UNIVERSITY OF CALIFORNIA

Los Angeles

Ideology Analysis for Social-Media Users via Multi-Modal Data Mining

A dissertation submitted in partial satisfaction
of the requirements for the degree
Doctor of Philosophy in Computer Science

by

Zhiping Xiao

2024

ABSTRACT OF THE DISSERTATION

Ideology Analysis for Social-Media Users via Multi-Modal Data Mining

by

Zhiping Xiao
Doctor of Philosophy in Computer Science
University of California, Los Angeles, 2024
Professor Yizhou Sun, Co-Chair
Professor Mason A. Porter, Co-Chair

Analyzing public opinions on political affairs never fails to attract researchers' attention. One popular application is analyzing ideologies of the ordinary citizens. Traditionally, researchers collect public opinions by conducting surveys, interviews, or estimate via the roll call data. It took a lot of time to come to a conclusion, and it was also hard to find representative objects and keep the objects' opinions unaffected by researchers who design the survey questions. By using social media data, some of the previously-mentioned problems are mitigated due to the unprecedented massive scale. However, we need to keep in mind that social media data might be systematically biased in some other ways. For instance, those who never use social media could not be included.

In recent years, social media, such as Twitter, plays an increasingly important role in people's life. People express opinions, digest information, and interact on social medias. All these behaviors left massive amount of observable clues online, which we collect as our data. Centered around the problems on political ideology analysis, we start from collecting a list of politicians who have verified accounts on Twitter. Then we build our data sets from the Twitter accounts who are not further than one hop away from the politicians, directly following or are followed by the politicians. Although it becomes much easier to collect massive amount of data in an efficient and timely manner, social media data bring us unique challenges. For example: (1) we need to deal with the data size which is typically large; (2) there are seldom ground-truth knowledge on social media data, which leads to the lack of labels; (3) we need to consider how to deal with the multi-modality nature of our data.

If we view each user account as a node, their interactive behaviors could be modeled as links in between. Since they interact in multiple ways, the graph structure we form is heterogeneous. If we view each account as an individual information source, we could represent its feature by the collection of tweets it posted in the past. If we consider temporal information as well, the whole system could also be regarded as a multi-agent dynamic system. Based on the observation of the data, we have proposed the following research problems to answer:

(1) Can we rely solely on the user behavior data, represented as heterogeneous types of links in the graph structure, to reveal the users' ideologies?

(2) Can we rely solely on the text information from tweets, to reveal the users' political polarities?

(3) By learning from the historical data on social media, containing text, link, and temporal information, can we predict the future trend?

To answer the first research problem, we proposed a model that successfully uses the heterogeneous types of relations, called **TIMME** (Twitter Ideology-detection via Multi-task Multi-relational Embedding). Challenges come from (1) the extreme sparsity of the labels, (2) the incompleteness of the input features, and (3) the heterogeneous types of links. **TIMME** is overall better than the state-of-the-art models for ideology detection on Twitter. In theory, it could be extended to other data sets, and could be apply to tasks other than ideology detection.

The work we've done to answer the second problem resulted in an embedding approach called **PEM** (i.e., Polarity-Aware Embeddings Using Multi-Task Learning). The ideological divisions in the United States have become increasingly prominent in daily communication, and a lot of research has been conducted. We propose to quantify political polarity in social-media text data using a polarity-aware method of learning word representations. By learning a word embedding with an explicit polarity dimension, one can infer the polarity of a post and therefore of the social-media account that produced it. Decomposing a traditional embedding into a polarity-neutral semantic component and a polarity-aware component is a major challenge. Very sparse labels is another key challenge. Our experimental results demonstrate that our model can successfully learn high-quality polarity-aware embeddings.

The third research question is answered by our next project, a social dynamic system model that captures the updating patterns in real-world social network data sets. We faced challenges on both the data end and the model end. From the data set, there is no existing publicly-available data sets on real-world social network observations. From the model perspective, modeling continuous time is trickier than modeling discrete time by nature. After decided that we should consider using following the Neural-ODE framework, other challenges came about, such as the data size, and the selection of an appropriate decoder task. Our experimental results show that the framework we propose is capable of learning the dynamic changes in the social network data sets. And our framework could also be used to verify how well an opinion dynamic model captures the changes of a specific real-world data set, under a given task.

The dissertation of Zhiping Xiao is approved.

University of California, Los Angeles

2024

To my advisor Yizhou Sun and my co-advisor Mason A. Porter,

who are my guiding lights in my research journey;

To my mother and father,

who has always been my role model and support;

To my grandpa and other family members,

you are the fundamental motivation behind all my actions.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

Our first work in this thesis is supported by NSF III-1705169, NSF CAREER Award 1741634, NSF #1937599, DARPA HR00112090027, Okawa Foundation Grant, and Amazon Research Award.

Our second work in this thesis is supported by the National Science Foundation (through grants III-1705169, NSF 1937599, NSF 2119643, and 1922952), an Okawa Foundation Grant, Amazon Research Awards, Cisco research grant USA000EP280889, Picsart Gifts, and Snapchat Gifts.

Our third work in this thesis is supported by the National Science Foundation (NSF 2303037 and NSF 2312501), and NASA Grant (NASA M2300680).

CURRICULUM VITAE

2012 – 2016	B.S. in Computer Science, Peking University, Beijing, China.
2016 – 2018	M.S. in Computer Science, University of California at Berkeley (UCB).

SELECTED PUBLICATIONS

[1] Zhiping Xiao, a Jeffrey Zhu, Yining Wang, Pei Zhou, Wen Hong Lam, Mason A. Porter, and Yizhou Sun, "Detecting political biases of named entities and hashtags on Twitter," European Physical Journal (EPJ) — Data Science **12**, 1, 20 (2023) .

[2] Zhiping Xiao, Weiping Song, Haoyan Xu, Zhicheng Ren, and Yizhou Sun, "TIMME: Twitter Ideology-Detection via Multi-Task Multi-Relational Embedding," (2020), In Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)

CHAPTER 1

Introduction

In this chapter, we show the background of our research works, we start from discussing our motivation, and include the three research problems we study.

1.1 Motivation

Online social network platforms are platforms where people register accounts to become their users, generate content such as text or images, and interact with each other. These social networking service, or social media, provide a wide variety of data to be studied.

Those real-world data are meaningful to us if we are interested in public opinions [26, 32, 69, 71, 74]. They provide clues we can utilize for analyzing the public and each individual [3, 7, 26, 32, 59, 69, 71, 74, 79, 95, 122].

Unlike the traditional survey-based methods of collecting people's opinions, which were usually expensive and not efficient enough [3, 118], social media provide an opportunity of analyzing people's opinions easily, at a large scale, and in a timely manner. People express their opinions online, and are influenced by each other. While social opinions and social activities are becoming more and more indivisible from the online world, political-related affairs are not an exception. For example, some social movements are initially started online.

Among all the social networking service platforms, Twitter [1] is especially outstanding by the amount of active accounts, as well as the developer-friendly APIs they once provided

[1] https://twitter.com/

us in the past few years. [2] By using their APIs, we could easily and legally access a huge amount of data. Further more, due to the huge popularity on Twitter, most of the politicians (i.e., the congress members [3], the presidents in recent years, and the presidents' cabinets), have their verified accounts on Twitter. Those politicians' accounts could be used as as high-quality source of labeled data. The only downside is that they only take a small portion of the whole data set we have. Although our labels are of high quality in general, we still suffer from the scarcity of the ground-truth labels.

Our goal is to make good use of the multi-modality social media data to analyze the political tendencies of the accounts in social networks.

There are various ways of viewing the Twitter data structure. For example: if we view each account as a node, their interactive behaviors could be modeled as heterogeneous types of links connecting two nodes together [33, 59]. If we view each account as an individual information source, we focus on the collection of tweets it posted, and conduct studies on the text content. If we consider temporal information as well, the entire system could also be regarded as a multi-agent dynamic system. Based on the observations of the data, we propose to solve the following research problems:

(1) Can we rely solely on the account interaction data, represented as heterogeneous types of links in the graph structure, to reveal the accounts' ideologies?

(2) Can we rely solely on the text information from tweets, to reveal the accounts' political polarities?

(3) By learning from the historical data on social media, containing text, link, and temporal information, can we predict the future trend?

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

[3] https://www.congress.gov/congressional-record

1.2 Research Problems

In this thesis, we introduce two published research works and one ongoing research work.

The first two works, corresponding to the first two research problems we proposed, are focusing on different aspects of the social network data: links and text information, respectively. *TIMME: Twitter Ideology-detection via Multi-task Multi-relational Embedding* (Chapter 2) models the the relations among the accounts; *Constructing Polarity-Aware Embeddings using Multi-Task Learning* (Chapter 3) studies the text content posted as tweets.

The social-dynamical-system project aims to solve the third research problem. We propose to utilize our data, methods, and results from the first two projects to help us handle the heterogeneous graph data, and text information better. In this project, we propose to predict accounts' future posts and interactions on a social-media platform using their past behavior data.

The timeline of the projects involved are roughly listed in the Table 1.1:

Table 1.1: The Timeline of the Projects

Project	Years We Worked on	Other Contributors
TIMME	2018,2019	Weiping Song, Haoyan Xu, Zhicheng Ren, Haoran Wang, Zhiwen Hu, Prof. Yizhou Sun
PEM	2019,2020,2021	Pei Zhou, Jeffrey Zhu, Yining Wang, Wen Hong Lam, Prof. Mason A. Porter, Prof. Yizhou Sun
LSDS	2021, 2022, 2023, 2024	Zijie Huang, Xiao Luo, Yifang Qin, Xinyu Wang, Prof. Mason A. Porter, Prof. Yizhou Sun

1.2.1 Can Ideology be Inferred from Interactions?

The first project TIMME [158] was conducted to study whether or not we can infer an account's ideology from its interaction with other accounts. Motivated by the scarcity and noisiness of the real-world social network data sets, it proposed a novel Multi-Task GNN-based approach to model heterogeneous graphs.

