and b) T1–R2. R1 and R2 are 1st level reply tweets and R11 is a 2nd level reply tweet. Stance labels for each reply is relative to the tweet it is replied to i.e. stance for R11 is with-respect-to R1. There is a rumor-veracity label on the root tweet (T1 in the example above). The goal of this research is to learn the root tweet’s veracity based on pattern in replies.

Online misinformation, commonly called ‘fake news’, has become a serious problem in society [36] to the extent that they are impacting election decisions [2]. Many machine-learning approaches have been proposed to identify and contain the fake-news shared on online social-media platforms [57, 102, 103, 106, 113, 121, 122]. One approach that combines machine-learning and human-intelligence by exploiting stance in reply posts has gained significant attention recently

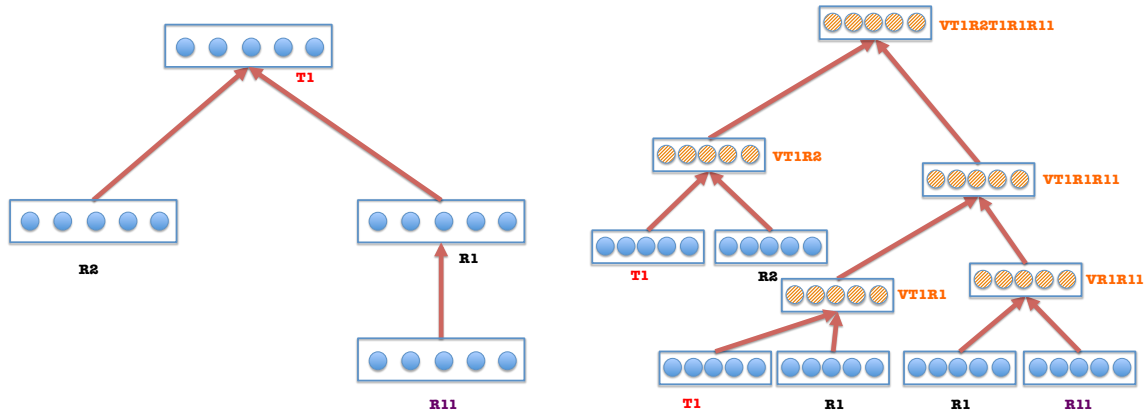


Figure 6.2: Normal tree structure (left) and the modified binarized constituency tree (BCTree) structure for the conversation shown in Fig. 6.1. On left, a tree with structure representing the original thread in which a node can have any number of children. On right, a binary tree structure where source post and reply posts are all leaf nodes such that each reply is placed next to the tweet it was made against and connected to a virtual parent node. E.g. R11 was made against R1 so are connected to VR1R11.

[135, 137]. In this approach, we first identify the stance – categorized as ‘supporting’, ‘denying’, ‘commenting’ and ‘querying’ – in the replies to the original post and then use the stance signal to find rumor veracity i.e. if a rumor is true or false. Prior work has confirmed that replies to a ‘false’ (misleading) rumor contain specific patterns, e.g. more replies deny the claim made in the source post [138]. This approach is promising as people are reasonably good at pointing out misinformation [6] and if such posts could be automatically found, the post could go through enhanced scrutiny before it gets circulated widely.

In this research, we extend this line of work on rumor-veracity and stance learning by proposing a new way to represent conversation trees and new LSTM cells that could be used to detect rumors more effectively. In past, researchers have explored various models to learn from tree structured data [42, 124]. For rumor veracity classification, prior research have found that the approach that performs the best on social-media conversations is a sequence model (like the Long Short Term Memory (LSTM) [53] as discussed in [136]). Sequential classifiers like LSTMs are good at learning temporal structure and are biased to use prior inputs to predict outputs [32]. However, when it comes to comparison tasks like stance classification in threaded discussions, each reply is made against a post or a response to a source post (see Fig. 1). So, we ask, is the regular sequential model apt to learn the relationship between a source post and its replies in conversations? Would a model that can learn the contrast between a source and the reply tweets be more appropriate for rumor classification? To this end, we propose a new tree structure that is obtained from social-media conversation trees but allows for easy comparison of the source and its replies. Additionally, we use a convolution unit to learn patterns in local features for stance classification, and the tree model propagates the signal up the tree for the rumor classification at

the root of the tree. Code to reproduce the experiments are available on Github ¹.

To evaluate our models, we use a human-labeled Twitter dataset that contains stance labels and rumor labels for around two thousand rumour threads related to five different events. Our proposed models achieve the state-of-the-art performance, outperforming the current best model by 12% and 15% on F1-macro for rumor classification and stance classification tasks respectively.

6.2 Models of Tree Structured Social Media Conversations

Tai et al. [114] proposed a tree structured LSTM networks and showed its utility on two tasks of semantic relatedness and sentiment classification. In their work, the tree LSTM is composed of sentence sub-phrases using a given syntactic structure. The benefits of using a recursive tree approach was discussed by Li et al. [75] where the authors concluded that tree models are more suitable for root level identification. Social-media conversations are naturally structured as trees. Can Tree LSTMs be used for classifying node labels in such conversations trees? In this work, we try to answer this question by modeling conversations as trees where each node in the tree is a sentence representation (Fig. 6.2). Node labels in tree structured conversations can be learned using: a) branches of the tree as input to an LSTM (Branch LSTM Model) as used in many prior research e.g. [135, 136] b) using the entire tree as the input (Tree LSTM Model) c) modifying the structure of the tree to better capture the inherent correlations in conversations for a given task (Binarized Constituency Tree LSTM Model). We discuss these formulations next.

6.2.1 Branch LSTM Model

In branch LSTM, the encodings of source-tweet text and the replies text along a tree branch are used as the input and the stance-labels are used as the output (as illustrated in Fig. 6.3). Using a simple text encoder (like mean of a word vectors), at each step, the LSTM gets a sentence embedding and predicts a label. The process is repeated for all nodes in the thread. For example, if we take the thread (T1-R1-R11) (see an example thread in Fig. 6.1), the LSTM takes the R11 as the input in the first time step, R1 as the input in the second time step and T1 as the input in the third time step.

Modelling tree conversations as branches of the tree has two limitations: a) repetition of input as many branches share nodes (e.g. root node is present in all branches) b) no communication between branches during the learning process. The LSTM uses branches independently. Thus, there is no communication between branches during training and inference. We expect that not all branches are useful to predict the veracity of a rumor post and a few branches might have stronger signal. The branch LSTM weighs all branches equally and therefore, is likely to under perform when there are many uninformative branches in a tree. This problem is solved in Tree LSTM.

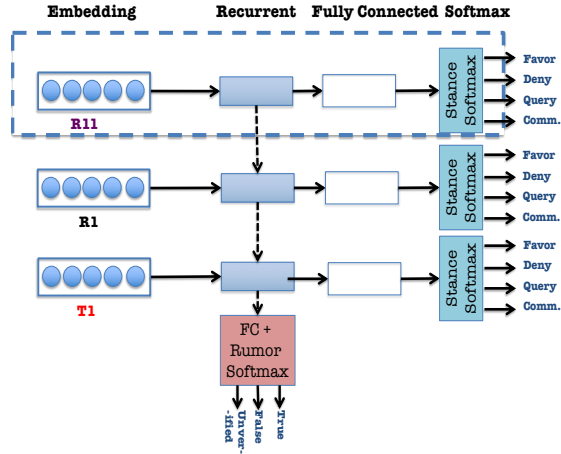


Figure 6.3: Branch LSTM: Recurrent Neural Network (RNN) architecture for sequence labeling. T1, R1 and R11 are embeddings. At each time step, the LSTM uses a sentence embedding vector as input to output a stance label. At the root node T1, the RNN outputs a rumor-veracity label.

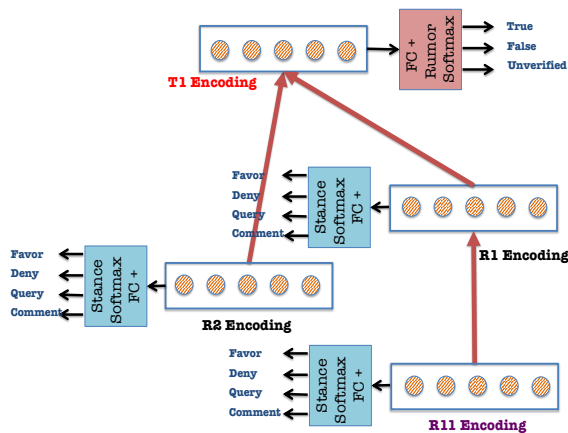


Figure 6.4: Tree LSTM model: Latent vectors at all nodes (except the root node) are used to predict stance label and the latent vector at the root node is used to predict the rumor-veracity label of the conversation.

6.2.2 Tree LSTM Model

A typical social-media conversations consists of a post (source post), its reply and reply to the replies. This is a tree structure with the source post as the root node and the replies as the child nodes. Models for such tree structures was explored in [114] where authors suggested a modification of the LSTM cell to accommodate an unknown number of inputs at a node. For a general tree with any number of child nodes, they suggested ‘Child Sum Unit’ that sums the hidden vectors of child nodes (as in eqn. 6.8). We generalize this formulation to accommodate

¹https://github.com/CASOS-IDEaS-CMU/Detecting_Rumors_In_Conversation_Trees

other operations as shown in Fig. 6.4.

$$\tilde{h} = O_{k \in C(j)} h_k \quad (6.1)$$

where $C(j)$ denotes the set of children of node j and O_k is an operator that acts on the hidden vector h_k of child k to output \tilde{h} . Using this, we define the LSTM transition equations as follows:

$$i_j = \sigma \left(W^{(i)} x_j + U^i \tilde{h}_j + b^{(i)} \right) \quad (6.2)$$

$$f_{jk} = \sigma \left(W^{(f)} x_j + U^{(f)} h_k + b^{(f)} \right) \quad (6.3)$$

$$o_j = \sigma \left(W^{(o)} x_j + U^o \tilde{h}_j + b^{(o)} \right) \quad (6.4)$$

$$u_j = \tanh \left(W^{(u)} x_j + U^{(u)} \tilde{h}_j + b^{(u)} \right) \quad (6.5)$$

$$c_j = i_j \odot u_j + \sum_{k \in C(j)} f_{jk} \odot c_k \quad (6.6)$$

$$h_j = o_j \odot \tanh(c_j) \quad (6.7)$$

Except wherever specified, the notations used are of standard Tree LSTM with input sentence vector x_j , input and output gates i_j and o_j , a memory cell c_j , hidden state h_j and σ denotes the logistic sigmoid function and \odot denotes elementwise multiplication. W and U are weight matrices and bias vector b are parameters which are learned during training.

Child Sum Tree Unit

The child-sum unit involves using sum of all h_k vectors which means $O = \sum$. Therefore

$$\tilde{h} = \sum_{k \in C(j)} h_k \quad (6.8)$$

Child Max-Pooling Unit

The child max-pooling unit involves using the maximum of all h_k vectors across a dimension. Therefore

$$\tilde{h} = \max_P \sum_{k \in C(j)} h_k \quad (6.9)$$

Child Convolve + MaxPooling Tree Unit

Child convolve uses convolution operation of the set of child hidden vectors i.e. $O = \otimes$ where \otimes denotes vector convolution operation. As a normal tree node can have any number of child nodes, convolution operation using all child nodes requires a max-pooling operation to preserve the dimension of \tilde{h} .

$$\tilde{h} = \max_P \otimes_{k \in C(j)} h_k \quad (6.10)$$

where \otimes denotes vector convolution operation and \max_P denotes max pooling operation. A 2d convolution over h matrix results in another matrix and the max pooling operator maps the matrix to vector containing the maximum value of each column in the matrix.

A neural-network model (like an LSTM) expects a pre-defined size of input. Using an operation that reduces the children hidden layer matrix \tilde{h} to fixed dimension vector like in equation 6.8 or in equation 6.10 attempts to solve the problem. However, these reduction operators have limitations e.g. ‘sum’ weighs all children equally and ‘convolve+maxpool’ only picks the convoluted features with maximum value. Ideally this importance factor should be learned from data itself, which is what we intend to achieve using Binarized Constituency Tree (BCTree) LSTM Model.

6.2.3 Binarized Constituency Tree (BCTree) LSTM Model

Social media conversations are in the format of a tree where a node can have many children. Converting this tree structure to another tree structure in which each node always contain two children creates a consistent format which is convenient for matrix operations needed to train neural networks. Additionally, for tasks like stance learning, where its important to compare a reply against its source post, a source reply-pair should be placed such that the contrast features can be effectively learned. To achieve this, we modify the original structure to a binary tree which we call Binarized Constituency Tree (BCTree).

In BCTree, all source posts and their replies appear as leaf nodes (Fig. 6.5). A reply is always paired with its source (this requires source node to be duplicated) and they are connected to a new (virtual) parent node. To construct a BCTree from a tree, we replace all parent node with a new virtual node. The original parent node and a child node are then connected to the new virtual parent node. If a parent node has more than one child, additional virtual nodes are created to keep the tree binary.

Because each node in a BCTree always has only two children, and therefore is consistent, many operators are trivially supported. E.g. we can use hidden vector concatenation. Similarly, for convolution, a convolution unit with kernel size 2 and stride size 1 (comparing a source post and a reply) preserves the dimension of h_k (as BCTree node always have 2 children). Thus additional operation like ‘Sum’ or ‘MaxPooling’ is not needed.

Child Sum BCTree Unit

This uses the same operation as in the normal tree structure (see equation 6.8).

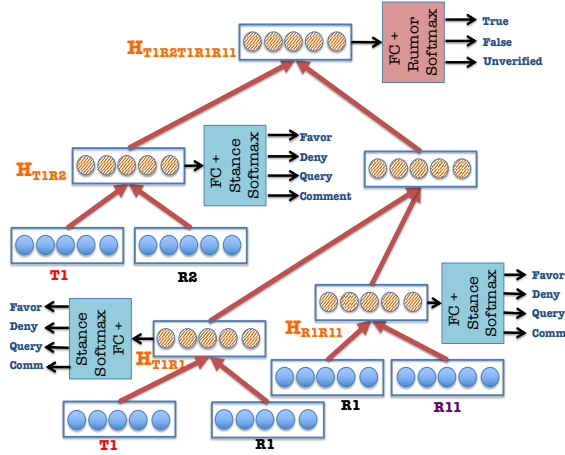


Figure 6.5: BCTree LSTM model: Latent vectors at virtual parent node of each leaf node is used to predict stance labels (e.g. H_{R1R11} to predict stance of $R11$) and the latent vector at the root node is used to predict the rumor-verity label of the conversation.

Child Concat BCTree Unit

$$\tilde{h} = \oplus_{k \in C(j)} h_k \quad (6.11)$$

where \oplus denotes vector concatenation operation.

Child Convolve BCTree Unit

$$\tilde{h} = \otimes_{k \in C(j)} h_k \quad (6.12)$$

where \otimes denotes vector convolution operation.

Combinations of BCTree Units

Because a BCTree has a uniform structure, any combination of the previous discussed units can also be combined together. Some possible combinations we try are 'Convolve + Concat', 'Convolve + Sum' and 'Convolve + Concat + Sum'.

6.3 Experiments and Results

6.3.1 Datasets

We use PHEME 5 events dataset. This dataset was created as a part of the PHEME project² which aims to find and verify rumors shared on social-media platforms [137, 138]. The dataset consists of Twitter conversation threads on five different events and contains three types of annotations. Each thread is labeled as either rumor or non-rumor. Rumors are annotated for their veracity as

²<https://www.pHEME.eu/>

‘true’, ‘false’ or ‘unverified’ (see Tab. 6.1). For a subset of the true rumors, we also have stance labels for each reply in the threaded conversations. The stance labels are ‘support’, ‘deny’, ‘comment’ and ‘query’ (see Tab. 6.2). As we can observe in Tab. 6.2, this dataset is highly skewed towards ‘comment’.

Table 6.1: Conversation threads in the PHEME dataset

Events	True	False	Unverified
Charlie Hebdo (CH)	193	116	149
Sydney siege (SS)	382	86	54
Ferguson (FG)	10	8	266
Ottawa shooting (OS)	329	72	69
Germanwings-crash (GC)	94	111	33
Total	1008	393	571

Table 6.2: Stance labels for Tweets in the conversations. Event codes are described in Tab. 6.1

Events	Support	Deny	Query	Comment
CH	239	58	53	721
SS	220	89	98	700
FG	176	91	99	718
OS	161	76	63	477
GC	69	11	28	173
Total	865	325	341	2789

6.3.2 Feature Representation

We use four different models that have shown good results on various NLP tasks to extract text features.

Mean of Glove word vectors

To get word vectors, we used Glove [96] and the mean of these word vectors are used as the sentence embedding. Before extracting the Glove word vectors, we perform some basic text cleaning which involves removing any @mentions, any URLs and the Twitter artifact (like ‘RT’)

which gets added before a re-tweet. Some tweets, after cleaning did not contain any text (e.g. a tweet that only contains a URL or an @mention). For such tweets, we generate an embedding vector containing uniformly generated numbers between -0.5 and 0.5. The same text cleaning was performed before generating features for all embeddings described in the rest of the paper.

BERT embeddings

BERT ³ is not a ready to use model to generate embeddings in its original form. It is rather a model that can be tuned for a task [29]. We first tried to tune the model on our rumor classification task. But since the rumor classification dataset is relatively small, while evaluating we found that tuning did not lead to a good performance. We then considered other datasets that can be used for tuning. Because natural language entailment task (which predicts entailment, contradiction, or neutral between two sentences) is similar to stance learning, we use the BERT model and tune it on Multi-Genre Natural Language Inference task [128]. The tuned model is then used to generate BERT embedding which is the vector representation on the last layer of the Bert model. This tuned BERT model generates a 768 dimension vector for each sentence.

Skipthought (SKP) embeddings

We use the pre-trained model shared by the authors of Skipthought [63] ⁴. The model uses a neural-network that takes sentences as input and generate a 4800 dimension embedding for each sentence. Thus, on our dataset, for each post in Twitter conversations, we get a 4800 dimension vector.

DeepMoji (EMT) embeddings

We use the DeepMoji [35] pre-trained model ⁵ to generate deepmoji vectors. Like skipthought, DeepMoji is a neural network model that takes sentences as input and outputs a 64 dimension feature vectors.

Skipthought and DeepMoji joint (SKPEMT) embeddings

Because DeepMoji and Skipthoughts are different types of encodings, we also tried a concatenated version of them which we call SKPEMT. This encoding is of size 4864 dimension.

6.3.3 Models Training

Following the convention in prior work [136], we use event wise cross-validation, which means out of five events, four events are used to train a model and one event is used to validate the performance.

³<https://github.com/huggingface/pytorch-pretrained-BERT>

⁴<https://github.com/ryankiros/skip-thoughts>

⁵<https://github.com/huggingface/torchMoji>

Table 6.3: Stance learning results: F1-score (macro) and mean of F1-macro (Mean-F1) for different events.

Model↓ Event →	CH	SS	FG	OS	GC	Mean F1
Majority	0.189	0.190	0.197	0.192	0.175	0.188
Branch LSTM Models						
GLOVE	0.332	0.322	0.298	0.305	0.385	0.329
BERT	0.384	0.393	0.332	0.380	0.425	0.383
SKP	0.424	0.417	0.373	0.454	0.455	0.425
EMT	0.370	0.332	0.365	0.399	0.442	0.381
SKPEMT	0.428	0.424	0.397	0.463	0.468	0.436
Tree LSTM Models - ‘Child Sum’ Cell Type						
BERT	0.512	0.580	0.528	0.481	0.522	0.524
SKP	0.490	0.565	0.540	0.495	0.568	0.532
EMT	0.443	0.514	0.444	0.453	0.509	0.473
SKPEMT	0.509	0.577	0.524	0.504	0.529	0.529
Tree LSTM Models - ‘Child Convolve + MaxPooling’ Cell Type						
BERT	0.510	0.564	0.522	0.476	0.530	0.520
SKP	0.514	0.579	0.553	0.469	0.547	0.532
EMT	0.486	0.478	0.530	0.439	0.496	0.486
SKPEMT	0.480	0.574	0.497	0.477	0.598	0.525
Prior Research						
[136]	0.465	0.446	0.373	0.475	0.543	0.460
[135]	0.427	0.495	0.390	0.457	0.523	0.458
[83]	0.326	0.323	0.260	0.323	NA	NA

We define the overall objective function using cross-entropy loss, as can be seen in equation 6.13, where $i \in n$ samples, j are classes, y is the (one-hot) true label, and p is the probability output for each label. In multi-task training, the total loss is the sum of loss for stance learning task and rumor learning task. As shown in Fig. 6.3, Fig. 6.4 and Fig. 6.5, we use the output of the softmax layer for classifying stance and rumor labels of nodes in trees.

$$L(y, p) = -\frac{1}{n} \sum_{i,j} y_{ij} \log(p_{ij}) \quad (6.13)$$

All operations in our models are fully differentiable, so these models can be trained end-to-end. Because the dataset has unbalanced labels, we can use over sampling of minority classes to create balanced input to train models. For rumor, balancing is easy as each tree has one rumor label, so we over-sample minority labeled trees to balance the training set. For stance labels, balancing is not trivial. The stance classes can be balanced by creating duplicate nodes of minority classes and connecting the new nodes to the original parent nodes. However, this results in changing the structure of trees. Thus we only used balancing on original conversation trees for stance classification and not for rumor classification on BCTrees.

Our LSTM models are built using PyTorch⁶ and DGL library⁷. The Branch LSTM models used feature vectors as input, adds an LSTM layer, a linear dense activation layer followed by a

⁶<https://pytorch.org/>

⁷<https://www.dgl.ai>

dropout (0.3) [112] and uses a softmax layer for the output (rumor or stance). The models are trained using stochastic gradient descent (SGD) optimization using a cross-entropy loss function. The size of LSTM hidden layer and learning rate were used as hyper-parameter. The learning rate we tried were in range .0001 to 0.01. The LSTM layer size we tried varied from 16 to 256. We found 64 to be the best hidden dimension vector size and 0.08 to be a good learning rate for training the branch LSTMs. Once we find the best value for these hyper parameters by initial experiments, they remain unchanged during training and evaluations of the model for all five events.

The training of tree models also followed the same pattern except they use an entire tree conversation. The convolution units use convolution kernels of size 2 (i.e. it used two hidden vectors at time) and stride of 1. We tried learning rate from 0.001 to 0.1, and .008 was found to work the best. We again used stochastic gradient descent (SGD) optimization with a cross-entropy loss function. For multi-task training, we used step wise training that alternates between rumor objective and stance objective. We train the models for 30 epochs.

To evaluate the trained models, we use F1-score which is defined as the harmonic mean of precision and recall. Rather than using accuracy, we use F1-score as the metric for evaluating the performance of the models for two reasons: a) PHEME dataset (the dataset we use) is skewed towards one class ('comment'), hence, a classifier that predicts the majority class can get a good accuracy. F1-score (macro) balances the classes and considers precision as well as recall. 2) Prior work on this dataset used F1-score [136]. Thus, the use of this measure allows to compare with prior research. The performance for a validation event is the F1-macro obtained by evaluating the model trained on all data except the validation event data. This step is performed for all five events, and the mean of F1-macro scores from all five events is used to compare the models. For the stance classification task, the F1-score (macro) is defined in Eqn. 6.14. For the rumor classification task, the F1-score (macro) is defined in Eqn. 6.15.

$$F1_{stance} = \frac{F1_{deny} + F1_{favor} + F1_{query} + F1_{com.}}{4} \quad (6.14)$$

$$F1_{rumor} = \frac{F1_{true} + F1_{false} + F1_{unverified}}{3} \quad (6.15)$$

6.3.4 Stance Classification Results

We present the results of evaluating the models for stance classification in Tab. 6.3. The Tree LSTM model that uses 'Child Convolve + Maxpooling' with skipthought features outperforms all other models (0.532 mean f1). The Tree LSTM model using 'Child sum' unit performs equally well on mean value but was worse on three events.

In Fig. 6.6, we show the confusion matrix for the best performing stance classifier. As we can observe, the model is best at classifying 'Comment' and is worst at classifying 'Denial'. The poor performance of the denial class could be partially attributed to the unbalance of classes ('Deny' being the smallest) in the dataset.

If we compare the stance classification results based on feature types, we see that BERT and SKP are often comparable and EMT is slightly worse than them. SKPEMT performs better than EMT and BERT, but is as not as good as SKP. Because of space limitation, we do not present

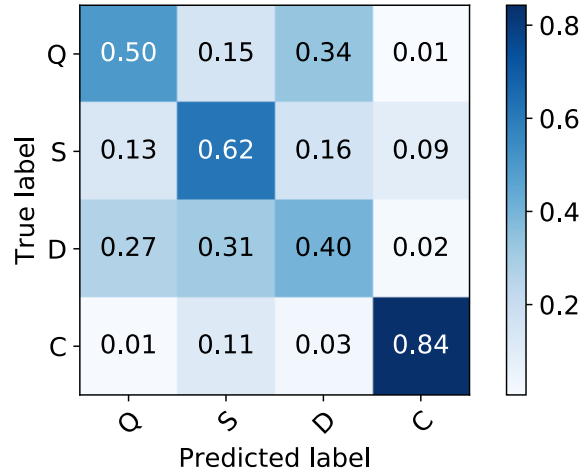


Figure 6.6: Normalized stance confusion matrix. Q, S, D and C labels indicate ‘Query’, ‘Support’, ‘Deny’ and ‘Comments’ respectively.

results for Glove features for Tree based models as, in almost all cases, the mean of Glove vectors as sentence representation performed worse than other features.

For stance learning, the BCTree based models did not work as well as the Tree LSTM based models. This is likely because we are not able to balance stance classes in BCT trees. BCTrees stance nodes can be balanced before binarizing, but that adds many additional new nodes. These new virtual nodes don’t have stance labels and results in poor performance.

6.3.5 Rumor Classification Results

We present the rumor classification results in Table 6.4.

For rumor classification, the best performing model uses ‘Convolve + MaxPool’ as units in Tree LSTM (Mean F1 of 0.379 using SKP features) and is trained in multi-task fashion. Other comparable models are ‘sum’ and ‘Convolve + concat’ units with BCTree LSTM. For SKPEMT features, the best performance was obtained using ‘Maxpool’ cell with a Tree LSTM model. We expected BCTree LSTM to work better than Tree LSTM. They are almost comparable but BCTree LSTM is slightly worse. This is likely because binarizing a tree creates many new nodes (without labels), and as height of trees increase it becomes more difficult for LSTMs to propagate useful information to the top root node for rumor-veracity classification.

If we compare the different types of features, SKP features outperformed others in almost all cases. It should be noted that SKP features are also higher in dimension (4800) in comparison to EMT 64 and BERT 768. If we compare, multi-task vs single-task, in almost all cases, performance improved by training in a multitask fashion.

Overall, for rumor classification, the best model is the LSTM model that uses ‘Convolve + MaxPool’ unit and trained on Tree LSTM using multi-task. This exceeds the best prior work

Table 6.4: Rumor classification results: Mean F1-score from different cell-type and feature-type combinations. For NileTMRG, we used the results presented in [64], Tbl. 3.

CellType ↓ Feature →	SKP	EMT	BERT	SKPEMT
Branch LSTM - Multitask				
	0.358	0.359	0.332	0.347
Tree LSTM - Multitask				
Sum	0.364	0.348	0.341	0.364
MaxPool	0.369	0.352	0.339	0.375
Convolve + MaxPool	0.379	0.365	0.359	0.370
BCTree LSTM - Multitask				
Sum	0.371	0.356	0.338	0.371
Convolve	0.367	0.335	0.337	0.362
Convolve+Sum	0.353	0.353	0.329	0.364
Convolve + Concat	0.370	0.354	0.340	0.364
MaxPool	0.353	0.354	0.326	0.352
Convolve+MaxPool	0.363	0.349	0.333	0.357
Concat + Sum	0.364	0.341	0.324	0.364
Convolve+Sum+Concat	0.366	0.343	0.342	0.354
Baselines and Prior Research				
Zubiaga et al.[64]	0.329			
NileTMRG [33]	0.339			
Majority	0.223			

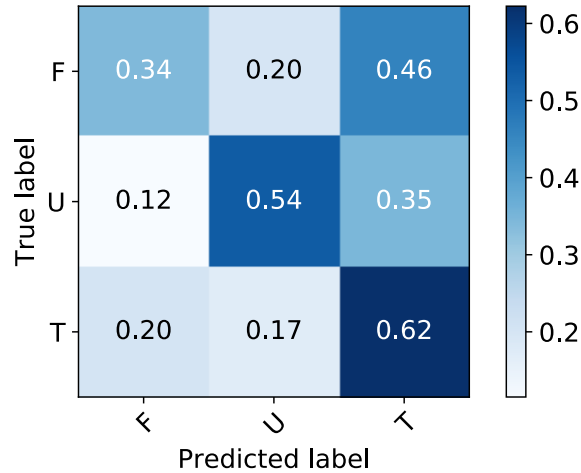


Figure 6.7: Normalized rumor confusion matrix. F, U and T labels indicate ‘False’, ‘Unverified’ and ‘True’ respectively.

by 12% in f1-score. For this model, we show the confusion matrix in Fig. 6.7. As we can observe, ‘True’ (T) and ‘Unknown’ (U) performs equally well and the ‘False’ (F) rumor is the most confusing class. The poor performance of ‘False’ rumors could be linked to the poor performance of ‘Denials’ stance in stance classification. Prior research have shown that a high number of denials is a good indicator of ‘False’ rumors, and therefore a model that is poor at

predicting denials also performs poorly at predicting ‘False’ rumors.