Inspired by previous works [59], we proposed to reveal the ideology of ordinary people by utilizing the multiple types of interactions among them, and by using a very limited amount of political-party labels. There, we model ideology-detection of Twitter accounts as political-tendency classification of nodes in a heterogeneous graph.

Multiple tasks are involved in the decoder part of the model: one classification task, and $|\mathcal{R}| = 5$ link-prediction tasks. We get the tweets from the politicians' and some of their one-hop neighbors' accounts, and extract the interactions among them, such as *retweet*, *reply*, *mention*, and *like*. The multi-task setting makes the model more stable, and lets each task help each other to achieve higher performances togetherly.

Designed specifically for real-world social network data, we focused on solving the scarcity of labels, and handling the incompleteness of data more carefully. We also explored the possibility of using text embedding as nodes' features to further improve the performance. However, our conclusion is, a low-quality text embedding is much worse than one-hot embedding or random embedding. Such low-quality text embeddings include but are not limited to: taking the mean value of GloVe embeddings of all tokens in a sentence as sentence-level embeddings. As a matter of fact, previous studies have shown that in many scenarios, a large language model that is not specifically designed for learning sentence-level embeddings, will not be significantly better than the naive models such as the GloVe baseline [125].

We have showed that inferring ideology of social network nodes from their links/relations is not only doable, but also much easier and much more reliable than inferring their ideologies from their tweet contents.

This work was published in the conference KDD (i.e., Knowledge Discovery in Database) year 2020 proceedings [158].

1.2.2 Can Polarity be Explained by Text?

The second project PEM (see Chapter 3), is to study whether or not we can provide explanations on the accounts' political polarity, using only their text. We proposed an effective novel approach to reveal political-polarities from tweet text, emphasizing the usefulness of text content. We found words and phrases naturally carrying political-polarity information, but those polarities are not reflected in today's word embeddings. We thereby learn an embedding that captures both the semantic meaning and the political polarity of words and phrases. Inspired by GN-Glove [172] which reveal gender polarity by learning a two-component embedding, we propose to learn a two-component embedding that reveals political polarity: semantic component and polarity component.

However, in gender bias studies, there exists many ground-truth counter pairs such as male and female, actor and actress, king and queen, and so on. And those ground-truth pairs exist for decades and seldom change their meanings from the gender perspective. Different from the gender bias cases, political polarity is ever changing in an extremely high frequency.

For example, at some point, certain groups of people use the hashtag **#AllLivesMatter** to show that they have something different to say about the **#BlackLivesMatter** movement. But after a few years we see **#AllLivesMatter** used together with **#Pro-Life**, meaning that they are against abortion. Then in some other case studies on the tweet content, we see **#AllLivesMatter** and **#Pro-Life** are used together again to call for animal right protection.

At an early stage of the project, we have tried to figure out some reliable ground-truth counter pairs of words from the politicians' tweets. Later on, inspired by some adversarial frameworks, such as Cycle-GAN [177], we decided to solve this problem by designing an adversarial framework for training, using the two sets of tweets from the democratic politicians and republican politicians respectively.

In this project there involves three tasks when learning such embeddings: (1) preserving the context information (2) using the polarity component to predict polarity (3) enforcing the semantic component and the polarity component to be as independent from each other as possible.

This work was published in the EPJ (i.e. European Physical Journal) – Data Science, in year 2023 [159].

1.2.3 Can Future Behaviors be Predicted using the Past Records?

In the work to be submitted soon, we are examining the third proposed research problem by studying changes in accounts' opinions with time. To do this, we are constructing a social dynamical systems to model opinion trajectories on a social network (see Chapter 4).

The existing works on analyzing how do a group of people influence each other and change their opinions over time, are mostly tested on simulated data, under a lot of constraints. Most of the existing opinion dynamic model studies focus more on the mathematical features of these models and the heuristics behind their design decisions. On the other hand, there are many neural network models targeting at modeling sequential data, but they are not specifically designed for real-world social network data analysis. Therefore, seldom do they consider the unique challenges of such applications, such as the noisiness and sparseness of the data. Besides, most of them are designed to handle observations in discrete time steps, while we propose to study continuous-time data.

In this project, we model the dynamical changes using an ordinary-differential-equation (ODE) solver, which enables the system to predict future observation at any continuous time, instead of using discrete time steps. Inspired by Neural-ODE [25], LG-ODE [67], and NRI [77], we implement this design by adopting a variational-autoencoder (VAE) framework. In terms of the VAE components, we use a temporal GNN model as its encoder to generate hidden representations at the starting time ($t = 0$), and we write down the update rules in a neural-network format our ODE function. We can use our decoder for several different downstream tasks, such as ideology prediction, hidden-representation reconstruction, and interaction prediction. And it could potentially be applied to many other different tasks, as long as reasonable training objective is provided.

We plan to submit this work in 2024.

1.2.4 Potential Future Direction

One of the most interesting yet challenging direction, is to study whether or not we can infer causality relation from the observations online.

This future plan is motivated by the previous projects. We are interested in further interpret the interventions and causal inferences in the social networking system. Even though we can model the links or relations, the node features or text content, and their dynamic changes properly, it still remains unclear what outcomes each factor causes (see Chapter 5.2). The social dynamical system project might be helpful in helping us overcome the lack-of-counter-factual-label problem.

CHAPTER 2

Twitter Ideology-detection via Multi-task Multi-relational Embedding

In this work, [1] we study whether or not we can infer an account's ideology from its interaction with other accounts. We estimate it by using Twitter data, and formalize it as a binary classification problem. Ideology-detection has long been a challenging yet important problem. Certain groups, such as the policy makers, rely on it to make wise decisions. Back in the old days when labor-intensive survey-studies were needed to collect public opinions, analyzing ordinary citizens' political tendencies was uneasy. The rise of social medias, such as Twitter, has enabled us to gather ordinary people's data easily. However, the incompleteness of the labels and the features in social network data sets is tricky, not to mention the enormous data size and the heterogeneousity. The data differ dramatically from many commonly-used data sets, thus brings unique challenges. In our work, first we built our own data sets from Twitter. Next, we proposed **TIMME**, a multi-task multi-relational embedding model, that works efficiently on sparsely-labeled heterogeneous real-world data set. It can also handle the incompleteness of the input features. Experimental results showed that **TIMME** is overall better than the state-of-the-art models for ideology detection on Twitter. Our findings include: edges can lead to good classification outcomes without text; conservative voice is under-represented on Twitter; follow is the most important relation to predict ideology; retweet and mention enhance a higher chance of like, etc. Last but not least, **TIMME** can be extended to other data sets and tasks in theory.

[1] This work was published in the conference KDD (i.e., Knowledge Discovery in Database) year 2020 proceedings [158]. Zhiping Xiao lead the discussions, designed the model under supervision of Prof. Yizhou Sun, and was in charge of most of the coding and writing.

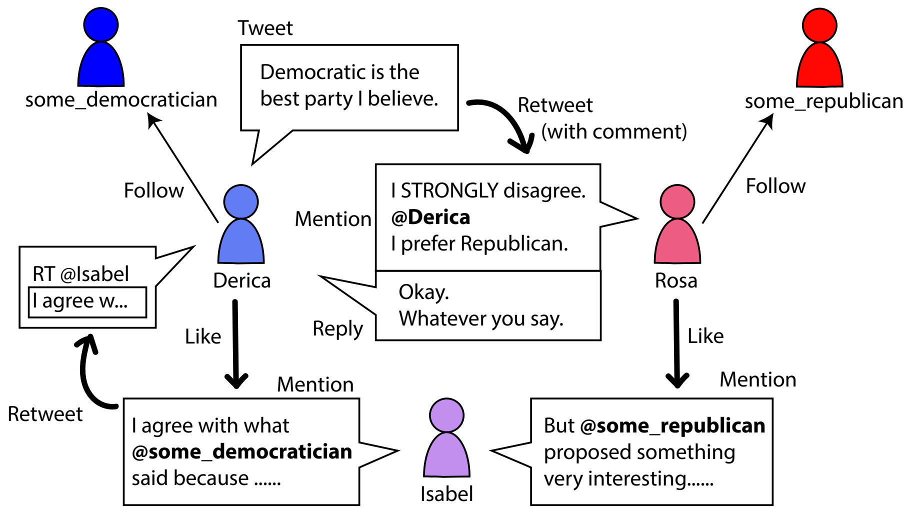

some_democratician
Tweet
Democratic is the best party I believe.
Retweet (with comment)
some_republican
Follow
Follow
Mention
I STRONGLY disagree. @Derica I prefer Republican.
Derica
Rosa
RT @Isabel
I agree w...
Okay. Whatever you say.
Reply
Like
Like
Mention
Mention
Retweet
I agree with what @some_democratician said because ......
Isabel
But @some_republican proposed something very interesting......

Figure 2.1: An example of different relation types on Twitter. Derica is on liberal (left) side while Rosa is on the conservative (right) side. Isabel does not have significant tendency.

2.1 Introduction

Studies on ideology are generally attracting more attention as the election year approaches. Ideology here refers to the political stance or tendency of people, often reflected as left- or right-leaning. Measuring the politicians' ideology helps predict some important decisions' final outcomes, but it does not provide more insights into ordinary citizens' views, which are also of decisive significance. Decades ago, social scientists have already started using probabilistic models to study the voting behaviors of the politicians[3, 118]. But seldom did they study the mass population's opinions, for the survey-based study is extremely labor-intensive and hard-to-scale. The booming development of social networks in the recent years shed light on detecting ordinary people's ideology. In social networks, people are more relaxed than in an offline interview, and behave naturally. Social networks, in return, has shaped people's habits, giving rise to opinion leaders, encouraging youngsters' political involvement [114].

Most existing approaches of ideology detection on social networks focus on text [26, 32, 69, 71, 74]. Most of their methodologies based on probabilistic models, following the long-lasting tradition started by social scientists. Some others [7, 59, 74, 122] noticed the advantages of neural networks, but seldom do they focus on edges. We will show that the social-network edges' contribution to ideology detection has been under-estimated.