6.4 Related Work

Stance learning and rumor detection lie at the intersection of many different fields. We highlight important related topics here.

6.4.1 Stance Learning

Computational approaches of Stance learning – which involves finding people’s attitude about a topic of interest – have primarily appeared in two flavors. 1) Recognizing stance in debates [95, 110] 2) Conversations on online social-media platforms. Since our research focuses on conversations on social-media platforms, we discuss some important contributions here. Mohammad et al. built a stance dataset using Tweets and organized a SemEval competition in 2016 (Task 6). Many researchers [5, 79, 126] used the dataset and proposed algorithms to learn stance from this text data. In almost the same time frame, work on stance in conversations appeared in the context of fake-news and misinformation identification, we discuss this in the next section.

6.4.2 Rumor and Misinformation Identification

Finding misinformation on social-media platforms has been an active area of research in recent years [27, 50, 82, 107, 121, 134, 136]. Rumor detection that uses stance in the reply posts was initiated by the PHEME project ⁸ and was popularized as a SemEval 2017 task ⁹. The task involved predicting stance (‘supporting’, ‘denying’, ‘commenting’ and ‘querying’) in replies to rumor posts on Twitter and the dataset is described in [137, 138]. A number of researchers used this dataset and proposed many algorithms. For example, [28] proposed an LSTM that uses branches in conversation trees to classify stance in reply posts, and [64] used sequential classifiers for joint stance and rumor classification. More recently [84] suggested two tree structured neural-networks to find rumors i.e. if a post is rumor or not. In this work, we focus on rumor-veracity and stance learning objectives. Our work extends this thread of research by showing that convolution operations that compare source and reply tweets are more effective in learning stance and rumor-veracity.

6.4.3 LSTM and Convolutional Neural Networks

Deep neural networks (DNN) have shown great success in many fields [51]. Researchers have used DNNs for various NLP tasks like POS tagging, named entity recognition [24]. Convolution neural networks [72] are popular in computer vision tasks for quite some time but lately they have shown potential in NLP tasks as well [133]. Yoon Kim [62] used convolution neural networks (CNN) for various NLP tasks. To the best of our knowledge, this is the first work that uses a convolution unit in LSTMs.

⁸<https://www.pHEME.eu/>

⁹<http://www.aclweb.org/anthology/S17-2006>

6.5 Conclusion

In this work, we explored a few variants of LSTM cells for rumor-veracity and stance learning tasks in social-media conversations. We also proposed a new Binarized Constituency Tree structure to model social-media conversations. Using a human labeled dataset with rumor-veracity labels for source posts and stance labels for replies, we evaluated the proposed models and compared their strengths and weaknesses. We find that using convolution unit in LSTMs is useful for both stance and rumor classification. We also experimented with different types of features and find that skipthoughts and BERT are competitive features while skipthoughts have slight advantage for rumor-veracity prediction task.

Chapter 7

Conclusions, Limitation and Future Work

In this chapter, I highlight the main conclusions of this thesis. I also provide the limitations and the potential future work. Besides, I also discuss the scalability and my thoughts on integration of the different methods that I proposed.

7.1 Conclusions

This thesis aimed to resolve the three crucial challenges of stance mining:

1. How do we train stance-learning models on new topics with minimal labeling effort?
2. How can we use multiple interaction modalities for stance mining?
3. How to leverage users' networks for stance prediction?

How do we train stance-learning models on new topics with minimal labeling effort? On social-media discussion, topics change fast, and new issues emerge, making it difficult to reuse prior labeled data. Therefore, there is a need to train stance-learning models on new topics with minimal labeling effort. As we show in this thesis, simple artifacts of social networks like hashtags have noisy signal about the stance of the users who have used those hashtags. The challenge is how to extract the signal from noise. For this, we proposed a semi-supervised learning approach that uses two or more models that are trained in co-training setting, and plenty of unlabeled social-media data to build stance classifiers (chapters 4 and 5). In the proposed approach, using the stance given by a few seed hashtags, multiple stance classifiers are jointly trained which results in reducing the effect of noisy hashtag labels and improving the performance of the classifiers over the training iterations. As the models get trained using only the stance given by a few seed hashtags, the approach is very flexible and could be easily extended to new topics.

How can we use multiple interaction modalities for stance mining? Users' opinions are evident in different types of interactions, e.g., tweeting, retweeting, or liking. Depending on users' preference some users prefer to use tweeting, some others use retweeting and others use liking. Though having multiple interactions allows different ways to learn and predict users' stance, and it is not apparent how we can correctly use the various interaction modalities for stance mining. As discussed in the last paragraph, we proposed a semi-supervised method for jointly training two or more models which are based on different interactions. The proposed co-training method could learn from complementary information in different interactions to train

better classifiers. Though in this work, we only considered three interactions comprised of two networks (namely user-hashtags and user-retweets) and one text classifier (based on users' tweets or user-to-user conversations), the simplicity of our approach allows extending the method to other interactions.

How to leverage users' networks for stance prediction? A person's preference can also be known from his friends' preferences. The current approaches to stance learning ignore important network factors. Since stance is a public act, the actor's social network position should matter. We use the network alignment as one training signal to train the stance classifiers (chapters 3, 4, and 5 uses networks). One of the classifiers used for jointly training the model is a network-based label propagation classifier. The label propagation model effectively utilizes the similarity in the connected nodes to improve the overall stance classification performance. Therefore, the proposed method effectively uses networks to learn users' stance.

Based on our experiments on six controversial topics, we estimate that with 2-4 hashtags as weak labels, the user's stance classifiers could reach an accuracy of over 80%. For learning the stance in conversations, which is a more challenging problem, we achieve accuracy over 70%. More importantly, the proposed methods result in inductive classifiers that could be used with newer data without a need to retrain the models. Furthermore, as demonstrated in chapter 6, the stance learners could be applied to problems that are relevant in society, such as the spread of misinformation.

7.2 Scalability and Integration

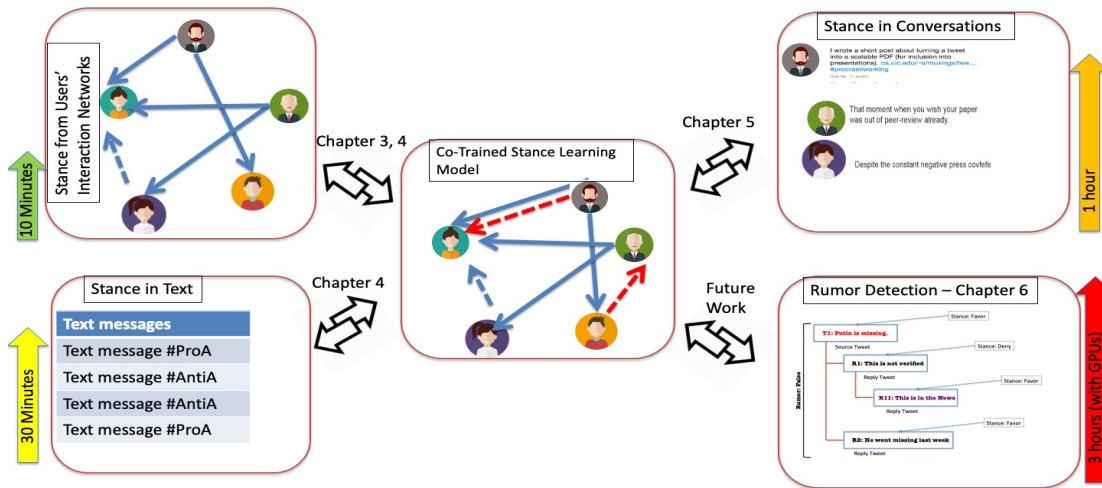


Figure 7.1: Scalability and Integration

Here I discuss the scalability issues and opportunities to integrate different works together. Some of the chapters are related, but not all techniques developed in different chapters are built on top of each other. Ch. 4 and Ch. 5 are related as Chapter 5 extends the ideas developed in chapter 4. Fig. 7.1, visualizes the dependency of different techniques, and as we can observe many of them are loosely coupled. The software/code also reflects this dependency.

Chapter 3 is on users stance prediction, and the output of the model is the same as the output of the model in chapter 4. However, one of the steps in chapter 3 (creating virtual connections based on similarity in node attributes) is very slow as each node needs to be compared with all other nodes to get the similar top nodes to create virtual connections. In my experience, a dataset with 100,000 nodes could take up to a week to get the result based on this model. In contrast, the proposed approach in chapter 4 more salable. For a network in order of 100,000 nodes, we can get the stance of all nodes predicted within an hour. Though theoretically it is possible to combine the ideas in chapter 3 and chapter 4, I would suggest not to do it because of practical reasons.

As mentioned earlier, chapter 4 and chapter 5 are related, and chapter 5 depends on the methods developed in chapter 4.

Ideas from Chapter 5 and chapter 6 can be combined; however, that is not straight forward. Chapter 5 is about stance in conversations, whereas ch. 6 is about using stance in conversations to detect rumors. Though the two chapters appear to be related, they are yet very different. Stance in Ch. 5 is about stance of users, whereas Chapter 6 explores the stance more as supporting or denying claims in posts. Though the difference is subtle, it is important. Supporting a post by expressing a view that the post is true (as in veracity) is different from supporting other persons stance. Because of this, merging the datasets used in the two chapters bring little improvement.

Still, some of the ideas developed in earlier chapters could be taken forward to chapter 6. One such idea is to use the reputation of users to identify false rumors. Reputation of not just the user that posts a message but also of users that engage in conversations. As one can expect, the reputation of users could be related to the extent he shares misinformation, therefore the spread of reputation of users in networks could be something to try in future research. If that works, it could be valuable for identifying potential misinformation posts.

Table 7.1 summarizes the scalability of the proposed approaches, where N is the count on nodes in a network. As we can observe in the table, methods in Ch. 4 are most scalable and methods in Ch. 3 and Ch. 5 are least scalable. For applying the methods in wild, chapter 4 and chapter 5 are most convenient. I have tested these methods on Twitter data and the results on such datasets are promising. Chapter 3 is very slow. As methods in chapter 4 are able to achieve what is being done in chapter 3, there is no reason to use chapter 3 in wild. Chapter 5 on rumor detection could be useful to apply in wild. But there are two complications with this approach. First, it requires conversation trees which is not trivial to collect. Second, the performance of the model, though better than other existing models, is still poor and only slightly above the random choice. This method is promising but requires more investigation before it could be used in wild.

7.3 Limitations

I am listing a few more potential situations where this line of research may not be applicable:

1. When threads are hijacked, which is also call threadjacking
2. When topics in the dataset are not controversial
3. When the dataset is a mixture of many subtopics
4. When the dataset is highly unbalanced

Table 7.1: Time Complexity of Methods

Chapter	Data Type	Time for Training	Scalability
Ch.3 Peopl2vec	Network + Node attributes	Few Days for ~100,000 node	Not scalable, works fine with ~10,000 nodes
Ch.4 Stance Analyzer	Multiple Networks + Node Text	An Hour for ~100,000 nodes	Scalable, can handle hundred thousand nodes
Ch.5 Stance in Conversations	Multiple Networks + Node Text + Conversations	Few Hours for ~100,000 nodes	Scalable, can handle hundred thousand nodes
Ch. 5 Rumor Detection	Conversation Trees	Many Hours for ~100,000 nodes	Not scalable, can handle ten thousand nodes
Ch. 8- Twitter Data Collection	Outputs Twitter Json Data	NA	NA

I explain these points in detail next.

Thread jacking remains a limitation of this work. Though the methods proposed in Ch. 4 are fairly robust to noise in using hashtags, they are not immune to threadjacking. To remove the noise from signal, we: a) use hashtags at the end of tweets, b) use influence functions, and c) use a co-training step that uses only the most robust predictions as labeled examples. However, if a group of actors decides to use hashtags in a way that defies the assumptions of the model, then, the proposed approach will fail. For example, if a group of users who are anti-gun-control, but always use ‘guncontrolnow at the end of all their tweets, our model will not be able to infer the stances of such users correctly, and perhaps will also reduce the accuracy of predicting stance of other users’ in the dataset.

Though our method does not handle the issue of threadjacking, there are many ways to identify threadjacking to remove the threadjacked data. The detailed steps to identify threadjacking is beyond the scope of this thesis. Some ideas that can be explored are: 1) Supervised machine learning-based approaches to identify a threadjacked thread. 2) Annotations by users that the thread has been hijacked as weak supervision.

When topics are not controversial, the network structure in the discussion is not likely to follow a polarized structure (two groups with higher interconnections and lesser cross-connections). In the absence of such a polarized structure, the network classifier (e.g., a label propagation model) is unlikely to add much benefit in users’ stance prediction. Therefore, the co-training method (ch-4) is unlikely to improve the stance classification accuracy as compared to a regular text classifier.

When the dataset is a mixture of many sub-topics, there may be many sub-topics that would

be neutral. Our approach in the current format does not cater to neutral conversations. For example, take the recent Twitter discussion on Coronavirus. The topic covers discussions on ‘Dr. Fauci’, ‘Opening up American economy’, ‘Ban on immigration’ etc. Many users may have stance on one of the sub-topics but could be neutral on others. Therefore, for this type of discussions, I recommend a two-step approach in which first we select a sup-topic and filter the dataset on that sub-topic and use the methods on the sub-topic to understand user’s opinion.

The last situation is when the dataset is highly unbalanced. In an unbalanced discussion, one side of the debate occupies the majority fraction. In such a case, the classifiers cannot be appropriately trained as the unbalanced dataset leads to poor training of classifiers. This problem could be partly solved by balancing the dataset by oversampling the minority class data. However, even this step would fail to improve the classifier if the dataset is extremely unbalanced.

Future work should consider neutral conversations as one of the classes and should extend the methods to work with the neutral class. More potential work has been added to the thesis.

7.4 Future Work

In the future, I would like to suggest ideas for some general questions and would like to consider and evaluate our approach on other types of users’ networks data. For example, we did not consider follower-followee network data which requires additional Twitter API calls. As our proposed approach is very generic, such additional networks could be plugged in easily. Moreover, in the chapter on co-training, we only consider the mixing strategy that combines the top most confident predictions from different models for mixing. However, the top prediction of one classifier need not always be useful to the other classifier. Therefore, it would be worth considering other mixing strategies e.g., one that balances the information gain and the sharing of confident examples. Furthermore, there are other problems where stance learning tools can be applied. For examples, pictures and memes shared on social media platforms while discussing controversial topics also convey stance. Also, in the context of community polarization, stance could be used to understand the sides taken by individual users, and therefore, could lead to a better estimate of the extent of polarization. A phenomenon related to polarization is the formation of echo-chambers. Next, I provide more details on research in these areas along with thoughts on future work that connects this work on stance mining.