An intuitive explanation of how edges could be telling is illustrated in Figure 2.1. Different types of edges come into being for different reasons. We have five relation types between accounts on Twitter today: follow, retweet, reply, mention, like, and the relations affect each other. For instance, after Rosa retweet from Derica and mention her, Derica reply to her; when Isabel mention some politicians in her posts, the politician's followers might come to interact with her. One might mention or reply to debate, but like always stands for agreement. The relations could reflect some opinions that a user would never tell you verbally. Words could be easily disguised, and there is always a problem called "the silent majority", for most people are unwilling to express.

Yet there are some uniqueness of Twitter data set, bringing about many challenges. It is especially the case when existing approaches are mostly dealing with smaller data sets with much sparser edges than ours, such as academic graphs, text-word graphs, and knowledge-graphs. First, our Twitter data set is large and the edges are relatively dense (Section 2.4). Some models such as GraphSAGE [62] will be super slow sampling our graph. Second, labels are extremely sparse, less than 1%. Most approaches will suffer from severe over-fitting, and the lack of reliable evaluation. Third, features are always incomplete, for in real-life data sets like Twitter, many accounts are removed or blocked. Fourth, modeling the heterogeneity is nontrivial. Many existing methods designed for homogeneous networks tend to ignore the information brought by the types of edges.

Existing works can not address the above challenges well. Even though some realized the importance of edges [33, 59], they failed to provide an embedding. Most people learn an embedding by separating the heterogeneous graph into different homogeneous views entirely, and combine them in the very end.

We propose to solve the above-listed problems by **TIMME** (Twitter Ideology-detection via Multi-task Multi-relational Embedding), a model good at handling sparsely-labeled large graph, utilizing multiple relation types, and optionally dealing with missing features. Our code with data is released on Github at https://github.com/PatriciaXiao/TIMME. Our major contributions are:

- We propose **TIMME** for ideology detection on Twitter, whose encoder captures the interactions between different relations, and decoder treats different relations separately while measuring the importance of each relation to ideology detection.
- The experimental results have proved that **TIMME** outperforms the state-of-the-art models. Case studies showed that conservative voice is typically under-represented on Twitter. There are also many findings on the relations' interactions.
- The large-scale data set we crawled, cleaned, and labeled (Appendix 2.7.1) provides a new benchmark to study heterogeneous information networks.

In this paper, we will walk through the related work in Section 2.2, introduce the preliminaries and the definition of the problem we are working on in Section 2.3, followed by the details of the model we propose in Section 2.4, experimental results and discussions in Section 2.5, and Section 2.6 for conclusion.

2.2 Related Work

In this section, we discuss our related works from three perspectives. They are, ideology detection, graph neural networks, and multi-task learning.

2.2.1 Ideology Detection

Ideology detection in general could be naturally divided into two directions, based on the targets to predict: of the politicians [30, 107, 119], and of the ordinary people [3, 7, 26, 32, 59, 69, 71, 74, 79, 95, 122]. The work conducted on ordinary people could also be categorized into two types according to the source of data being used: intentionally collected via strategies like survey [3, 79], and directly collected such as from news articles [7] or from social networks [59, 69, 74]. Some studies take advantages from both sides, asking self-reported responses from a group of users selected from social networks [122], and some researchers admitted the limitations of survey experiments [95]. Emerging from social science, probabilistic models have been widely used for such kinds of analysis since the early 1980s [7, 59, 119]. On the other hand, on social network data sets, it is quite intuitive trying to extract information from text data to do ideology-detection [26, 32, 69, 71, 74], only a few paid attention to edges [33, 59]. Our work differs from them all, since: (1) unlike probabilistic models, we use GNN approaches to solve this problem, so that we take advantage of the high-efficient computational resources, and we have the embeddings for further analysis; (2) we focus on **relations** between users, and proved how telling those relations are.

2.2.2 Graph Neural Networks (GNNs)

In this subsection, we summarize a few types of GNNs.

2.2.2.1 Graph Convolutional Networks (GCNs)

Inspired by the great success of convolutional neural networks (CNNs), researchers have been seeking for its extension onto information networks [39, 78] to learn the entities' embeddings. The Graph Convolutional Networks (GCNs) [78] could be regarded as an approximation of spectral-domain convolution of the graph signals. A deeper insight [85] shows that the key reason why GCN works so well on classification tasks is that its operation is a form of Laplacian smoothing, and concludes the potential over-smoothing problem, as well as emphasizes the harm of the lack of labels.

GCN convolutional operation could also be viewed as sampling and aggregating of the neighborhood information, such as GraphSAGE [62] and FastGCN [23], enabling training in batches. To improve GraphSAGE's expressiveness, GIN [160] is developed, enabling more complex forms of aggregation. In practice, due to the sampling time cost brought by our edges' high density, GIN, GraphSAGE and its extension onto heterogeneous information network such as HetGNN [166] and GATNE [20] are not very suitable on our data sets.

The RELATIONAL-GCN (R-GCN) model [132] extends GCN onto heterogeneous information networks. A very large number of relation-types $|\mathcal{R}|$ ends up in overwhelming parameters, thus they put some constraints on the weight matrices, referred to as weight-matrix decomposition. GEM [91] is almost a special case of R-GCN. Unfortunately, their code is kept confidential. According to the descriptions in their paper, they have a component of similar use as the attention weights α in our encoder, but it is treated as a free parameter.

Another way of dealing with multiple edge types is well-represented by SHINE [152], who treats the heterogeneous types of edges as separated homogeneous edges, and combines embeddings from all relations in the end. SHINE did not make good use of the multiple relations to its full potential, modeling the relations without allowing complex interactions among them. GTN [164] is similar with SHINE in splitting the graph into separate views and combining the output at the very end. Besides, GTN uses meta-path, thus is potentially more expressive than SHINE, but would rely heavily on the quality and quantity of the meta-paths being used.

2.2.2.2 Graph Attention Networks (GATs)

Graph Attention Networks (GATs) [149] are another nontrivial direction to go under the topic of graph neural networks. It incorporates attention into propagation by applying self-attention on the neighbors. Multi-head mechanism is typically good at capturing more aspects of the features, thus learn embeddings more effectively [89, 100].

An extension of GAT on heterogeneous information networks is the Heterogeneous Graph Attention Network (HAN) [154]. Beside inheriting the node-level attention from GAT, it considers different relation types by sampling its neighbors from different meta-paths. It first conducts type-specific transformation and compute the importance of neighbors of each node. After that, it aggregates the coefficients of all neighbor nodes to update the current node's representation. In addition, to obtain more comprehensive information, it conducts semantic-level attention, which takes the result of node-level attention as input and computes the importance of each meta-path. We use HAN as an important baseline in our experiments.

2.2.3 Multi-Task Learning (MTL)

In multi-task learning (MTL) settings, there are multiple tasks sharing the same inductive bias jointly trained. Ideally, the performance of every task should benefit from leveraging auxiliary knowledge from each other. As is concluded in [127], MTL could be applied with or without neural network structure. On neural network structure, the most common approach is to do hard parameter-sharing, where the tasks share some hidden layers. The most common way of optimizing an MTL problem is to solve it by joint-training fashion, with joint loss computed as a weighted combination of losses from different tasks [75]. It has a very wide range of applications, such as the DMT-demographic models [151] where multiple aspects of Twitter data (e.g. text, images) are fed into different tasks and trained jointly. Aron and Nirmal et al. [34] also applied MTL on Twitter, separating the tasks by user categories. Our multi-task design differs from theirs, and we treat node classification and edge prediction on different relation types as different tasks.

2.3 Problem Definition

Our goal is to predict Twitter accounts' ideologies by learning the ideological embedding of accounts in a political-centered social network.

Definition 2.3.1. (Heterogeneous Information Network) Following previous work [143], we say that an information network $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, where number of vertices is $|\mathcal{V}| = N$, is a **heterogeneous information network** when there are $|\mathcal{T}| = T$ types of vertices, $|\mathcal{R}| = R$ types of edges, and $\max(T, R) > 1$. The heterogeneous graph $\mathcal{G}$ can be represented as $\mathcal{G} = \{\{\mathcal{V}_1, \mathcal{V}_2, \dots \mathcal{V}_T\}, \{\mathcal{E}_1, \mathcal{E}_2, \dots, \mathcal{E}_R\}\}$

Each possible edge from the i^{th} node to the j^{th}, represented as $e_{ij} \in \mathcal{E}$ has a weight value $w_{ij} > 0$ associated to it, where $w_{ij} = 0$ representing $e_{ij} \notin \mathcal{E}$. In our case, $\mathcal{G}$ is a directed graph. In general, we have $\langle v_i, v_j \rangle \not\equiv \langle v_j, v_i \rangle$ and $w_{ij} \not\equiv w_{ji}$.

Our Twitter data $\mathcal{G}_{\text{Twitter}}\{\mathcal{V}, \{\mathcal{E}_1, \mathcal{E}_2, \mathcal{E}_3, \mathcal{E}_4, \mathcal{E}_5\}\}$ contains $T = 1$ type of entities (accounts), and $R = 5$ different types of edges (i.e., relations) among the entities, namely follow, retweet, like, mention, reply.

A detailed description about Twitter data is included in Appendix 2.7.1, and we call the subgraph we selected from Twitter-network a political-centered social network, which is defined as follows:

Definition 2.3.2. (Political-Centered Social Network) The political-centered social network is a special case of directed heterogeneous information network. With a pre-defined politicians set $\mathcal{P} \subseteq \mathcal{V}$, in our selected heterogeneous network $\mathcal{G}_{\text{Twitter}}$, for all $e = \langle v_i, v_j \rangle \in \mathcal{E}_r$ where $r \in \{1, 2, \dots, R\}$, there has to be either $v_i \in \mathcal{P}$ or $v_j \in \mathcal{P}$. All the politicians in this data set have ground-truth labels indicating their political stance. The political-centered social networks are represented as $\mathcal{G}_p$.