7.4.1 What Would it Take to Use Data that Does not Have Hashtags?

Hashtags are useful entities as they are commonly used to highlight the topic of discussion and intent of the author. Some hashtags give clear stance information, and we could use them as a weak signal to extract data that could be used to train machine learning models for stance mining. This is what is proposed in the thesis.

However, hashtags are not the only entities that are suitable for stance learning problems. The stance of users could also be evident from the domain names of the websites that they share, or the user-mentions that they use. So mentions and domain names are other good candidates for feeding the model the stance signal.

It is also possible to use words or phrases (e.g., n-grams) as the input signal. For example, on the topic of gun control, phrases like ‘gun control now or ‘guns should be banned could be used to get the initial seed users. Though promising, there are two problems with this approach: 1) Such phrases or n-grams require the entire text to be pre-processed which could be a challenge with big data 2) we also need to create user-phrases network, and creating such a network would also require processing large data. Therefore, though the use of text phrases is possible, the complexity of identifying and extracting such phrases would require careful consideration.

7.4.2 Big Topics Might be Influencing Smaller topics. How to Handle This?

Larger topics impact sub-topics. For example, a user favoring Trump is likely to favor discussions around Trumps policies, e.g., say ‘Immigration Ban’. This phenomenon could be put to the advantage if the sub-topic has a smaller amount of data. As our model requires more data for semi-supervised learning, in the absence of data on a not so popular topic, one we can collect data on the bigger topic. The result obtained from the bigger issue can then be extrapolated to the smaller topic.

In the case of diversity of conversation, especially if the conversation is not controversial, the recommended step is to filter the entire conversation with the targeted discussion. For example, consider the discussion on the Coronavirus on Twitter. The large topic has many aspects, including ‘how to stop the spread’, ‘effective medications’, ‘performance of Dr. Fauci’ etc. In this diverse conversation, there are a few controversial topics, e.g., ‘performance of Dr. Fauci’ is controversial with a group wanting Dr. Fauci to be fired from his official position. Therefore, a good approach would be first to find the smaller controversial topic and filter the data related to those topics from the larger discussion. Besides, resulting in a more accurate model, this would also speed up the model training process.

The proposed techniques in the thesis are best suited for controversial topics with bi-polar discussions. In case a debate has more than two sides, the proposed co-training method will need to be extended to work with such datasets. This should not be very challenging as both classifiers used in the approach allows multi-class labeling. The challenge could be in how the labels are mixed in the label mixing step and need to be carefully explored in future research.

7.4.3 Extending the Stance Learning Models to Use Pictures and Memes Posted on Social Media

There is limited work on analyzing pictures shared on social-networks for stance mining. However, there is strong evidence that pictures are increasing being used to convey stance [12]. A good example of stance in pictures is the use of memes. While memes can exist as words, emoticons, videos, or gifs, a common form is an image with superimposed text that conveys some message. Various political actors are increasingly using memes to communicate political messaging and memes are increasingly being used in public discourse as elaborated in the New York Times article “The mainstreaming of political memes online” [16]. Though unexplored, pictures and memes are going to be important for the holistic understanding of stance in future. In [12],

we show that a multi-model deep learning model could be trained to extract memes from pictures shared on Twitter. These memes could then be analyzed to find pro-party memes. We use image similarity, meme specific optical character recognition, and face detection to find and study families of memes shared on Twitter in the 2018 US Mid-term elections.

This kind of work could very well be extended to find memes that are pro or anti a given topic. The signal needed to make the stance distinction could come from the methods proposed in this thesis. Weak supervision could come through social-media artifacts like by hashtags or URL domains. Though noisy, these artifacts contain useful stance signal. As we demonstrated in this thesis, it could be better to first extract the set of users that are frequently using these hashtags and group them as pro and con. This labeled group of users could then be used to further spread the stance signal to other users through the co-training approach that we experimented with in this thesis. Based on how the meme is being used by the users in different set, we could give a probability estimate of the stance conveyed by the meme itself.

7.4.4 Analyzing Polarized Communities in Social-Networks via Stance Mining

Community polarization and partisanship are both heavily studied in social-science. Recently, with the presence of abundant data on social-media platforms, computer science researchers are exploring ways to utilize data to better understand such social interactions [37, 39, 43]. Akoglu et al. [1] used signed bipartite networks that represents opinions of individuals to understand political polarity of individuals. Lahoti et al. [70] used constrained non-negative matrix factorization to explore liberal-conservative ideology space on Twitter. Using twitter dataset of controversial topics, the authors were able to separate users by ideology with over 90% accuracy. Unlike these work, I think there is still more value in combining community structure and users' stance to quantify the extent of polarization. Typically, the network used in the quantifying polarization involves extracting the largest connected component from the data [39]. Also, this largest component is often composed one of the interaction modality, typically the retweets network. The main issue with such a network is that there are many users who exist in the network but do not engage in that particular activity e.g, retweeting in this case. The similar phenomenon of not engaging in particular activity could lead to users not present in the largest connected component. From our empirical analysis, we find that the users in this largest component is a small fraction (< 50%) of total users. Quantifying the extent of polarization based on the largest component obtained from one of the modalities could at best be a crude approximation of the real polarization. With our work on stance mining, we are better able to evaluate the stance of each user even those who are not involve in certain activities. Therefore, our approach to stance mining when combined with metrics for quantifying polarization, could better estimate the presence of polarization in social media data.

7.4.5 Stance Mining for Discovering Echo-chambers on Social Media

A concept very closely related to polarized communities is of the echo-chambers. The Oxford dictionary defines an echo-chamber as "An environment in which a person encounters only be-

liefs or opinions that coincide with their own, so that their existing views are reinforced and alternative ideas are not considered”¹. Gilbert et al. [41] used blogs to find echo-chambers. By manually annotating comments on blog posts, they found that comments that agree to the blog-posts outnumber the comments that disagree on most blogs. The authors defined echo-chamber as a blog on which more than 64% of the opinionated commenters agree with the blogger, and they find that majority of blogs are echo chambers. The paper also provides a good perspective of prior research focusing on understanding homophily on social platforms.

Margetts et al. [86] used ‘echo chambers’, in which people are surrounded by like-minded people and opinions that reinforce their own belief systems (in the same way that acoustic echo chambers use hollow enclosures to produce reverberated sounds). It is argued that inhabitants of these ideologically narrow environments are vulnerable to distorted versions of events or fake news, which bounce around the chamber and become regarded as the truth. In this way, echo chambers lead to polarization, dragging those in the middle ground towards more extreme opinions. Echo-chambers are also close knit communities that behave in a particular way depending on the type of information shared. For example, the anti-climate change echo-chamber becomes very active when an anti-climate change article is circulated. In contrast, an article that is pro-climate change will hardly percolate through the same network. If the article diffuses, it mostly gathers negative comments. This is because such communities become highly polarized for certain types of information, thus changing their structure (turtling). The same community might behave like a normal community when an article related to other topic is shared. Thus echo-chambers illustrate the dynamic and complex nature of communities present on social media platforms.

Though there may be variations in the definition of the echo-chamber, it is accepted is that an echo-chambers stop the flow of ideas, especially the ideas that are against the beliefs of the echo-chambers. Though echo-chambers are easy to think of, they are difficult to discover. Learning the stance of users in a network and getting the stance in the articles shared through the networks along with the stance in the conversations around these articles could be a potential way to find and understand echo-chambers. Much of the work in this thesis revolved around finding stance of users and stance in conversations. However, given the stance on users, and using our proposed approach of filtering articles through the user-article sharing matrix, it would be possible to get the stance in the articles. Moreover, as we demonstrate in chapter 4, our models are better able to capture the polarized nature of the users’ networks. Thus this thesis contains the necessary ingredients needed for analyzing echo-chambers. A thorough study of topics that can have echo-chambers would be a natural extension.

¹https://en.oxforddictionaries.com/definition/echo_chamber

0.9

Bibliography

- [1] Akoglu, L. (2014). Quantifying political polarity based on bipartite opinion networks. In *ICWSM*.
- [2] Allcott, H. and Gentzkow, M. (2017). Social media and fake news in the 2016 election. *Journal of Economic Perspectives*, 31(2) 211-36.
- [3] Auer, P. and Ortner, R. (2010). Ucb revisited: Improved regret bounds for the stochastic multi-armed bandit problem. *Periodica Mathematica Hungarica*, 61(1-2):55–65.
- [4] Augenstein, I., Rocktäschel, T., Vlachos, A., and Bontcheva, K. (2016a). Stance detection with bidirectional conditional encoding. *arXiv preprint arXiv:1606.05464*.
- [5] Augenstein, I., Vlachos, A., and Bontcheva, K. (2016b). Usfd at semeval-2016 task 6: Any-target stance detection on twitter with autoencoders. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 389–393.
- [6] Babcock, M., Villa-Cox, R., and Kumar, S. (2019). Diffusion of pro- and anti-false information tweets: the black panther movie case. *Computational and Mathematical Organization Theory*, 25(1):72–84.
- [7] Badanidiyuru, A., Kleinberg, R., and Slivkins, A. (2013). Bandits with knapsacks. In *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, pages 207–216. IEEE.
- [8] Baltrušaitis, T., Ahuja, C., and Morency, L.-P. (2018). Multimodal machine learning: A survey and taxonomy. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.
- [9] Becker, H., Naaman, M., and Gravano, L. (2011). Beyond trending topics: Real-world event identification on twitter. In *Fifth international AAAI conference on weblogs and social media*.
- [10] Beigi, G., Tang, J., and Liu, H. (2016). Signed link analysis in social media networks. In *Tenth International AAAI Conference on Web and Social Media*.
- [11] Bengio, Y., Ducharme, R., Vincent, P., and Jauvin, C. (2003). A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155.
- [12] Beskow, D. M., Kumar, S., and Carley, K. M. (2020). The evolution of political memes: Detecting and characterizing internet memes with multi-modal deep learning. *Information Processing & Management*, 57(2):102170.
- [13] Blum, A. and Mitchell, T. (1998). Combining labeled and unlabeled data with co-training. In *Proceedings of the eleventh annual conference on Computational learning theory*, pages 92–100. ACM.
- [14] Bollen, J., Mao, H., and Zeng, X. (2011). Twitter mood predicts the stock market. *Journal*

of computational science, 2(1):1–8.

- [15] Bonchi, F. (2011). Influence propagation in social networks: A data mining perspective. *IEEE Intelligent Informatics Bulletin*, 12(1):8–16.
- [16] Bowles, N. (2018). The mainstreaming of political memes online. *New York Times*.
- [17] Buntain, C. and Golbeck, J. (2017). Automatically identifying fake news in popular twitter threads. In *2017 IEEE International Conference on Smart Cloud (SmartCloud)*, pages 208–215. IEEE.
- [18] Campan, A., Atnafu, T., Truta, T. M., and Nolan, J. (2018). Is data collection through twitter streaming api useful for academic research? In *2018 IEEE International Conference on Big Data (Big Data)*, pages 3638–3643. IEEE.
- [19] Chatfield, A. T. and Brajawidagda, U. (2013). Twitter early tsunami warning system: A case study in indonesia’s natural disaster management. In *2013 46th Hawaii International Conference on System Sciences*, pages 2050–2060. IEEE.
- [20] Chen, T. and Guestrin, C. (2016). Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794. ACM.
- [21] Chen, W., Lakshmanan, L. V., and Castillo, C. (2013). Information and influence propagation in social networks. *Synthesis Lectures on Data Management*, 5(4):1–177.
- [22] Choi, E., Rashkin, H., Zettlemoyer, L., and Choi, Y. (2016). Document-level sentiment inference with social, faction, and discourse context. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 333–343.
- [23] Colleoni, E., Rozza, A., and Arvidsson, A. (2014). Echo chamber or public sphere? predicting political orientation and measuring political homophily in twitter using big data. *Journal of Communication*, 64(2):317–332.
- [24] Collobert, R. and Weston, J. (2008). A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM.
- [25] Conover, M., Ratkiewicz, J., Francisco, M. R., Gonçalves, B., Menczer, F., and Flammini, A. (2011). Political polarization on twitter. *Icwsn*, 133:89–96.
- [26] Conover, M. D., Goncalves, B., Ratkiewicz, J., Flammini, A., and Menczer, F. (2011). Predicting the political alignment of twitter users. In *2011 IEEE Third International Conference on Privacy, Security, Risk and Trust and 2011 IEEE Third International Conference on Social Computing*, pages 192–199.
- [27] Dang, A., Smit, M., Moh’d, A., Minghim, R., and Milios, E. (2016). Toward understanding how users respond to rumours in social media. In *2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 777–784. IEEE.
- [28] Derczynski, L., Bontcheva, K., Liakata, M., Procter, R., Hoi, G. W. S., and Zubiaga, A. (2017). Semeval-2017 task 8: Rumoureal: Determining rumour veracity and support for rumours. *arXiv preprint arXiv:1704.05972*.

- [29] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.
- [30] Du, J., Ling, C. X., and Zhou, Z.-H. (2010). When does cotraining work in real data? *IEEE Transactions on Knowledge and Data Engineering*, 23(5):788–799.
- [31] Du Bois, J. W. (2007). The stance triangle. *Stancetaking in discourse: Subjectivity, evaluation, interaction*, 164(3):139–182.
- [32] Eck, D. and Schmidhuber, J. (2002). Finding temporal structure in music: Blues improvisation with lstm recurrent networks. In *Neural Networks for Signal Processing, 2002. Proceedings of the 2002 12th IEEE Workshop on*, pages 747–756. IEEE.
- [33] Enayet, O. and El-Beltagy, S. R. (2017). Niletmg at semeval-2017 task 8: Determining rumour and veracity support for rumours on twitter. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 470–474.
- [34] Evans, A. (2016). Stance and identity in twitter hashtags. *Language@ internet*, 13(1).
- [35] Felbo, B., Mislove, A., Søggaard, A., Rahwan, I., and Lehmann, S. (2017). Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm. *arXiv preprint arXiv:1708.00524*.
- [36] Ferrara, E. (2015). Manipulation and abuse on social media. *ACM SIGWEB Newsletter*, (Spring):4.
- [37] Garimella, K., De Francisci Morales, G., Gionis, A., and Mathioudakis, M. (2018a). Political discourse on social media: Echo chambers, gatekeepers, and the price of bipartisanship. In *Proceedings of the 2018 World Wide Web Conference, WWW '18*, pages 913–922, Republic and Canton of Geneva, Switzerland. International World Wide Web Conferences Steering Committee.
- [38] Garimella, K., Morales, G. D. F., Gionis, A., and Mathioudakis, M. (2018b). Quantifying controversy on social media. *Trans. Soc. Comput.*, 1(1):3:1–3:27.
- [39] Garimella, K., Morales, G. D. F., Gionis, A., and Mathioudakis, M. (2018c). Quantifying controversy on social media. *ACM Transactions on Social Computing*, 1(1):3.
- [40] Garimella, K., Weber, I., and De Choudhury, M. (2016). Quote rts on twitter: usage of the new feature for political discourse. In *Proceedings of the 8th ACM Conference on Web Science*, pages 200–204. ACM.
- [41] Gilbert, E., Bergstrom, T., and Karahalios, K. (2009). Blogs are echo chambers: Blogs are echo chambers. In *2009 42nd Hawaii International Conference on System Sciences*, pages 1–10.
- [42] Gildea, D. (2004). Dependencies vs. constituents for tree-based alignment. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*.
- [43] Gillani, N., Yuan, A., Saveski, M., Vosoughi, S., and Roy, D. (2018). Me, my echo chamber, and i: Introspection on social media polarization. In *Proceedings of the 2018 World Wide Web Conference, WWW '18*, pages 823–831, Republic and Canton of Geneva, Switzerland. International World Wide Web Conferences Steering Committee.