We would like to leverage the information we have to learn the representation of the accounts, which can help us reveal their ideologies. Due to the lack of Independent representatives (only two in total), we consider the binary-set labels only: {liberal, conservative}. Based on the ideology tendencies of the two political parties in recent years [43], we label the Democratic accounts as liberal, and Republican as conservative.

Definition 2.3.3. (Multi-task Multi-relational Network Embedding) Given a network $\mathcal{G}_p = \{\mathcal{V}, \{\mathcal{E}_1, \mathcal{E}_2, \mathcal{E}_3, \mathcal{E}_4, \mathcal{E}_5\}\}$ where the number of nodes is $|\mathcal{V}| = N$, the goal of TIMME is to learn such a representation $h_i \in \mathbb{R}^d$ where $d \ll N$ for all $v_i \in \mathcal{V}$, that captures the categorical information of nodes, such as their ideology tendencies. As a measurement, we want the representation $H \in \mathbb{R}^{N \times d}$, to success on both node-classification and edge-prediction.

2.4 Methodology

The general architecture of our proposed model is illustrated in Figure 2.2. It contains two components: encoder and decoder. The encoder contains two multi-relational convolutional layers. The output of the encoder is passed on to the decoder, who handles the downstream tasks.

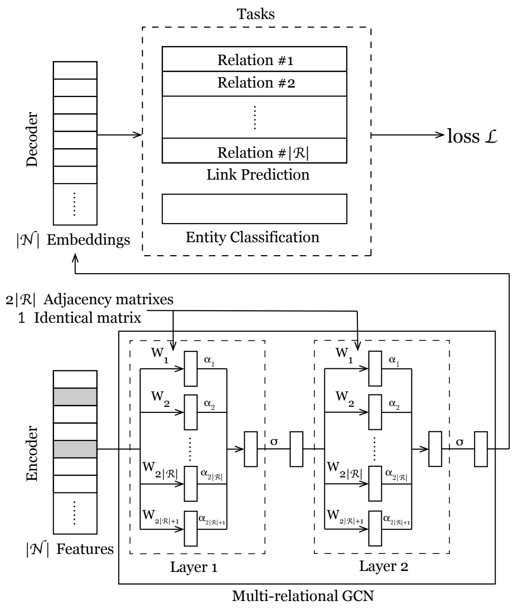


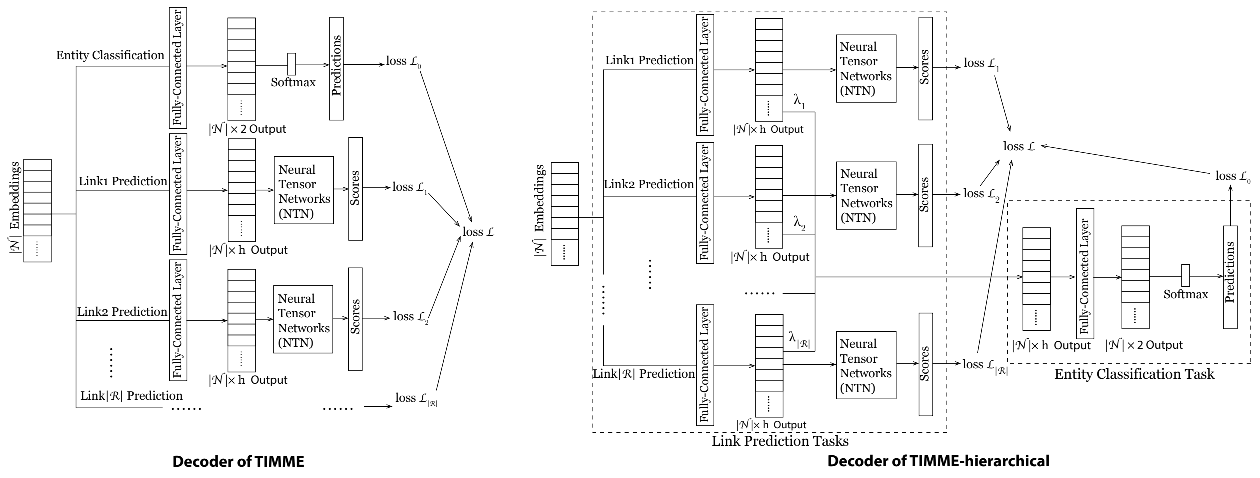
Figure 2.2: The general architecture of our model, with the encoder shown in details. Grey blocks represent missing features. Our model can either handle them by treating them as learnable parameters, or use one-hot features.

2.4.1 Multi-Relation Encoder

As mentioned before in Section 2.1, the challenges faced by the encoder part are the large data scale, the heterogeneous edge types, and the missing features.

GCN is very effective in learning the nodes' embeddings, especially good at classification tasks. Meanwhile, it is also naturally efficient, in terms of handling the large amount of vertices N.

Random-walk-based approaches such as node2vec [57] with time complexity $\mathcal{O}(a^2N)$, where a is the average degree of the graph, suffer from the relatively-high degree in our data set. On the other hand, GCN-based approaches are naturally efficient here. Like is analyzed in Cluster-GCN [28], the time complexity of the standard GCN model is $\mathcal{O}(L\|A\|_0F + LNF^2)$, where L is the number of layers, $\|A\|_0$ the number of non-zeros in the adjacency matrix, F the number of features. Note that the time complexity increases linearly when N increases. A GCN model's layer-wise propagation can be written as



Entity Classification
Fully-Connected Layer
Softmax
Predictions
loss $\mathcal{L}_0$
$|\mathcal{N}| \times 2$ Output
$|\mathcal{N}|$ Embeddings
Link1 Prediction
Neural Tensor Networks (NTN)
Scores
loss $\mathcal{L}_1$
$|\mathcal{N}| \times$ h Output
loss $\mathcal{L}$
Link2 Prediction
loss $\mathcal{L}_2$
Link$|\mathcal{R}|$ Prediction
loss $\mathcal{L}_{|\mathcal{R}|}$
Decoder of TIMME
λ_1
λ_2
$\lambda_{|\mathcal{R}|}$
Link Prediction Tasks
Entity Classification Task
Decoder of TIMME-hierarchical

Figure 2.3: The two types of decoder in our multi-task framework, referred to as **TIMME** and **TIMME-hierarchical**.

$$H^{(l+1)} = \sigma\Big(\hat{A}H^{(l)}W^{(l)}\Big),$$

where $\hat{A} = \tilde{D}^{\frac{1}{2}}(A + I_N)\tilde{D}^{\frac{1}{2}}$, and $\tilde{D}$ is defined as the diagonal matrix and A the adjacency matrix. Each diagonal element D_{ii} of D could also be written as d_i, and it is equal to the sum of all the edges attached to v_i; Matrix $H^{(l)} \in \mathbb{R}^{N\times d^{(l)}}$ is the $d^{(l)}$-dimensional representation of the N nodes at the l^{th} layer; and $W^{(l)} \in \mathbb{R}^{d^{(l)}\times d^{(l+1)}}$ is the weight parameters at layer l. This is similar to the the case of an ordinary MLP (i.e., Multi-layer Perceptron) model.In a certain way, $\hat{A}$ can be viewed as A after being normalized.

We propose to model the heterogeneous types of edges and their interactions in the encoder. Otherwise, if we split the views like many others did, the model will never be expressive enough to capture the interactions among relations. For any given political-centered graph $\mathcal{G}_P$, let's denote the total number of nodes $|\mathcal{V}| = N$, the number of relations $|\mathcal{R}| = R$, the set of nodes $\mathcal{V}$, the set of relations $\mathcal{R}$, and $\mathcal{E}_r$ being the set of edges under relation $r \in \mathcal{R}$. Representation being learned after layer l ($l \in \{1, 2\}$) is represented as $H^{(l)} \in \mathbb{R}^{N\times d^{(l)}}$, and the input features form the matrix $H^{(0)} \in \mathbb{R}^{N\times d^{(0)}}$. The set of relations $\hat{\mathcal{R}}$, where $|\hat{\mathcal{R}}| = 2R + 1$, includes all relations in the original direction (R), the relations in reversed direction (R), plus an identical-matrix relation (1). Our data set has only $|\mathcal{R}| = R = 5$ relation types, so it should be fine not to conduct a weight-matrix decomposition like R-

GCN [132]. We model the layer-wise propagation at layer $l+1$ as

$$H^{(l+1)} = \sigma\Big(\sum_{r\in\hat{\mathcal{R}}} \alpha_r \hat{A}_r H^{(l)} W_r^{(l)}\Big)\,, \tag{2.1}$$

where $H^{(l)} \in \mathbb{R}^{N\times d^{(l)}}$ is used to denote the representation of the nodes after the l^{th} encoder layer, and the initial input feature is $H^{(0)}$. $\hat{A}_r = \tilde{D}_r^{\frac{1}{2}}(A_r + I_N)\tilde{D}_r^{\frac{1}{2}}$ is defined in similar way as $\hat{A}$ in GCN, but it is calculated per relation. The activation function σ we use is ReLU. By default, $\alpha = [\alpha_1, \ldots \alpha_r \ldots]^T \in \mathbb{R}^{2R+1}$ is calculated by scaled dot-product self-attention over the outputs of $H_r^{(l+1)} = \hat{A}_r H^{(l)} W_r^{(l)}$:

$$A = \text{Attention}(Q,K,V) = \text{softmax}\Big(\frac{QK^T}{\sqrt{d}}\Big)V \in \mathbb{R}^{(2R+1)\times d}\,, \tag{2.2}$$

where $Q = K = V \in \mathbb{R}^{(2R+1)\times d}$ comes from the $2R+1$ matrices $H_r^{(l+1)} \in \mathbb{R}^{N\times d}$, stacking up as $O \in \mathbb{R}^{(2R+1)\times N\times d}$, taking the mean value over the N entities. We then calculate an attention to apply to the $2R+1$ outputs as

$$\alpha = \text{softmax}\Big(\text{sum}_{\text{col}}\Big(\frac{QK^T}{\sqrt{d}}\Big)\Big) \in \mathbb{R}^{2R+1}\,, \tag{2.3}$$

where $\text{sum}_{\text{col}}(X)$ takes the sum of each column in $X \in \mathbb{R}^{d_1\times d_2}$ and ends up in a vector $\in \mathbb{R}^{d_2}$.