- [44] Go, A., Bhayani, R., and Huang, L. (2009). Twitter sentiment classification using distant supervision. *CS224N Project Report, Stanford*, 1(12).
- [45] Grover, A. and Leskovec, J. (2016). node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 855–864. ACM.
- [46] Gurini, D. F., Gasparetti, F., Micarelli, A., and Sansonetti, G. (2014). iscur: Interest and sentiment-based community detection for user recommendation on twitter. In Dimitrova, V., Kuflik, T., Chin, D., Ricci, F., Dolog, P., and Houben, G.-J., editors, *User Modeling, Adaptation, and Personalization*, pages 314–319, Cham. Springer International Publishing.
- [47] Han, B., Yao, Q., Yu, X., Niu, G., Xu, M., Hu, W., Tsang, I., and Sugiyama, M. (2018). Co-teaching: Robust training of deep neural networks with extremely noisy labels. In *Advances in Neural Information Processing Systems*, pages 8536–8546.
- [48] Hasan, K. S. and Ng, V. (2013a). Extra-linguistic constraints on stance recognition in ideological debates. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 816–821.
- [49] Hasan, K. S. and Ng, V. (2013b). Stance classification of ideological debates: Data, models, features, and constraints. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pages 1348–1356.
- [50] Hassan, N., Li, C., and Tremayne, M. (2015). Detecting check-worthy factual claims in presidential debates. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 1835–1838. ACM.
- [51] Hinton, G., Deng, L., Yu, D., Dahl, G. E., Mohamed, A.-r., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T. N., et al. (2012). Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97.
- [52] Hirose, H. and Wang, L. (2012). Prediction of infectious disease spread using twitter: A case of influenza. In *2012 Fifth International Symposium on Parallel Architectures, Algorithms and Programming*, pages 100–105. IEEE.
- [53] Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.
- [54] Huang, X., Li, J., and Hu, X. (2017). Label informed attributed network embedding. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, pages 731–739. ACM.
- [55] Imran, M., Mitra, P., and Castillo, C. (2016). Twitter as a lifeline: Human-annotated twitter corpora for nlp of crisis-related messages. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Paris, France. European Language Resources Association (ELRA).
- [56] Jin, S.-A. A. and Phua, J. (2014). Following celebrities tweets about brands: The impact of twitter-based electronic word-of-mouth on consumers source credibility perception, buying intention, and social identification with celebrities. *Journal of Advertising*, 43(2):181–195.

- [57] Jin, Z., Cao, J., Zhang, Y., and Luo, J. (2016). News verification by exploiting conflicting social viewpoints in microblogs. In *AAAI*, pages 2972–2978.
- [58] Johnson, K. and Goldwasser, D. (2016). Identifying stance by analyzing political discourse on twitter. In *Proceedings of the First Workshop on NLP and Computational Social Science*, pages 66–75.
- [59] Joseph, K., Friedland, L., Hobbs, W., Lazer, D., and Tsur, O. (2017). Constance: Modeling annotation contexts to improve stance classification. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1115–1124. Association for Computational Linguistics.
- [60] Kiesling, S. F., Pavalanathan, U., Fitzpatrick, J., Han, X., and Eisenstein, J. (2018). Interactional stancetaking in online forums. *Computational Linguistics*, 44(4):683–718.
- [61] Kim, J., Park, H., Lee, J.-E., and Kang, U. (2018). Side: representation learning in signed directed networks. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web*, pages 509–518. International World Wide Web Conferences Steering Committee.
- [62] Kim, Y. (2014). Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.
- [63] Kiros, R., Zhu, Y., Salakhutdinov, R. R., Zemel, R., Urtasun, R., Torralba, A., and Fidler, S. (2015). Skip-thought vectors. In *Advances in neural information processing systems*, pages 3294–3302.
- [64] Kochkina, E., Liakata, M., and Zubiaga, A. (2018). All-in-one: Multi-task learning for rumour verification. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3402–3413. Association for Computational Linguistics.
- [65] Kohavi, R., Deng, A., Frasca, B., Walker, T., Xu, Y., and Pohlmann, N. (2013). Online controlled experiments at large scale. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1168–1176. ACM.
- [66] Kumar, S. and Carley, K. M. (2016). Approaches to understanding the motivations behind cyber attacks. In *2016 IEEE Conference on Intelligence and Security Informatics (ISI)*, pages 307–309.
- [67] Kumar, S. and Carley, K. M. (2018). People2vec: Learning latent representations of users using their social-media activities. In *International Conference on Social Computing, Behavioral-Cultural Modeling and Prediction and Behavior Representation in Modeling and Simulation*, pages 154–163. Springer.
- [68] Kumar, S. and Carley, K. M. (2019a). Tree lstms with convolution units to predict stance and rumor veracity in social media conversations. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5047–5058.
- [69] Kumar, S. and Carley, K. M. (2019b). What to track on the twitter streaming api? a knapsack bandits approach to dynamically update the search terms. In *Proceedings of the 2019 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining, ASONAM 19*, page 158163, New York, NY, USA. Association for Computing Machinery.
- [70] Lahoti, P., Garimella, K., and Gionis, A. (2018). Joint non-negative matrix factorization

- for learning ideological leaning on twitter. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, pages 351–359. ACM.
- [71] Lamb, A., Paul, M. J., and Dredze, M. (2013). Separating fact from fear: Tracking flu infections on twitter. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 789–795.
- [72] LeCun, Y., Kavukcuoglu, K., and Farabet, C. (2010). Convolutional networks and applications in vision. In *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*, pages 253–256. IEEE.
- [73] Leskovec, J., Huttenlocher, D., and Kleinberg, J. (2010). Signed networks in social media. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pages 1361–1370. ACM.
- [74] Leskovec, J., Krause, A., Guestrin, C., Faloutsos, C., Faloutsos, C., VanBriesen, J., and Glance, N. (2007). Cost-effective outbreak detection in networks. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 420–429. ACM.
- [75] Li, J., Luong, T., Jurafsky, D., and Hovy, E. (2015). When are tree structures necessary for deep learning of representations? In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2304–2314, Lisbon, Portugal. Association for Computational Linguistics.
- [76] Li, J., Ritter, A., Cardie, C., and Hovy, E. (2014). Major life event extraction from twitter based on congratulations/condolences speech acts. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1997–2007.
- [77] Lim, K. W. and Buntine, W. (2014). Twitter opinion topic model: Extracting product opinions from tweets by leveraging hashtags and sentiment lexicon. In *Proceedings of the 23rd ACM international conference on conference on information and knowledge management*, pages 1319–1328. ACM.
- [78] Liu, B. (2012). Sentiment analysis and opinion mining. *Synthesis lectures on human language technologies*, 5(1):1–167.
- [79] Liu, C., Li, W., Demarest, B., Chen, Y., Couture, S., Dakota, D., Haduong, N., Kaufman, N., Lamont, A., Pancholi, M., et al. (2016). Iucl at semeval-2016 task 6: An ensemble model for stance detection in twitter. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 394–400.
- [80] Lu, H., Caverlee, J., and Niu, W. (2015a). Biaswatch: A lightweight system for discovering and tracking topic-sensitive opinion bias in social media. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 213–222. ACM.
- [81] Lu, H., Caverlee, J., and Niu, W. (2015b). Biaswatch: A lightweight system for discovering and tracking topic-sensitive opinion bias in social media. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management, CIKM '15*, pages

213–222, New York, NY, USA. ACM.

- [82] Lukasik, M., Cohn, T., and Bontcheva, K. (2015). Classifying tweet level judgements of rumours in social media. In *EMNLP*.
- [83] Lukasik, M., Srijith, P., Vu, D., Bontcheva, K., Zubiaga, A., and Cohn, T. (2016). Hawkes processes for continuous time sequence classification: an application to rumour stance classification in twitter. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 393–398.
- [84] Ma, J., Gao, W., and Wong, K.-F. (2018). Rumor detection on twitter with tree-structured recursive neural networks. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1980–1989, Melbourne, Australia. Association for Computational Linguistics.
- [85] Magazine, M. J. and Chern, M.-S. (1984). A note on approximation schemes for multidimensional knapsack problems. *Mathematics of Operations Research*, 9(2):244–247.
- [86] Margetts, H. (2017). Political behaviour and the acoustics of social media. *Nature Human Behaviour*.
- [87] Mejova, Y., Srinivasan, P., and Boynton, B. (2013). Gop primary season on twitter: popular political sentiment in social media. In *Proceedings of the sixth ACM international conference on Web search and data mining*, pages 517–526. ACM.
- [88] Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.
- [89] Misra, A., Ecker, B., Handleman, T., Hahn, N., and Walker, M. (2016). Nlds-ucsc at semeval-2016 task 6: a semi-supervised approach to detecting stance in tweets. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 420–427.
- [90] Mohammad, S., Kiritchenko, S., Sobhani, P., Zhu, X., and Cherry, C. (2016). 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. Association for Computational Linguistics.
- [91] Mohammad, S. M., Sobhani, P., and Kiritchenko, S. (2017). Stance and sentiment in tweets. *ACM Transactions on Internet Technology (TOIT)*, 17(3):26.
- [92] Morency, L.-P., Mihalcea, R., and Doshi, P. (2011). Towards multimodal sentiment analysis: Harvesting opinions from the web. In *Proceedings of the 13th international conference on multimodal interfaces*, pages 169–176. ACM.
- [93] Nigam, K. and Ghani, R. (2000). Analyzing the effectiveness and applicability of co-training. In *Cikm*, volume 5, page 3.
- [94] Nigam, K., McCallum, A., Thrun, S., Mitchell, T., et al. (1998). Learning to classify text from labeled and unlabeled documents. *AAAI/IAAI*, 792:6.
- [95] Ozer, M., Kim, N., and Davulcu, H. (2016). Community detection in political twitter networks using nonnegative matrix factorization methods. In *Advances in Social Networks Analysis and Mining (ASONAM), 2016 IEEE/ACM International Conference on*, pages 81–

88. IEEE.

- [96] Pennington, J., Socher, R., and Manning, C. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.
- [97] Perozzi, B., Al-Rfou, R., and Skiena, S. (2014). Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710. ACM.
- [98] Platanios, E. A., Dubey, A., and Mitchell, T. (2016). Estimating accuracy from unlabeled data: A bayesian approach. In *International Conference on Machine Learning*, pages 1416–1425.
- [99] Procter, R., Vis, F., and Voss, A. (2013). Reading the riots on twitter: methodological innovation for the analysis of big data. *International journal of social research methodology*, 16(3):197–214.
- [100] Rashkin, H., Choi, E., Jang, J. Y., Volkova, S., and Choi, Y. (2017). Truth of varying shades: Analyzing language in fake news and political fact-checking. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2921–2927.
- [101] Rennie, J. and McCallum, A. (1999). Using reinforcement learning to spider the web efficiently. In *Proceedings of the Sixteenth International Conference on Machine Learning, ICML '99*, pages 335–343, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.
- [102] Rubin, V., Conroy, N., Chen, Y., and Cornwell, S. (2016). Fake news or truth? using satirical cues to detect potentially misleading news. In *Proceedings of the Second Workshop on Computational Approaches to Deception Detection*, pages 7–17.
- [103] Rubin, V. L. and Lukoianova, T. (2015). Truth and deception at the rhetorical structure level. *Journal of the Association for Information Science and Technology*, 66(5):905–917.
- [104] Salton, G. and Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. *Information processing & management*, 24(5):513–523.
- [105] Saraçlı, S., Doğan, N., and Doğan, İ. (2013). Comparison of hierarchical cluster analysis methods by cophenetic correlation. *Journal of Inequalities and Applications*, 2013(1):203.
- [106] Schifferes, S., Newman, N., Thurman, N., Corney, D., Göker, A., and Martin, C. (2014). Identifying and verifying news through social media: Developing a user-centred tool for professional journalists. *Digital Journalism*, 2(3):406–418.
- [107] Sharma, K., Qian, F., Jiang, H., Ruchansky, N., Zhang, M., and Liu, Y. (2019). Combating fake news: A survey on identification and mitigation techniques. *ACM Trans. Intell. Syst. Technol.*, 10(3):21:1–21:42.
- [108] Sobhani, P., Inkpen, D., and Matwin, S. (2015). From argumentation mining to stance classification. In *Proceedings of the 2nd Workshop on Argumentation Mining*, pages 67–77.
- [109] Sobhani, P., Inkpen, D., and Zhu, X. (2017). A dataset for multi-target stance detection. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, volume 2, pages 551–557.

- [110] Somasundaran, S. and Wiebe, J. (2010). Recognizing stances in ideological on-line debates. In *Proceedings of the NAACL HLT 2010 Workshop on Computational Approaches to Analysis and Generation of Emotion in Text*, pages 116–124. Association for Computational Linguistics.
- [111] Sridhar, D., Getoor, L., and Walker, M. (2014). Collective stance classification of posts in online debate forums. In *Proceedings of the Joint Workshop on Social Dynamics and Personal Attributes in Social Media*, pages 109–117.
- [112] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.
- [113] Tacchini, E., Ballarin, G., Della Vedova, M. L., Moret, S., and de Alfaro, L. (2017). Some like it hoax: Automated fake news detection in social networks. *arXiv preprint arXiv:1704.07506*.
- [114] Tai, K. S., Socher, R., and Manning, C. D. (2015). Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1556–1566, Beijing, China. Association for Computational Linguistics.
- [115] Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., and Mei, Q. (2015). Line: Large-scale information network embedding. In *Proceedings of the 24th International Conference on World Wide Web*, pages 1067–1077. International World Wide Web Conferences Steering Committee.
- [116] Toth, P. (1980). Dynamic programming algorithms for the zero-one knapsack problem. *Computing*, 25(1):29–45.
- [117] Tran-Thanh, L., Chapman, A., Rogers, A., and Jennings, N. R. (2012). Knapsack based optimal policies for budget-limited multi-armed bandits. In *Twenty-Sixth AAAI Conference on Artificial Intelligence*.
- [118] Tu, C., Liu, H., Liu, Z., and Sun, M. (2017). Cane: Context-aware network embedding for relation modeling. In *Proceedings of ACL*.
- [119] Vieweg, S., Hughes, A. L., Starbird, K., and Palen, L. (2010). Microblogging during two natural hazards events: what twitter may contribute to situational awareness. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pages 1079–1088. ACM.
- [120] Visée, M., Teghem, J., Pirlot, M., and Ulungu, E. L. (1998). Two-phases method and branch and bound procedures to solve the bi-objective knapsack problem. *Journal of Global Optimization*, 12(2):139–155.
- [121] Volkova, S., Shaffer, K., Jang, J. Y., and Hodas, N. (2017). Separating facts from fiction: Linguistic models to classify suspicious and trusted news posts on twitter. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 647–653.
- [122] Vosoughi, S., Roy, D., and Aral, S. (2018). The spread of true and false news online.

Science, 359(6380):1146–1151.

- [123] Wang, S., Tang, J., Aggarwal, C., Chang, Y., and Liu, H. (2017). Signed network embedding in social media. In *Proceedings of the 2017 SIAM International Conference on Data Mining*, pages 327–335. SIAM.
- [124] Wang, W., Knight, K., and Marcu, D. (2007). Binarizing syntax trees to improve syntax-based machine translation accuracy. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*.
- [125] Wang, Y., Callan, J., and Zheng, B. (2015). Should we use the sample? analyzing datasets sampled from twitters stream api. *ACM Transactions on the Web (TWEB)*, 9(3):13.
- [126] Wei, W., Zhang, X., Liu, X., Chen, W., and Wang, T. (2016). pkudblab at semeval-2016 task 6: A specific convolutional neural network system for effective stance detection. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 384–388.
- [127] West, R., Paskov, H. S., Leskovec, J., and Potts, C. (2014). Exploiting social network structure for person-to-person sentiment analysis. *arXiv preprint arXiv:1409.2450*.
- [128] Williams, A., Nangia, N., and Bowman, S. R. (2017). A broad-coverage challenge corpus for sentence understanding through inference. *arXiv preprint arXiv:1704.05426*.
- [129] Wong, F. M. F., Tan, C. W., Sen, S., and Chiang, M. (2016). Quantifying political leaning from tweets, retweets, and retweeters. *IEEE Transactions on Knowledge and Data Engineering*, 28(8):2158–2172.
- [130] Xing, C., Wang, Y., Liu, J., Huang, Y., and Ma, W.-Y. (2016). Hashtag-based sub-event discovery using mutually generative lda in twitter. In *AAAI*, pages 2666–2672.
- [131] Xu, P. (1998). Truncated svd methods for discrete linear ill-posed problems. *Geophysical Journal International*, 135(2):505–514.
- [132] Yu, X., Chakraborty, S., and Brady, E. (2019). A co-training model with label propagation on a bipartite graph to identify online users with disabilities. In *Proceedings of the International AAAI Conference on Web and Social Media*, volume 13, pages 667–670.
- [133] Zhang, X., Zhao, J., and LeCun, Y. (2015). Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pages 649–657.
- [134] Zhou, K., Shu, C., Li, B., and Lau, J. H. (2019). Early rumour detection. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1614–1623, Minneapolis, Minnesota. Association for Computational Linguistics.
- [135] Zubiaga, A., Kochkina, E., Liakata, M., Procter, R., and Lukasik, M. (2016a). Stance classification in rumours as a sequential task exploiting the tree structure of social media conversations. *arXiv preprint arXiv:1609.09028*.
- [136] Zubiaga, A., Kochkina, E., Liakata, M., Procter, R., Lukasik, M., Bontcheva, K., Cohn, T., and Augenstein, I. (2018). Discourse-aware rumour stance classification in social media using sequential classifiers. *Information Processing & Management*, 54(2):273–290.

- [137] Zubiaga, A., Liakata, M., Procter, R., Bontcheva, K., and Tolmie, P. (2015). Crowdsourcing the annotation of rumourous conversations in social media. In *Proceedings of the 24th International Conference on World Wide Web*, pages 347–353. ACM.
- [138] Zubiaga, A., Liakata, M., Procter, R., Hoi, G. W. S., and Tolmie, P. (2016b). Analysing how people orient to and spread rumours in social media by looking at conversational threads. *PloS one*, 11(3):e0150989.

Appendix A

Better ways of collecting Twitter data: What to Track on the Twitter Streaming API? A Knapsack Bandits Approach to Dynamically Update the Search Terms

A.1 Introduction

Imagine a disaster scenario. An earthquake hits the city of Anchorage. As people in Anchorage start responding to the event, many people start to tweet about their situation. Many research agencies and disaster response teams monitor Twitter to find such tweets and to respond to them as soon as possible. The standard way to monitor events on Twitter is to use the Twitter streaming API that allows tracking search terms. The streaming API is limited to approximately 1% (or 10% for the paid service) of Twitter data, and the proportion of tweets generated to tweets collected is even less if we track trendy terms [18]. Therefore, despite the best efforts, in events like an earthquake, often a large fraction of useful tweets do not reach the agencies because they do not contain the exact search terms which the agencies are monitoring. For example, if the disaster agency is tracking ‘#earthquake’, the agency will miss tweets that contain ‘#Alaska’ (e.g. Tweet in Fig. A.1). Their data collection could be improved by adding new relevant search terms as events unfold, e.g. using ‘#Anchorage’ and ‘#Alaska’ soon after the earthquake in Anchorage.

Twitter has been shown to be useful in disasters [19, 119]. However, that is not its only use case. Twitter remains a popular source of data both for researchers [6, 52, 66] and social-media analytic companies. The common approach to collect tweets is to use a set of words-of-interest as search terms to track on Twitter streaming API. However, as events happen and discussions evolve, the relevant search terms change with time. Thereby, if the search terms are not updated, the old search terms get misaligned with the goals of the data collection. For example, in the earthquake scenario discussed earlier, the new search term ‘#Alaska’ could get dis-aligned with the goal of collecting data on earthquakes in a few days. This begs a question. Is it possible to use the goal of the search in the data collection itself to collect more relevant data over time?

In this research, we propose an iterative two step online algorithm that: 1) gets live data

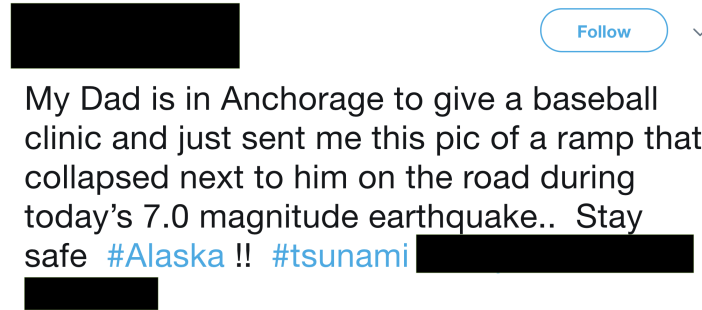


Figure A.1: A sample Tweet sent after an earthquake with some parts blacked out to preserve anonymity. A data collection approach that only tracks ‘#earthquake’ on Twitter streaming API would miss such tweets. In this research, we propose an approach to dynamically update the search terms based on prior collected data. For example, if the goal is to find Tweets relevant to earthquakes then our proposed algorithm would be able to find new relevant search terms like ‘#Alaska’ in case of an earthquake in Alaska.

from Twitter using a set of search-terms, then 2) finds the next set of search-terms based on the data retrieved (in the previous step or in the last few iterations). Rather than formulating the data collection goal as a set of fixed search terms, our approach allows using the goal in a more flexible way (e.g. as a text classifier) and our solution embeds this higher level formulation in the data collection process. We model the search-terms selection problem as a knapsack problem and solve it using standard knapsack and knapsack bandits. The knapsack bandits effectively handle exploration (new search terms to explore) and exploitation (keep using the most useful search terms) and respect the constraints of data collection such as ‘number of terms that can be used’ or ‘the amount of data that can be downloaded’ in a time window. We summarize our main contributions below:

- We suggest ways to collect more relevant Twitter data using the Twitter streaming API. We model the Twitter data collection as a knapsack problem with cost, value and constraints (Section A.3), and propose two solutions.
- The first solution uses a dynamic programming based knapsack solver that estimates the cost and the value of search terms independently in each time iteration (Section A.4).
- The second solution proposes a multi-armed-bandit approach to estimate the cost and value of search terms over multiple time iterations (Section A.5).
- To the best of our knowledge, this is the first work that suggests a principled approach to dynamically update the search terms while staying relevant to the goal of the data collection. We show the utility of our approach using a real example (Section A.6).

Section A.2 provides background on Twitter data collection, highlighting the limitations of the Twitter streaming API. We describe the related prior research that are relevant to this work in Sec. A.7. Finally, at the end, we conclude and provide directions for future research. Code to rerun the experiments is available on GitHub ¹.

¹https://github.com/CASOS-IDeaS-CMU/What_to_track_on_Twitter_Streaming_API

A.2 Twitter Data Collection Background

Twitter Streaming API allows three parameters to search the real-time data which are ‘follow’, ‘track’ and ‘locations’². Here we focus on ‘track’ as that is commonly used to track a comma-separated list of phrases (e.g. words, mentions, #hashtags). As mentioned earlier, given a topic of interest, the most common approach is to use intuition to come up with a few generic phrases that overlap with the discussions on the topic. For example, if someone is interested in finding information about earthquakes, the phrases to use could be ‘#earthquake’, ‘earthquake’ or ‘earthquake now’. Though this is how Twitter API is commonly used, this approach has a few limitations.

Some phrases are trendy and result in a vast amount of data, much of which is not relevant to the goals of the data collection. Therefore the collected data needs to be filtered later which results in processing and discarding a significant proportion of data. There is one more problem. If multiple search terms are used, the majority of tweets obtained using the API will be from the more commonly used phrases. Searching for more terms results in less number of tweets per search term [18] and, in our experience, the returned data never exceeds around 25GB per day (raw JSON files obtained using Tweepy library³) irrespective of the number of search terms used (tested on a computer with 1 Gbps Internet connection speed at the Carnegie Mellon University campus). This limitation of Twitter is not well documented so to better understand it, we used the Twitter Streaming API to track all 195 country names in English.

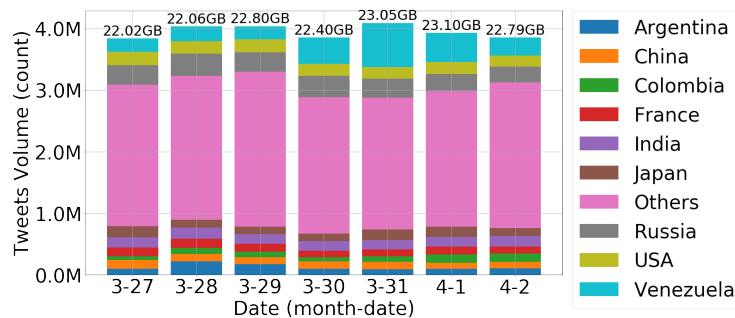


Figure A.2: Tweets volume obtained on different days using all country names as search terms. On y axis, ‘M’ indicates tweets count in millions. The total data received each day is shown on the top of the day’s bar. As we can observe, though there is a wide variation in tweets with certain country-names (e.g. check Venezuela), but still the total volume of tweets has remained between 22 GB and 23.1 GB and the total count is approximately 4 million.

In Fig. A.2, we show the volume and the quantity of data obtained by searching 195 countries names for over a week. As we can observe, though there is a wide variation in tweets from specific countries (e.g., check Venezuela), but still the total volume of tweets has remained between 22 GB and 23.1 GB and the total count of tweets is approximately 4 million. This finding confirms the observations in [18] where the authors find that Twitter returns around 2900 tweets per minute when search terms are used with Twitter streaming API. 2900 tweets per minute is

²<https://developer.twitter.com/en/docs/tweets/filter-realtime/guides/basic-stream-parameters.html>

³https://tweepy.readthedocs.io/en/v3.5.0/streaming_how_to.html

4,176,000 tweets in a day i.e. approximately 4 million. Note that these limits are different while using the Twitter Streaming API without any search terms.

A.3 Problem Formulation

In collecting Twitter data, there is a cost in collecting data, there is some value of the data collected, and there are some constraints. We describe these next:

1. Cost: Cost in data collection is due to the costs of data streaming (e.g., the internet bandwidth), data storage and/or data processing. Because these costs are proportional to the volume of the data obtained, to keep our model simple, we aggregate the different costs and call the overall cost as w_i , where i is the search term index. Let x_i be the number of tweets retrieved per minute by searching the i^{th} search term. Cost w_i is a function of the amount of data x_i . We expect the cost to be low if the volume of data retrieved is low, but if the data volume x_i is large, the cost should much higher as it reduces the collection of data associated with other search terms (as discussed earlier). Therefore, the cost function should be non-linear and should satisfy the following conditions: a) Cost is proportional to number of tweets collected if the total volume of tweets is low b) If the total volume obtained reaches the maximum limit (2900 per minute), then the cost should be very high as we can't get data at any higher rate. Many functions can possibly respect these conditions. We model the cost function as a non-linear logit function (see Fig. A.3). As shown in the figure, this function is approximately linear till 2000 tweets per minute and increases rapidly later.

$$w_i(x_i) = \log \left(\frac{(x_i + 2900)/5800}{1 - (x_i + 2900)/5800} \right) \tag{A.1}$$

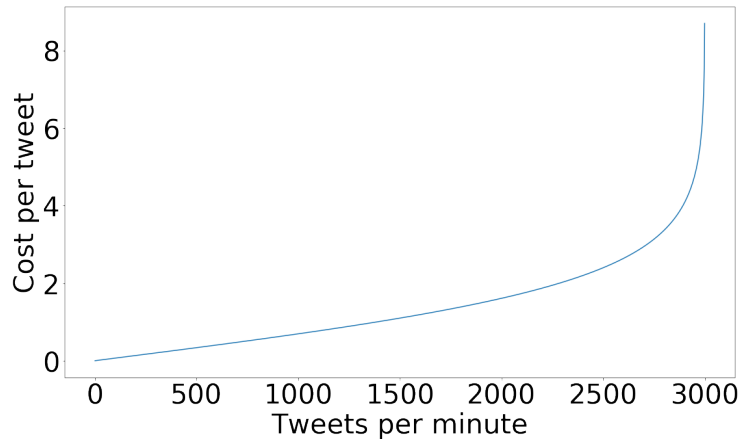


Figure A.3: Cost function plot

The goal of this cost function is to encourage search terms that results in smaller fraction of tweets.