The last problem to solve is that the initial features $H^{(0)}$ are often incomplete in real life. In most cases, people would go by one-hot features or randomized features. But we want to enable our model to use the real features, even if the real-features are incomplete. Inspired by graph representation learning strategies such as LINE [145], we propose to treat the unknown features as trainable parameters. That is, for a graph $\mathcal{G}_p$ whose vertice set is $\mathcal{V}$, $\mathcal{V}_{featured} \bigcap \mathcal{V}_{featureless} = \varnothing$ and $\mathcal{V}_{featured} \bigcup \mathcal{V}_{featureless} = \mathcal{V}$, for any node with valid feature $\forall v_i \in \mathcal{V}_{featured}$, the node's feature vector $H_i^{(0)}$ is known and fixed. For $\forall v_j \in \mathcal{V}_{featureless}$, the corresponding row vector $H_j^{(0)}$ is unknown and treated as a trainable parameter. The generation of the features will be discussed in the Appendix 2.7.1. In brief, TIMME can handle any missing input feature.

2.4.2 Multi-Task Decoder

We propose **TIMME** as a multi-task learning model such that the sparsity of the labels can be overcome with the help of the edge information. As is shown in Figure 2.3, we propose two architectures of the multi-task decoder. When we test it on a single-task i, we simply disable the remaining losses but a single $\mathcal{L}_i$, and name our model in single-task mode **TIMME-single**.

$\mathcal{L}_0$ is defined the same way as was proposed in [78], in our case a binary cross-entropy loss:

$$\mathcal{L}_0 = -\sum_{y \in Y_{train}} \Big(y \log(y) + (1-y)\log(1-y) \Big) \,, \tag{2.4}$$

where Y_{train} contains the labels in the training set we have.

$\mathcal{L}_1, \ldots \mathcal{L}_R$ are edge-prediction losses, calculated by binary cross-entropy loss between edge-labels and the predicted edge scores' logits. To keep the edges asymmetric, we used Neural Tensor Network (NTN) structure [140], with simplification inspired by DistMult [161]. We set the number of slices be $k = 1$ for $W_r \in \mathbb{R}^{d \times d \times k}$, omitting the linear transformer U, and restricting the weight matrices W_r each being a diagonal matrix. For convenience, we refer to this edge-prediction cell as **TIMME-NTN**. Consider triplet (v_i, r, v_j), and denote the encoder output of $v_i, v_j \in \mathcal{V}$ as $h_i, h_j \in \mathbb{R}^d$, the score function of the edge is calculated as:

$$s(i, r, j) = h_i W_r h_j + V \begin{bmatrix} h_i \\ h_j \end{bmatrix} + b \,, \tag{2.5}$$

where $W_r \in \mathbb{R}^{d \times d}$ is a diagonal matrix for any $r \in \mathcal{R}$. The parameters W_r, $V \in \mathbb{R}^{2d}$ and $b \in \mathbb{R}$ are to be learned in our training phase. Group-truth label of a positive (existing) edge is 1, otherwise 0.

The first decoder-architecture **TIMME** sums all $R + 1$ losses as $\mathcal{L} = \sum_{i=0}^{R} \mathcal{L}_i$. Without average, each task's loss is directly proportional to the amount of data points sampled at the current batch. Low-resource tasks will take a smaller portion. This is the most straightforward design of a MTL decoder.

Table 2.1: Descriptive statistics of the three selected subsets of our data set.

	PureP	P50	P20∼50	P+all
# Account	583	5,435	12,103	20,811
# Edge	122,347	1,593,721	1,976,985	6,496,107
# Labeled Account	581	759	961	1,206
# Featured Account	579	5,149	11,725	19,418
# Follow-Edge	59,073	529,448	158,746	915,438
# Reply-Edge	1,451	96,757	121,133	530,598
# Retweet-Edge	19,760	311,359	595,030	1,684,023
# Like-Edge	14,381	302,571	562,496	1,794,111
# Mention-Edge	27,682	353,586	539,580	1,571,937

The second, **TIMME-hierarchical**, has $\lambda = [\lambda_1, \ldots, \lambda_{|\mathcal{R}|}]^T$ being computed via self-attention on the average embedding over the R edge-prediction task-specific embeddings. Here, $\mathcal{L} = \sum_{i=0}^{R} \mathcal{L}_i$ is the same with **TIMME**. **TIMME-hierarchical** essentially derives the node-label information from the edge relations, thus provides some insights on each relation's importance to ideology prediction. **TIMME**, **TIMME-hierarchical**, **TIMME-single** models share exactly the same encoder architecture.

2.5 Experiments

In this section, we introduce the data set we crawled, cleaned and labeled, together with our experimental results and analysis.

2.5.1 Data Preparation

The statics of the political-centered social network data sets we have are listed in Table 2.1. Data prepared is described in Appendix 2.7.1, ready by April, 2019. In brief, we did the following:

1. Collecting some Twitter accounts of the politicians $\mathcal{P}$.

2. For every politician $p \in \mathcal{P}$, crawl their most-recent s followers and s followees, putting them in a candidate set $\mathcal{C}$.

3. For every candidate $c \in \mathcal{C}$, we also crawl their most-recent s followers to make the follow relation more complete.

4. For every account $u \in \mathcal{P} \cup \mathcal{C}$, crawl their tweets as much as possible, until we hit the limit. The limit is set by Twitter Developer API, which allows us to visit no more than $3,200$ most recent tweets per account.

5. From the followers and followees, we collect follow relation, from the tweets we extract the following relationships: retweet, mention, reply, and like.

6. Select different groups of accounts from $\mathcal{C}$, based on how many connections they have with members in $\mathcal{P}$, and making those groups into the 4 subsets, as is shown in Table 2.1.

7. We filter the relations within any selected group so that if a relation $e = \langle v_i, v_j \rangle \in \mathcal{G}_p$, there must be $v_i \in \mathcal{G}_p$ and $v_j \in \mathcal{G}_p$.

Our four data sets represent different account groups. **PureP** contains only the politicians. **P50** contains politicians and accounts keen on political affairs, indicated by that they are following or followed by at least 50 politicians. **P20∼50** is politicians with the group of accounts who are of moderate interests on politics, following or are followed by 20 to 50 politicians. The accounts considered in **P+all** is a union set of the three, plus a subset of the accounts following or being followed by only 1 to 5 politicians' accounts, selected uniformly at random, from all the followers and followees of the politicians. **P+all** is the most challenging subset to all models. More details on the data set, including how we generated features and how we tried to get more labels, are all described in details in Appendix 2.7.1.

Table 2.2: Node classification measured by F1-score/accuracy. The best performances are in **bold**.

Model	PureP	P50	P20∼50	P+all
GCN	**1.0000/1.0000**	0.9600/0.9600	0.9895/0.9895	0.9076/0.9083
R-GCN	**1.0000/1.0000**	0.9733/0.9733	0.9895/0.9895	0.9327/0.9333
HAN	0.9825/0.9824	0.9466/0.9467	0.9789/0.9789	0.9238/0.9250
TIMME-SINGLE	**1.0000/1.0000**	0.9733/0.9733	0.9895/0.9895	0.9333/0.9324
TIMME	0.9825/0.9824	**0.9867/0.9867**	**1.0000/1.0000**	0.9495/0.9500
TIMME-HIERARCHICAL	**1.0000/1.0000**	0.9733/0.9780	0.9895/0.9895	**0.9580/0.9583**

2.5.2 Performance Evaluation

In practice, we found that we do not need any features for nodes, and use one-hot encoding vector as initial feature.

We split the train, validation, and test set of node labels by 8:1:1, keep it the same across all data sets and throughout all models, measuring the labels' prediction quality by F1-score and accuracy. For edge-prediction tasks, we split all positive edges into training, validation, and testing sets by 85:5:10, keeping same portion across all data sets and all models, evaluating by ROC-AUC and PR-AUC.[2]

2.5.2.1 Baseline Methods

[2]AUC refers to area under the curve, PR refers to the precision–recuall curve, and ROC refers to the receiver operating characteristic curve.

Table 2.3: Edge-prediction measured by ROC-AUC/PR-AUC. The best performances are in **bold**.