2. Value: This value of the search term i is the mean utility of the data. For example, in the case of an earthquake, the value is a function of the fraction of tweets that are relevant to a real shock. We call this value v_i where again i is the search term index. v_i is estimated based on the goals of the data collection and could be as simple as a text pattern match. For instance, if one is interested in the text of the tweets matching some pattern, we can model the value function as:

$$Value_Func(t_k) = \begin{cases} 1 & \text{if } t_k \text{ matches } p \\ 0 & \text{otherwise} \end{cases}$$

and $v_i = \text{Mean}(Value_Func(t_k))$ where t_k is a tweet associated with search term with index i .

3. Constraints: While collecting Twitter data, a critical limitation is the total amount of data that we can download. As discussed earlier, it appears that there is a hard limit on the amount of streaming data that can be obtained using a single API connection. We define this constraint as W where $W \leq 2900$ tweets per minute (based on the Internet bandwidth of the system or the expected amount of data to be collected).

Thus, $\sum_{i \in I(t)} w_i(x_i(t)) \leq W$ in every iteration $t = 0, 1, 2, \dots$, where $I(t)$ is the set of indices of the selected search terms. There are other possible constraints as well e.g. Twitter limits the number of search terms to 400⁴. To satisfy the search term limit, we have: $\sum_{i \in I(t)} 1 \leq 400$.

Using cost, value and constraints, we define our optimization problem as:

$$\max\left(\sum_i \left[v_i(x_i(t))\right]\right) \tag{A.2}$$

subject to:

$$\sum_{i \in I(t)} w_i(x_i(t)) \leq W \tag{A.3}$$

and

$$\sum_{i \in I(t)} 1 \leq 400 \tag{A.4}$$

at iteration $t = 0, 1, 2, \dots$, where x_i is data collected for the i^{th} search term, and $I(t)$ is the index of all ‘search terms’ used at time t . The objective is to maximize the expected value at time $t + 1$ based on the estimates of v_i and w_i at time t .

Given cost, value and constraints, the standard approach is to model such problems as a knapsack problem (described next).

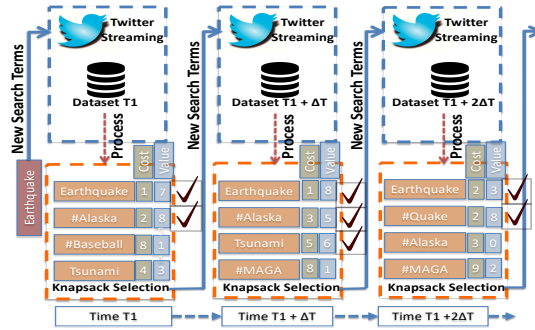


Figure A.4: In every time iteration, a set of search-terms are used to get data from the Twitter Streaming API. The data is then processed to find high frequency terms. For each of these terms, value and cost are estimated (shown with light blue and light green background respectively) based on the tweets associated with each term. Cost and value are then used to find the next set of search-terms using a knapsack solver that also considers the constraints of the data collection. The result of one iteration is used as the search-terms for the next iteration.

A.4 New Search Terms as a Knapsack Problem

We first solve the simpler version of the problem in which cost and value are estimated each iteration. An iteration consists of a small batch of data obtained by connecting to the Twitter streaming API for some time. The time duration of the iteration depends on how often the search terms need to be updated based on the goal of the data collection. For example, in case of an earthquake, since such events are instantaneous, an iteration could be of short time like a few minutes. In contrast, for slow changing goals like political discussions, an iteration could be of larger time duration like 30 minutes. At the end of the iteration, the tweets dataset is processed to find the high-frequency terms which are the potential search-term candidates for the next batch. For each of these terms, value and cost are estimated based on the tweets associated with each term. The search terms are first filtered to remove any unwanted content (i.e. stop words or pornographic content). Cost and value for the filtered terms are estimated using the cost and value function as described in the last section. Knapsack problems, though NP-hard [85], have many efficient solutions [120]. We use a dynamic programming based knapsack solver to find the next set of search terms [116]. We show the steps in Fig. A.4, and summarize the steps as an algorithm which we name as ‘Dynamically Update Search Terms’ (DUST1) (see Alg. 4).

DUST1 algorithm returns a set of new terms (in addition to seed terms) after each iteration. Though useful for dynamically updating the search, this approach has two limitations: 1) The approach only considers the current data, thereby ignoring the cost and value estimated in previous iterations. Because the streaming API only consists of a small fraction of the total tweets, it’s possible that in a particular iteration, there is no data from a search-term though the search-term is generally useful. 2) The approach does not consider the confidence in estimating the value of the search terms. For example, if a search-term ‘X1’ had only a single tweet of high value, the mean-value based algorithm is more likely to suggest it when compared to another term ‘X2’

⁴<https://developer.twitter.com/en/docs/tweets/filter-realtime/overview/statuses-filter.html>

Algorithm 4 DUST1: Dynamically Update Search Terms

Require: *data* is the tweets collected

```
1: function DUST1(data) terms, terms_tweets_dict Process(data) utility_scores[]
2:   for  $k \leftarrow 0$  to  $\text{len}(\textit{terms})$  do
3:      $\textit{term} \leftarrow \textit{terms}[k]$ 
4:      $\textit{term\_tweets} \leftarrow \textit{terms\_tweets\_dict}[\textit{term}]$   $\textit{valueMean}(\text{Value\_Func}(\textit{term\_tweets}))$ 
        $\textit{costMean}(\text{Cost\_Func}(\textit{term\_tweets}))$ 
5:      $\textit{utility\_scores.add}(\textit{term}, \textit{cost}, \textit{value})$ 
6:   end for
7:    $\textit{search\_terms} \leftarrow \text{KnapsackSolver}(\textit{utility\_scores})$ 
8:   return  $\textit{search\_terms}$ 
9: end function
```

that has a few hundred tweets, many of which are useful. We improve on these two limitations in the next section.

A.5 A MAB Approach to Dynamically Update the Search Terms

Here we propose an algorithm that estimates and maintains cost and value of each search term overtime. A common approach to estimate the utility of different options is the ‘*online controlled testing*’, popularly called A/B testing [65]. Re-looking at our earthquake example, if the options are ‘#Alaska’ and ‘#Canada’, one can wait for certain number of tweets on both the terms to arrive before using A/B test to determine if ‘#Alaska’ is more useful than ‘#Canada’. However, A/B testing requires large enough sample set to derive the confidence of the benefit of option A over B. The opportunity cost of waiting to get the sample set is high in many cases (like the earthquake example) and we want the more useful options to be picked quickly (i.e. greedily) to get more relevant data. In such situations, the framework of multi-armed bandits is preferred⁵. Therefore, given our goal of greedily exploring more useful search terms, we model the problem as a knapsack-bandit where bandits are used to estimate the value of the search terms and a knapsack solver is used to filter the top search-terms that satisfy the constraints.

In this simple case, the goal in Multi-Armed Bandits (MAB) optimization is to estimate the reward of each arm to find the arm which leads to maximum reward over multiple trials. Such MAB problems can be solved using many different strategies. These strategies attempt to strike a balance between exploration and exploitation in different ways. To suit the MAB paradigm to our problem, we need two changes 1) need to select a set of search-terms that are more useful (in contrast MAB selects the best search-term). 2) selected terms should also satisfy the data collection constraints that we discussed earlier. Therefore, we use MAB only to estimate the value of search-terms over multiple iterations (using two different strategies) and use the knapsack solver to get the final set of search-terms for data collection. The steps of the approach is shown as an algorithm in Alg. 5 where the *MAB_Strategy* returns a list of

⁵<https://conversionxl.com/blog/bandit-tests/>

(*term*, *cost*, *value*) tuples.

Algorithm 5 DUST2: Update Search Terms using Bandits

Require: *data* is the tweets collected and *util_scores_queue* is a dictionary of FIFO queues that maintains the costs and values of terms

```

1: function DUST2(data) terms, terms_tweets_dict Process(data) utility_scores[]
2:   for  $k \leftarrow 0$  to  $\text{len}(\textit{terms})$  do
3:     term  $\leftarrow \textit{terms}[k]$ 
4:     term_tweets  $\leftarrow \textit{terms\_tweets\_dict}[\textit{term}]$   $\text{valueMean}(\text{Value\_Func}(\textit{term\_tweets}))$ 
        $\text{costMean}(\text{Cost\_Func}(\textit{term\_tweets}))$ 
5:     utility_scores_queue[term].enqueue((cost, value))
6:   end for
7:   utility_scores = MAB_Strategy(utility_scores_queue)
8:   search_terms  $\leftarrow \text{KnapsackSolver}(\textit{utility\_scores})$ 
9:   return search_terms
10: end function

```

Next, we describe two strategies which we use for search-terms selection.

Mean-k Strategy

Mean-k estimates the mean of a function (value or cost) over last k iterations. Thus, mean-k value of term i at iteration n is written as:

$$\hat{v}_i(n) = \frac{\sum_{t=n-k}^n v_i(x_i(t))}{\sum_{t=n-k}^n 1} \quad (\text{A.5})$$

where \hat{v}_i is the estimated mean value of search term i over last k iterations.

Upper-Confidence-Bound (UCB) Strategy

UCB strategy allows for better exploration by giving higher probability to actions for which reward estimate is not available [3]. Intuitively, UCB uses two criterion 1) try if better candidate i.e. $\hat{v}_i(t)$ is large 2) try if less explored i.e. $N_n(i)$ is small.

$$\tilde{v}_i(n) = \left(\hat{v}_i(n) + c \sqrt{\frac{\ln n}{N_n(i)}} \right) \quad (\text{A.6})$$

where $\tilde{v}_i(n)$ is the UCB value associated with term i after iteration n , $N_n(i)$ denotes the total number of times i has been selected before iteration n and c is a parameter to control exploration.

DUST2 algorithm is simple but we found some practical limitations while implementing. a) cost and value estimation of some search terms get stale over time. b) large number of search terms needs to be tracked as we get more and more data which slows down the algorithm. To

resolve ‘a’, we use a FIFO queue (data structure) of limited size to store the cost and value of each search term. In addition, if we don’t get data for a search term in an iteration, an empty ($cost, value$) is added for the term to the queue. With this modifications, we are able to get rid of the stale data. But we still keep on aggregating all search terms. To fix ‘b’, after every iteration, we run a subroutine that removes any search term that contains only empty ($cost, value$) in the queue.

A.6 Experiments and Results

We conducted a data-collection experiment for a few weeks to measure the quantity and the relevancy of data collected using our proposed approach. Our goal is to collect tweets that are relevant to earthquake beyond what could be obtained by using the seed terms. To compare different approaches, we collected data in four different ways: 1) Searching with seed terms ‘#earthquake’ and ‘earthquake’. 2) Searching using the terms suggested by DUST 1(see Alg. 4) 3) Searching using the terms suggested by DUST2 (see Alg. 5) with mean of cost and value estimation for last 10 iterations. 4) searching using the terms suggested by DUST2 (see Alg. 5) with mean for cost and UCB (strategy) for value estimation using last 10 iterations data. For estimating the value of a tweet, we use a small tweets dataset from prior research [55] that has relevancy labels (relevant vs non-relevant) for a set of tweets related to an earthquake in Nepal. We first removed any words that contain ‘Nepal’ and use the filtered dataset to build a Support Vector Machine classifier using TF-IDF features that predicts whether a given tweet is relevant or not. Using a separate test set that is around 40% the entire dataset, we found the accuracy of the classifier to be 81%. We use this classifier in our value function as defined below:

$$Value_Func(t_k) = \begin{cases} 1 & \text{if } EQakeClassifier(t_k) = 1 \\ 0 & \text{otherwise} \end{cases}$$

and $v_i = \text{Mean}(Value_Func(t_k))$ where t_k is a tweet associated with search term with index i .

Using this value function, we present the amount of tweets obtained by just searching for seed terms (only ‘earthquake’), searching for the terms suggest by DUST-1, and by DUST-2 (Mean-10 and UCB strategies) in Fig. A.5. As we can see in the plot, on most days, data collected by DUST-1 exceed the data obtained by just searching for seed terms. For the entire time during for the experiment, we estimate that DUST-1 gets 1.71 times the data obtained using the seed terms. For MAB based search, Mean-10 gets 0.65 times and UCB-10 gets 0.53 times additional data. The total data that was found to be relevant using the relevancy classifier is 3.89 times the total data that was collected only using the seed terms (see Fig. A.6 for trends).

To summarize, even if we discount the accuracy estimate of the relevancy classifier (which can always be improved with more labeled data), we can still expect the gain of our approach to be over 2x times the data collected using the seed terms.

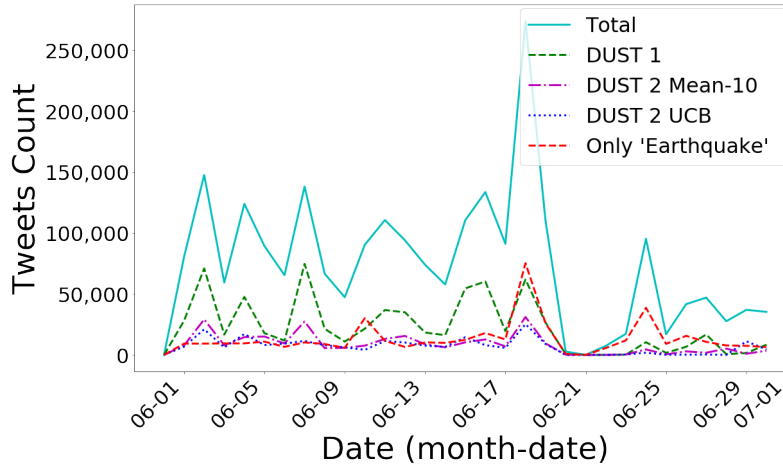


Figure A.5: Results of earthquake data collection using only ‘earthquake’ like search-terms and our proposed approaches.

A.7 Related Work

A.7.1 Twitter Data Collection and Event Detection

A number of researchers have explored and compared Twitter Streaming APIs ([18, 125]). Campan et al. [18] compares the Twitter Streaming API based on popular and not so popular terms and find that if filtering is used for terms that are not very popular, it’s likely that all matching Tweets are provided by Twitter. In contrast, if very popular filtering terms are used, the collected data leads to biased results. Wang et al. [125] used samples obtained from the Spritzer Twitter stream API and Gardenhose Twitter stream API to find that the actual sampling ratios are around 0.95% and 9.6% for Spritzer and Gardenhose respectively. They also suggested that though Spritzer is sufficient when using text terms and URL domains, for hashtags, the small Spritzer sample is not adequate to preserve accurate data statistics. There is also a rich literature on using Twitter for event detection [9]. In most prior work on event detection, the data that was collected apriori. In this work, instead, we suggest a way to get more data by adding new search terms, and to the best of our knowledge, this is the first work of this type.

A.7.2 Multi-Armed Bandit Problems (MAB) and Web Crawling

MABs are commonly used for optimization in noisy environments where there is an exploration (more labels to try) and exploitation (use the best one) trade-off. Many extensions to the standard MAB have been proposed like the contextual-bandit, the collaborative-bandit and the knapsack bandits [7]. In particular, knapsack bandits are useful when both exploration and exploitation incur cost and the total possible cost is constrained[117]. In knapsack bandits, in every iteration, a set of bandit arms are selected that satisfy the knapsack constraints to maximize the value of arms selected in the knapsack. Our problem differs from the previous formulation of knapsack bandits. In the earlier studies, arms are scheduled independently, but in our formulation, a set of arms are scheduled at a time and we use bandits to only estimate the value of arms (search terms)

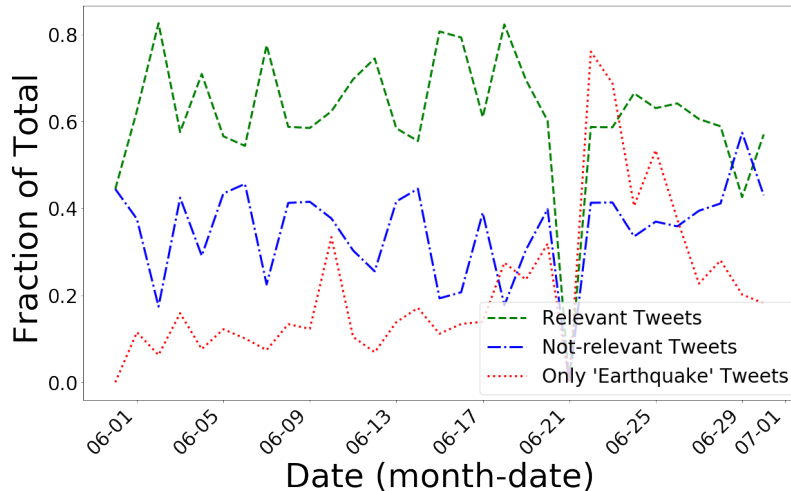


Figure A.6: Relevant vs non-relevant trend of data obtained using our proposed approaches. The relevancy and non-relevancy is determined using a classifier based on a labeled dataset for earthquakes from a prior research.

over multiple iterations. In the domain of focused (web) crawling, reinforcement learning has been studied before e.g. to design a web spider[101]. However, to the best of our knowledge, no one has applied our formulation of knapsack-bandits in the context of Twitter data collection earlier.

A.8 Conclusion and Future Work

In this research, we first show that Twitter limits the amount of data that can be retrieved using their Streaming API to around 4 million tweets in a day. We then propose two novel approaches that respect the constraints on data collection volume and the number of search terms, still get additional data. Given a value function that can quantify the utility of a tweet, our proposed algorithm allows embedding the function in the data collection process itself. Our solution uses the ‘search terms’ as bandit-arms to find the best set of arms that satisfies the constraints. Using ‘earthquake’ related data collection as an example, we estimate that our suggested approach gets data that is more than twice the data retrieved by just searching for ‘earthquake’. In the future, we would like to extend our approach to a distributed system so that multiple machines/processes can coordinate to get more useful data.

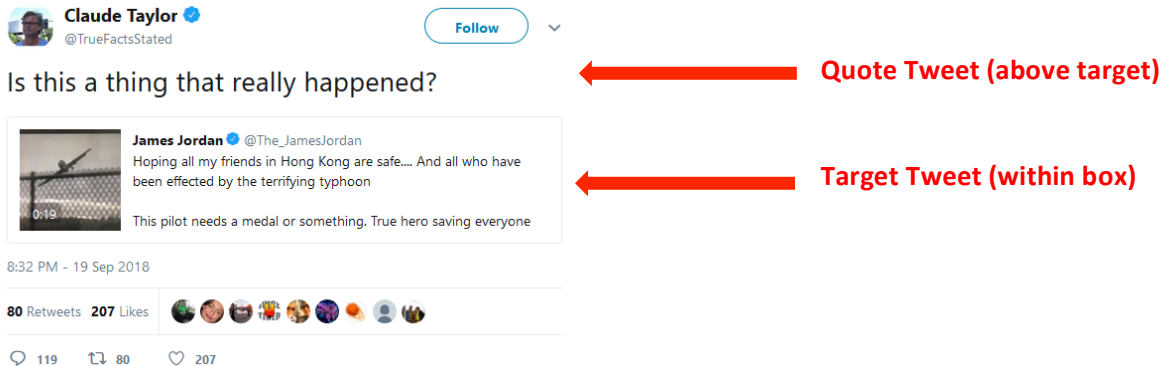
Appendix B

A Users Guide for Labeling Stance in Conversations

CASOS Tweet Labeling Tutorial v2

In this task you will be shown pairs of tweets. These pairs are made up of either a quote tweet and its target tweet or a reply tweet and its target tweet.

Here is what a **quote** pair looks like:

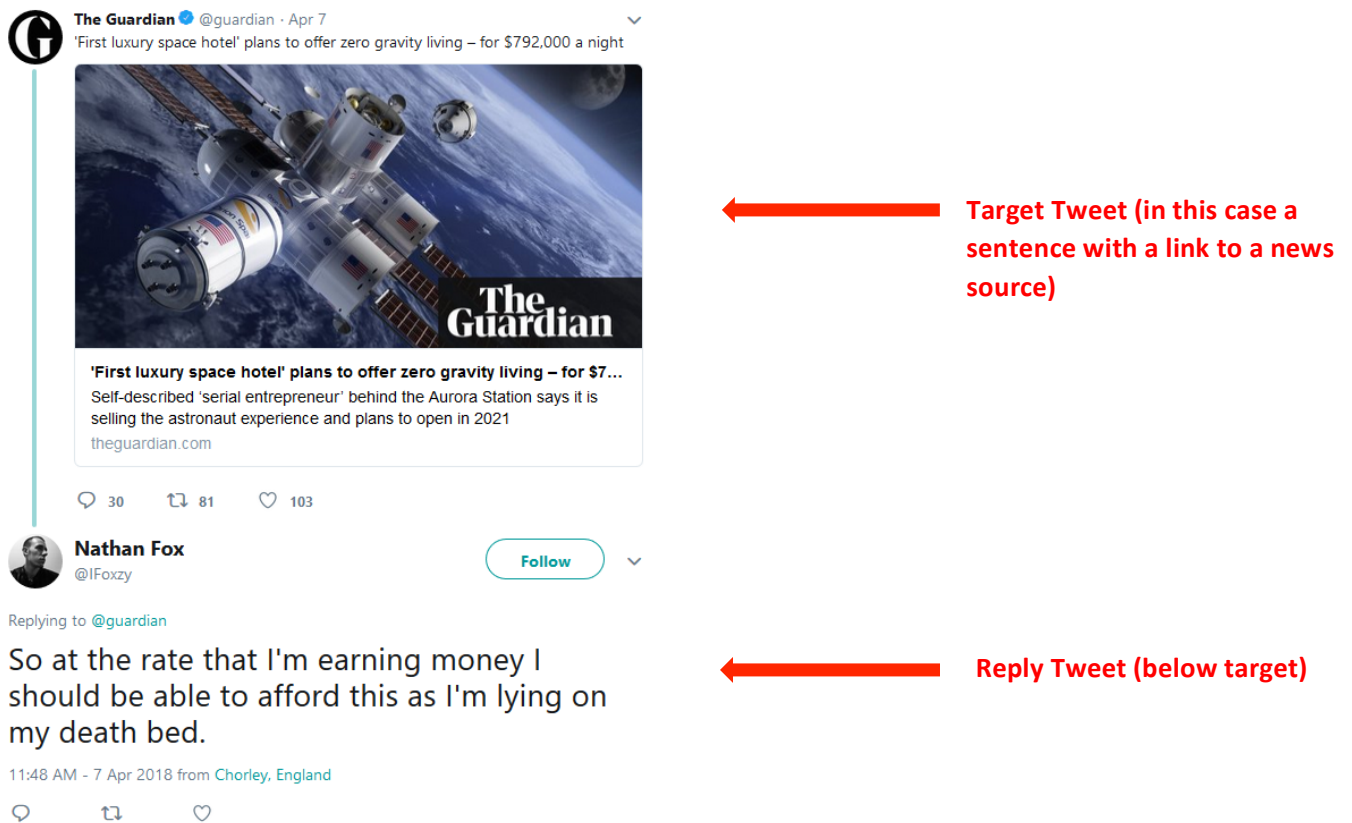


The screenshot shows a tweet by Claude Taylor (@TrueFactsStated) with the text "Is this a thing that really happened?". Below the text is a quote tweet box containing a tweet by James Jordan (@The_JamesJordan) with a video thumbnail and the text "Hoping all my friends in Hong Kong are safe... And all who have been effected by the terrifying typhoon. This pilot needs a medal or something. True hero saving everyone". Red arrows point from the labels "Quote Tweet (above target)" and "Target Tweet (within box)" to the respective parts of the tweet.

Quote Tweet (above target)

Target Tweet (within box)

Here is what a **reply** pair looks like:



The screenshot shows a tweet by The Guardian (@guardian) with the text "'First luxury space hotel' plans to offer zero gravity living – for \$792,000 a night". Below the text is a video thumbnail showing a space station in orbit over Earth. Below the video is a link to the Guardian website. Below the video is a reply tweet by Nathan Fox (@IFozzy) with the text "So at the rate that I'm earning money I should be able to afford this as I'm lying on my death bed." Red arrows point from the labels "Target Tweet (in this case a sentence with a link to a news source)" and "Reply Tweet (below target)" to the respective parts of the tweet.

Target Tweet (in this case a sentence with a link to a news source)

Reply Tweet (below target)

For each pair of tweets the task requires you to judge whether the quote/reply is **denying** that what is being said in the target tweet is true or is **supporting** that what is being said in the target tweet is true.

The response scale looks like this:

Explicitly Supports – Implicitly Supports – Neutral – Implicitly Denies – Explicitly Denies

Explicitly Supports means that the quote/tweet outright states that what the target tweets says is true.

Implicitly Supports means that the quote/tweet implies that the tweeter believes that what the target tweet says is true.

Neutral (/comment/Query) means that the quote/reply **does not support or deny** that what the target tweet says is true. This can be because the quote/reply talks about a different topic than the target tweet or because the quote/reply **talks about some separate issue** within the same topic. This can also be because the quote/reply is **asking** for more information about the target tweet in a way that indicates the quoter/replier has not judged the veracity of the target tweet yet. If the target tweet is reporting on comments made by a third party and the quote/reply is denying or supporting those comments but not the report of those comments, then the relationship should be labeled **neutral**.

Implicitly Denies means that the quote/tweet implies that the tweeter believes that what the target tweet says is false.

Explicitly Denies means that the quote/tweet outright states that what the target tweets says is false.

It should be noted that many of the tweet pairs deal with politically and emotionally contentious topics. Your task is not to determine whether a tweet is true or false, or if a response is wrong or right. Your task is only to determine the relationship between the two tweets in each pair (whether the quote/reply is disagreeing or agreeing that what the target tweet says is true).

For some pairs of tweets, this will be easy. For other pairs, it can be difficult. Please take the time to make a careful judgement.

Following are five examples:

Example 1:

Explanation 1: In this tweet pair, the reply (bottom tweet) is directly stating that the target tweet (top tweet) is lying. The correct choice is “Explicitly Denying”.

Example 2:

Explanation 2: In this tweet pair, the quote (top tweet) is saying that what the target tweet (in the box) says is true. The correct choice is “Explicitly Supporting”.

Example 3:

Claude Taylor [@TrueFactsStated](#) [Follow](#)

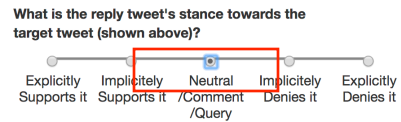
Is this a thing that really happened?

James Jordan [@The_JamesJordan](#)
Hoping all my friends in Hong Kong are safe... And all who have been effected by the terrifying typhoon
0:19 This pilot needs a medal or something. True hero saving everyone

8:32 PM - 19 Sep 2018

80 Retweets 207 Likes

119 80 207



Explanation 3: In this tweet pair, the quote (top tweet) is asking whether what the target tweet (in the box) says happened really happened. The correct choice is “Neutral”. If the quote had been phrased more sarcastically, it could have been labeled as “Implicitly Denying”.

Example 4:

The Guardian [@guardian](#) · Apr 7
'First luxury space hotel' plans to offer zero gravity living – for \$792,000 a night

The Guardian

'First luxury space hotel' plans to offer zero gravity living – for \$7...
Self-described 'serial entrepreneur' behind the Aurora Station says it is selling the astronaut experience and plans to open in 2021
theguardian.com

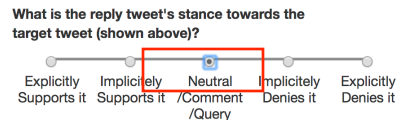
30 81 103

Nathan Fox [@IFoxy](#) [Follow](#)

Replying to [@guardian](#)

So at the rate that I'm earning money I should be able to afford this as I'm lying on my death bed.

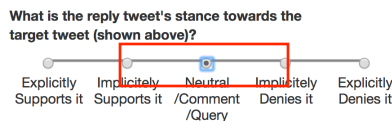
11:48 AM - 7 Apr 2018 from Chorley, England



Explanation 4: In this tweet pair, the reply (bottom tweet) is making a statement regarding the content of the target tweet (top tweet) but is not directly supporting or denying the content of the target tweet. The correct choice is “Neutral”.

Example 5:

The screenshot shows a tweet from Scott Mortrude (@samortrude) with a 'Follow' button. The tweet text is "And your proof of this bullshit would be what, NRA?". Below it is a quote from HuffPost (@HuffPost) that reads: "Gun-hating billionaires and Hollywood elites are manipulating and exploiting children," the gun group said of the Saturday march. [huffp.st/398Po/b](https://huffpost.com/398Po/b). The tweet is dated 12:04 PM - 25 Mar 2018 and has 1 Like. At the bottom, there are icons for reply, retweet, and like.



Explanation 5: In this tweet pair, the quote (top tweet) is making a statement regarding the content of the news report in the target tweet (in the box). The quote is not calling into question the veracity of the news report (whether the NRA made the statement or not), it is calling into question the veracity of the statement that the NRA made. The correct choice is therefore “**Neutral**”. This is an example of the difficulty of determining the relationship between tweets when the target is a news source. If the target is a news source, then care should be taken to decide if the quote/reply is questioning whether what was reported happened or if the quote/reply is accepting the report and questioning the statements of those in the report (as in this example).