Model	**PureP**	**P50**	**P20∼50**	**P+all**
		Follow Relation		
GCN+	0.8696/0.6167	0.9593/0.8308	0.9870/0.9576	0.9855/0.9329
r-GCN	0.8596/0.6091	0.9488/0.8023	0.9872/0.9537	0.9685/0.9201
HAN+	**0.8891/0.7267**	0.9598/0.8642	0.9620/0.8850	0.9723/0.9256
TIMME-single	0.8809/0.6325	0.9717/0.8792	0.9920/0.9709	0.9936/0.9696
TIMME	0.8763/0.6324	**0.9811/0.9154**	0.9945/0.9799	0.9943/0.9736
TIMME-hierarchical	0.8812/0.6409	0.9809/0.9145	**0.9984/0.9813**	**0.9944/0.9739**
		Reply Relation		
GCN+	0.8602/0.7306	0.9625/0.9022	0.9381/0.8665	0.9705/0.9154
r-GCN	0.7962/0.6279	0.9421/0.8714	0.8868/0.7815	0.9640/0.9085
HAN+	0.8445/0.6359	0.9598/0.8616	0.9495/0.8664	0.9757/0.9210
TIMME-single	0.8685/0.7018	0.9695/0.9307	0.9593/0.9070	0.9775/0.9508
TIMME	0.9077/0.8004	**0.9781/0.9417**	**0.9747/0.9347**	0.9849/0.9612
TIMME-hierarchical	**0.9224/0.8152**	0.9766/0.9409	0.9737/0.9341	**0.9854/0.9629**
		Retweet Relation		
GCN+	0.8955/0.7145	0.9574/0.8493	0.9351/0.8408	0.9724/0.9303
r-GCN	0.8865/0.6895	0.9411/0.8084	0.9063/0.7728	0.9735/0.9326
HAN+	0.7646/0.6139	0.9658/0.9213	0.9478/0.8962	0.9750/0.9424
TIMME-single	0.9015/ 0.7202	0.9754/0.9127	0.9673/0.9073	0.9824/0.9424
TIMME	0.9094/0.7285	0.9779/0.9181	**0.9772/0.9291**	0.9858/0.9511
TIMME-hierarchical	**0.9105/0.7344**	**0.9780/0.9190**	0.9766/0.9275	**0.9869/0.9543**
		Like Relation		
GCN+	0.9007/0.7259	0.9527/0.8499	0.9349/0.8400	0.9690/0.9032
r-GCN	0.8924/0.7161	0.9343/0.7966	0.9038/0.7681	0.9510/0.8945
HAN+	0.8606/0.6176	0.9733/0.8851	0.9611/0.9062	**0.9894**/0.9481

Continued on next page

Table 2.3 – *Continued from previous page*

Model	**PureP**	**P50**	**P20∼50**	**P+all**
TIMME-SINGLE	0.9113/0.7654	0.9725/0.9119	0.9655/0.9069	0.9796/0.9374
TIMME	0.9249/0.7926	**0.9753**/0.9171	**0.9759**/**0.9292**	0.9846/0.9504
TIMME-HIERARCHICAL	**0.9278**/**0.7945**	0.9752/**0.9175**	0.9752/0.9271	0.9851/**0.9518**
Mention Relation				
GCN+	0.8480/0.6233	0.9602/0.8617	0.9261/0.8170	0.9665/0.8910
R-GCN	0.8312/0.6023	0.9382/0.7963	0.8938/0.7563	0.9640/0.8902
HAN+	**0.9000**/**0.7206**	0.9573/0.8616	0.9574/0.8891	0.9724/0.9119
TIMME-SINGLE	0.8587/0.6502	0.9713/0.8981	0.9614/0.8923	0.9725/0.9096
TIMME	0.8684/0.6689	0.9730/0.9035	**0.9730**/**0.9185**	0.9839/0.9446
TIMME-HIERARCHICAL	0.8643/0.6597	**0.9732**/**0.9046**	0.9723/0.9166	**0.9846**/**0.9463**

We explored a lot of potential baseline methods, while only a tiny subset of them are suitable for our case. Some methods we mentioned in section 2.2, HETGNN [166], GATNE [20] and GTN [164] generally converge approximately 10 to 100 times slower than our model on any task we have. The GRAPHSAGE model [62] is not very suitable on our data set. Moreover, other well-designed models such as GIN [160] are way too different from our approach at a very fundamental level, thus are not considered as baselines. Some other methods such as GEM [91] and SHINE [152] should be capable of handling the data set at this scale, but they are not releasing their code to the public, and we can not easily guarantee reproduction.

We decided to use the three baselines: GCN, R-GCN, and HAN. They are closely-related to our model, open-sourced, and efficient. We understand that none of them were specifically designed for social networks. Early explorations without tuning them resulted in terrible outcomes. To make the comparisons fair, we did a lot of work in hyper-parameter optimization, so that their performances are significantly improved. The GCN baseline treats all edges as the same type and put them into one adjacency matrix. We also extend the baseline models to new tasks that were not mentioned in their original papers. We refer to GCN+ and HAN+ as the GCN-base-model or HAN-base-model with **TIMME-NTN** attached to it. By comparing with GCN/GCN+, we show that reserving heterogeneousity is beneficial. Comparing with R-GCN, we prove that their design is not as suitable for social networks as ours. With HAN/HAN+ we show that, although their model is potentially more expressive, our model still outperforms theirs in most cases, even after we carefully improved it to its highest potential (Appendix 2.7.3). We did not have to tune the hyperparameters of **TIMME** models closely as hard, thanks to its robustness.

HAN+ has an expressive and flexible structure that helps it achieve high in some tasks. The downsides of HAN/HAN+ are also obvious: it easily gets over-fitting, and is extremely sensitive to data set statistics, with large memory consumption that typically more than $32G$ to run tasks on **P+all**, where TIMME models takes less than $4G$ space with the same hidden size and embedding dimensions as the baseline model's settings.

2.5.2.2 TIMME

To stabilize training, we would have to use the step-decay learning rate scheduler, the same with that for ResNet. The optimizer that we use is Adam, kept consistent with GCN and R-GCN. We do not need input features for nodes, thus our encoder utilizes one-hot embedding by default. One of the many advantages of **TIMME** is how robust it is to the hyper-parameters and all other settings, reflected by that the same default parameter settings serve all experiments well. Like many others have done before, to avoid information leakage, whenever we run tasks involving edge-prediction, we will remove all edge-prediction test-set edges from our adjacency matrices.

We show in Table 2.2 and 2.3 that multi-task models **TIMME** and **TIMME-hierarchical** are generally better than **TIMME-single** on most tasks. Even **TIMME-single** is superior to the baseline models most of the times. TIMME models are stable and scalable. The classification task, despite the many labels we manually added, easily over-estimating the models. Models trained on single node-classification task will easily get over-fitted. If we

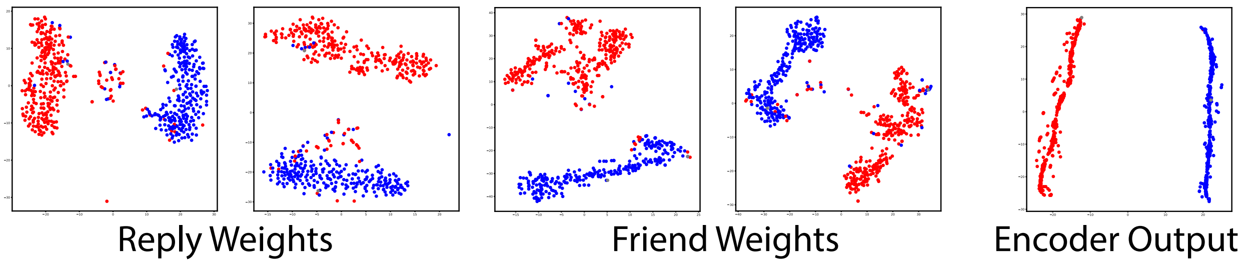


Figure 2.4: t-SNE of matrices onto 2D space. We show reply (and reversed) and friend (and reversed) weight matrices of the first convolutional layer ($W_r^{(0)}$) and the encoder output embeddings ($H^{(2)}$). Red for ground-truth Republican nodes, and blue for Democratic.

force them to keep training after convergence, only multi-task **TIMME models** keep stable. The baselines and **TIMME-single** suffer from dramatic performance-drop, especially HAN/HAN+.

2.5.3 Case Studies

In this subsection, we discuss our findings from the case studies on our **TIMME** models.

2.5.3.1 Selection of Input Features

To justify the reason that we do not need any features for nodes, we show the node-classification training-curves of **TIMME-single** with one-hot features, randomized features, partly-known-partly-randomized features, and with partly-known-partly-trainable features. The results are collected from **P50** data set. To make it easier to compare, we have fixed training epochs 300 for node-classification, and 200 for follow-relation edge-prediction. It is shown that text feature is significantly better than randomized feature, and treating the missing part of the text-generated feature as trainable is better than treat it as fixed randomized feature. However, one-hot feature always outperforms them all, essentially means that relations are more reliable and less noisy than text information in training our network embedding. We have proved in Appendix 2.7.2 that the $2R + 1$ weight matrices at the first convolutional layer captures the nodes' learned features when using one-hot features. Experimental evidence is shown in Figure 2.4. It shows that although worse than the encoder

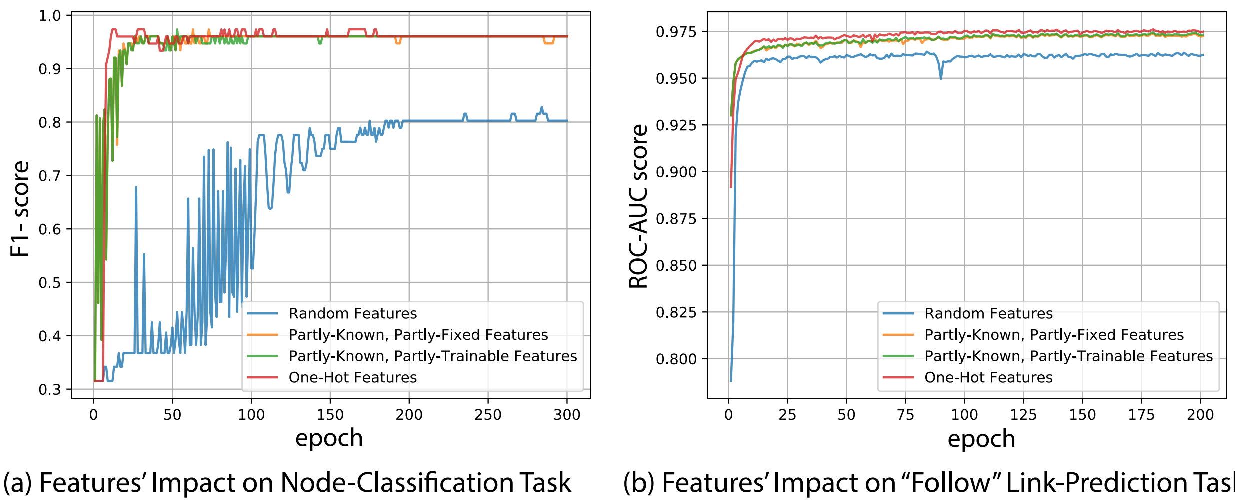


Figure 2.5: Illustration of impact of features. We show random features in blue, partly known and partly randomized (and fixed) in yellow, partly-known and partly-trainable features in green, and one-hot features in red.

output, the first embedding layer also captured the features of nodes. The embedding comes from epoch 300, node-classification task on **PureP**.

2.5.3.2 Performance Measurement on News Agencies

A good measurement of our prediction's quality is to evaluate the predicted ideology on some accounts with ground-truth ideologies, while being included but remain unlabeled in our data set (i.e., not belonging to our training set). News agents' accounts are typically such accounts, as is shown in Figure 2.8. Among them we select some of the agencies believed to have clear tendencies.[3] The continuous scores we have for prediction come from the softmax of the last-layer output of our node-classification task, which is in the format of $(\text{prob}_{\text{left}}, \text{prob}_{\text{right}})$. Right in the middle is $(\text{prob}_{\text{left}}, \text{prob}_{\text{right}}) = (0.5, 0.5)$, left-most is $(1.0, 0.0)$, and right-most is $(0.0, 1.0)$. For most cases, our model's predictions agree with people's common belief. But CNN News is an interesting case. Many people voted for that CNN is extremely left, but our model output shows that it is a slightly-left-leaning centrist.

[3]We fetch most of the ground-truth labels of the news agents from the public voting results on https://www.allsides.com/media-bias/media-bias-ratings, got them **after** the prediction results are ready.

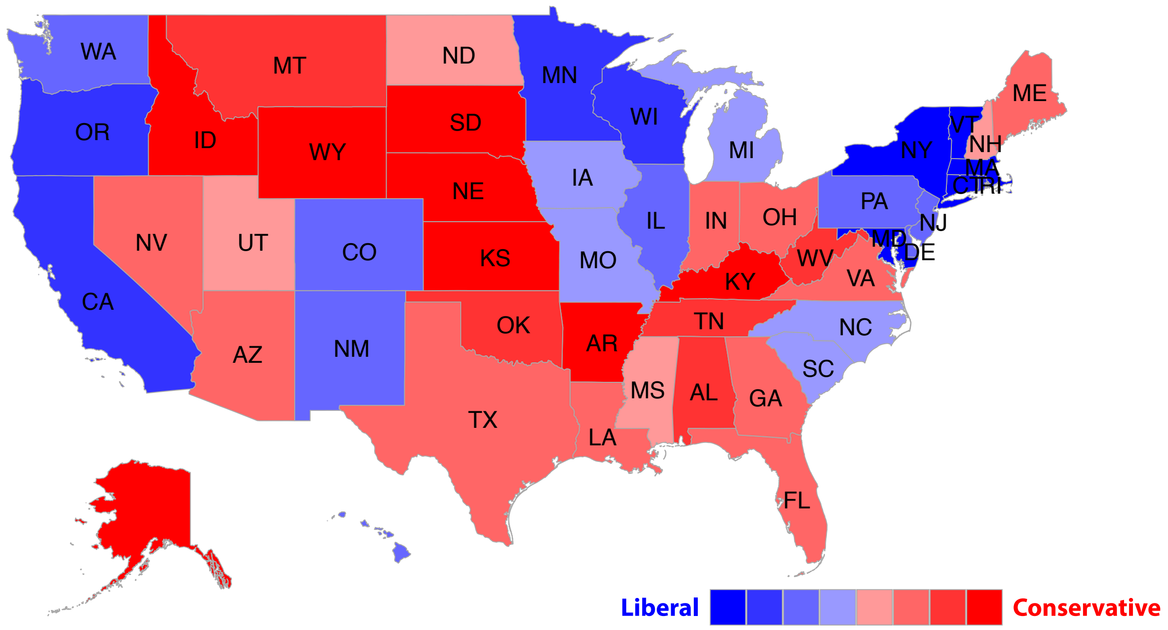


Figure 2.6: Overall ideology on Twitter in each state.

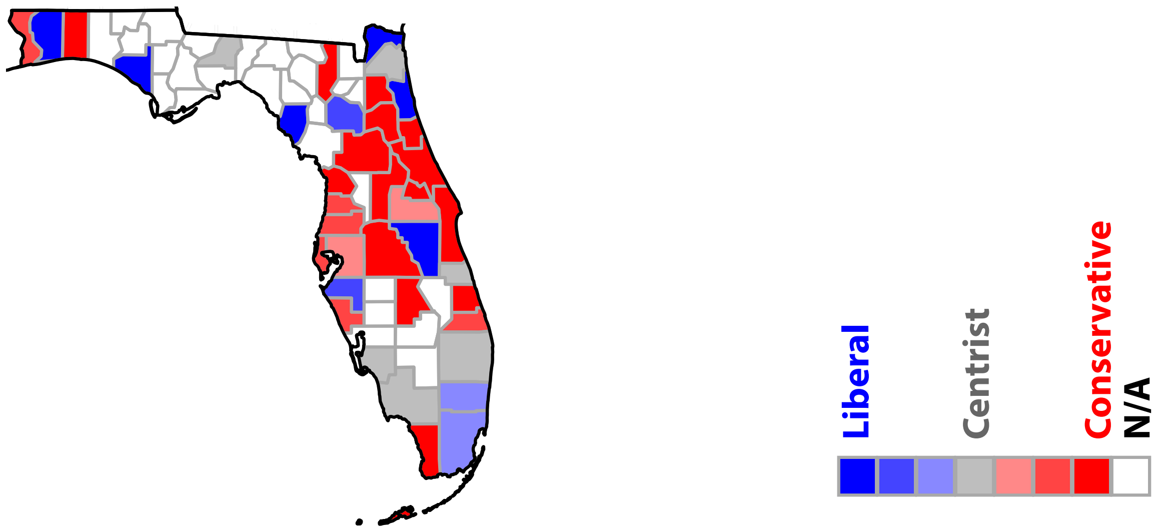


Figure 2.7: Overall ideology on Twitter, Florida (FL).

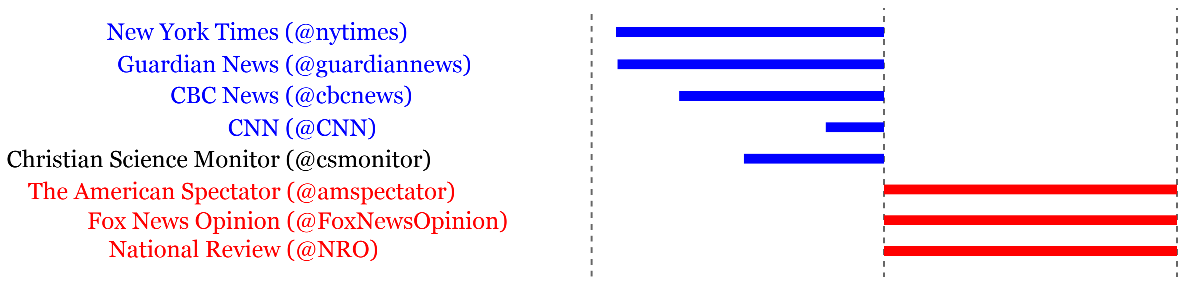


Figure 2.8: The News Agencies' Ideologies. Text colors come from the public's voting online, blue for left and red for right, and black for middle (centrist). The lengths of the bars, starting from the middle, come from the softmax of the last-layer output of our node-classification task, which is in the format of $(\text{prob}_{\text{left}}, \text{prob}_{\text{right}})$. Right in the middle is $(\text{prob}_{\text{left}}, \text{prob}_{\text{right}}) = (0.5, 0.5)$, left-most is $(1.0, 0.0)$, and right-most is $(0.0, 1.0)$.

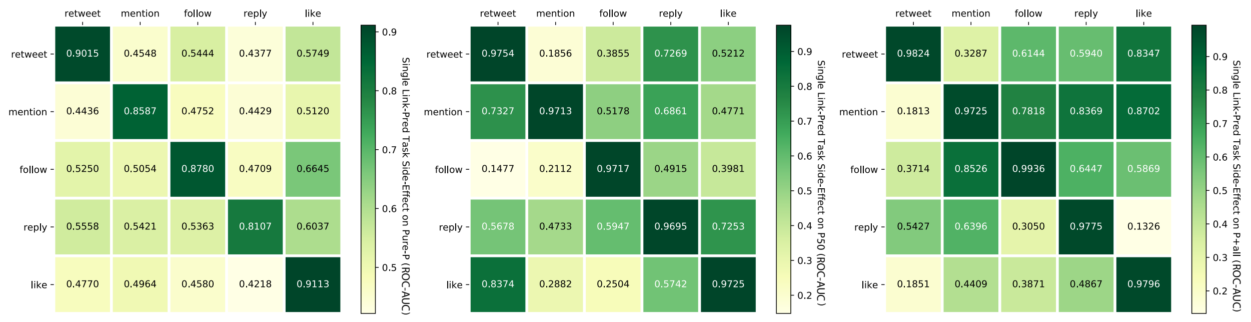


Figure 2.9: The impact of training on single-edge-prediction tasks, on Pure-P (left), P50 (middle), P+all (right) data set respectively.

Some others have findings supporting our results: CNN is actually only a little bit left-leaning.[4,5] Although the public tends to believe that CNN is extremely liberal, it is more reasonable to consider it as centrist biased towards left-side. People's general belief on news agencies' ideologies might be more polarized than their actual stance. Besides, although there are typically more famous news agencies on the liberal side [46, 70], according to our experimental results, those right-leaning ones tend to support their side more firmly.

2.5.3.3 Geography Distribution

Consider results from the largest data set (**P+all**), and with predictions coming out from **TIMME-hierarchical**. We predict each Twitter account's ideology as either liberal or conservative. Then we calculate the percentage of the accounts on both sides, and depict it in Figure 2.6. Darkest red represents $p \in [0, \frac{1}{8}]$ of accounts in that area are liberal, remaining $[\frac{7}{8}, 1]$ are conservative; darkest blue areas have $[\frac{7}{8}, 1]$ accounts being liberal, $[0, \frac{1}{8}]$ conservative. The intermediate colors represent the evenly-divided ranges in between. The accounts' locations are collected from the public information in their account profile. From our observation, conservative people are typically under-represented. [6,7] For instance, as a well-known firmly-conservative state, Utah (UT) is only shown as slightly right-leaning on our map.

That our results on states like Utah being less extreme than people's common belief is intuitively reasonable, since Twitter accounts are also biased. Typically biased towards youngsters and urban citizens. Although we are able to solve the problem of silent-majority by utilizing their edge relations instead of text expressions, we know nothing about offline ideology. We suppose that some areas are silent on Twitter, and this guess is supported by

[4]https://libguides.com.edu/c.php?g=649909&p=4556556

[5]https://www.allsides.com/news-source/cnn-media-bias

[6]National General Election Polls data are partly available at https://www.realclearpolitics.com/epolls/2020/president/National.html.

[7]Compare with the visualization of previous election at https://en.wikipedia.org/wiki/Political_party_strength_in_U.S._states.

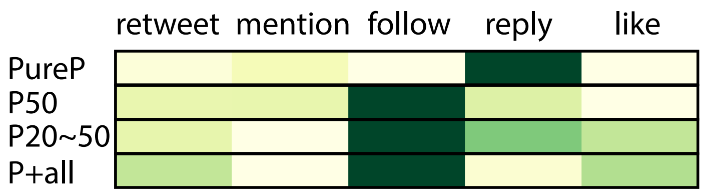


Figure 2.10: Illustration of the task-weight parameter λ value in decoder on each data set.

the county-level results at Florida, shown in Figure 2.7. This time the color-code represents evenly-divided seven ranges from $[0, \frac{1}{7}]$ to $[\frac{6}{7}, 1]$, because of the necessity of reserving one color for representing silent areas (denoted as white for N/A). The silent counties, typically some rural areas, have no account in our data set, inferring that people living there do not use Twitter very often. The remaining parts of the graph makes complete sense, demonstrating a typical swing state.[8]

2.5.3.4 Correlated Relations

When we train **TIMME-single** with only one relation type, some other relations' predictions benefit from it; they become increasingly accurate. We assume that, if by training on relation r_i we achieve a good performance on relation r_j, then we say relation r_i probably leads to r_j. As is shown in Figure 2.9, relations among politicians are relatively independent except that all other relations might stimulate like. In more ordinary account groups, reply is the one that significantly benefit from all other relations. It is also interesting to observe that the highly-political P50 shows that like leads to retweet, while from more ordinary accounts' perspective once they liked they are less likely to retweet. The relations among the relations are asymmetric.

2.5.3.5 Relation's Contributions to Ideology Detection

The importance of each relation to ideology prediction can be measured by the value of the corresponding weight value λ_r that measures the relative importance of the corresponding

[8]The ground-truth election outcome in Florida at 2016 is at https://en.wikipedia.org/wiki/2016_United_States_presidential_election_in_Florida.

task r, in the decoder of **TIMME-hierarchical**. All the values are close to 0.2 in practice, in the range $[0.99, 2.01]$, but they still have some common trends, as is shown in Figure 2.10. Despite that reply pops out rather than follow on **PureP**, we still insist that follow is the most important relation. That is because we only crawled the most recent about 5000 followers and the most recent 5000 followees. If one account follows another account, a long time ago, we do not capture it. The follow relation is especially incomplete on **PureP**.

2.6 Conclusion

The **TIMME models** that we proposed handles multiple relations, with a multi-relational encoder, and multi-task decoder. We step aside the silent-majority problem[9] by relying mostly on the relations, instead of the text information. Optionally, we accept incomplete input features, but we showed that edges are able to do well on generating the ideology embedding without additional text information. From our observation, the edges help much more than naively-processed text in ideology-detection problem, and follow is the most important relation to ideology detection. We also concluded from visualizing the state-level overall ideology map that conservative voices tend to be under-represented on Twitter. Meanwhile we confirmed that public opinions on news agencies' ideology can be polarized, with very obvious tendencies. Our model can be easily extended to any other social network embedding problem, such as on any other data set like Facebook as long as the data set is legally available, and of course it works on predicting other tendencies like preferring Superman or Batman. We also believe that our data set would be beneficial to the community.

[9]In terms of the tweet contents, y, about 97% of the tweets are produced by only the most active 25% of accounts [142].

2.7 Appendix

In this section, we provide the supplementary materials that can help our readers better understand our work.

2.7.1 Data Preparation

We target at building a data set representing the political-centered social network (Section 2.3), a selected subset from the giant Twitter network. Handling this data set would be challenging. For example, for GraphSAGE, neighborhood-sampling can not be easily done both effectively and efficiently. Our data set reaches the blind spots of many existing models.

The tools we used to crawl politicians' name lists from the government website, and their potential Twitter accounts from Google, is Scrapy.[10] To legally and reliably crawl from Twitter data, we first applied for Developer API from Twitter[11], and then used Tweepy[12] for crawling. We set very strict rate limits for our crawlers so as not to harm any server. Our data set is released at `https://github.com/PatriciaXiao/TIMME`. Raw data was collected starting Jan 2019, and finished by April 2019.

2.7.1.1 Twitter IDs Preparation

Using the same notation as in Section 2.3, we describe our procedure to prepare Twitter IDs. To construct $\mathcal{G}_p = \{\mathcal{V}, \{\mathcal{E}_1, \mathcal{E}_2, \mathcal{E}_3, \mathcal{E}_4, \mathcal{E}_5\}\}$, we first select the users to be included $\mathcal{V}$, then we include the links between vertices in $\mathcal{V}$ under each relation $r \in \mathcal{R} = \{1, 2, 3, 4, 5\}$ in $\mathcal{E}_r$.

Politicians Twitter IDs

As is described briefly in Section 2.5.1, we need to start from a set of politicians $\mathcal{P}$, which we treat as seeds for further crawling.

[10] `https://scrapy.org/`

[11] `https://developer.twitter.com/`

[12] `https://www.tweepy.org/`

To start, we first obtain the name list of the recently-active politicians. This list consists of the following items:

- The union-set of 115th and 116th US Congress members, where we observe a lot of overlap between the two groups.[13]
- Recent-years' presidents and their cabinets.[14]
- Additional politicians that are well-known to the public: Hilary Clinton, who recently ran for US president; and Michelle Obama, who was the former First Lady.

Next, with the help of Google, we crawled the most-likely Twitter names and IDs of the politicians. We do so automatically, by providing Google a politician's name and the keyword "twitter" and parsing the first response. Then after manual filtering, we have 583 politicians' Twitter accounts available, who make up our politicians set $\mathcal{P}$. Anyone else to be included in our data set must be in the 1-hop neighborhood of a politician (Section 2.3).

Candidate Non-Politicians Twitter IDs

With the help of Twitter Developer API, we are able to get the **full** followers and followees list of any Twitter user.

However, it is not affordable to include all followers and followees of the politicians, thus we set a limit on *window size* s when crawling the candidate non-politicians list, only accepting the most-recent $s = 5{,}000$ followers or followees of any politician. These followers and followees we collected form a raw candidate set $\mathcal{C}_{raw}$. Then we remove the politicians from this set, resulting in the final candidates set $\mathcal{C} = \mathcal{C}_{raw} - \mathcal{P}$. For all $v_i \in \mathcal{C}$, we apply the same window size $s = 5{,}000$ and crawled their most recent s followers, and s followees. All follower-followee pairs are stored into a database for the convenience of the following steps.

[13]Congress members' name list with party information is publicly available at `https://www.congress.gov/members` .

[14]Obama and Trump's cabinets are publicly available at `https://obamawhitehouse.archives.gov/administration/cabinet` and `https://www.whitehouse.gov/the-trump-administration/the-cabinet/`, respectively.

Selecting Subgroups from Candidates

The user set $\mathcal{C}$ is still too large, and we do not know anything about its components. To conduct meaningful analysis, we need to select some meaningful subgroups from it, such as a very-political subgroup and a political-outliers subgroup, etc.

The criteria we used to select the desired subgroups of users is some thresholds. We define a political-measurement t_i for each user $v_i \in \mathcal{C}$, who is followed by $t_{i,1}$ politicians $p \in \mathcal{P}$, and meanwhile following $t_{i,2}$ politicians, so t_i is computed by $t_i = \max(t_{i,1}, t_{i,2})$.

Then we set a threshold range t, set upon each t_i, used for filtering the groups of users. Considering we set t as threshold range for graph $\mathcal{G}_p$, for all $v_i \in \mathcal{V}$, if $t_i \in t$, then $v_i \in \mathcal{G}_p$; otherwise $v_i \notin \mathcal{G}_p$. By having $t = \{\infty\}$, we select a minimum subgraph containing purely politicians, resulting in our **PureP** data set. Setting the threshold $t \in [50, \infty)$ allows us to select a small group of users who are keen on political topics, together with the politicians, being our **P50** data set. We set $t \in [20, 50)$ for less-political users, plus the politicians, being our **P20∼50** data set. The data set generated using $t \in [20, \infty)$ includes all nodes v_i with $t_i \geq 20$. We want to have a data set representing more general users, containing some users from each group. Therefore, we include another 3,000 users randomly selected from the group $t \in [0, 5)$. Adding these random political-outlier users will make the data set resembles the real network even more. Putting together the politicians, the $t \in [20, \infty)$ group, and the $3,000$ random outliers from $t \in [0, 5)$ group, we form the data set **P+all**. Ideally, **P+all** has representatives of all groups of users on Twitter. The statistics are concluded in Table 2.1.

2.7.1.2 Relation Preparation

Only the *follow* relation is directly observed and already well-prepared at this stage (stored in a database, as we mentioned before). The other Twitter relations: *retweet*, *mention*, *like*, *reply* — must be concluded from tweets. We distinguish the different relation types from the tweets by the tweets' fields in responded Json from API. For example, there are some fields indicating if an **“”** mark is a *mention*, a *retweet*, or it links to nothing. According to our observation, the fields in the Json file responded from Twitter API might change across time. We don't know when will it be the next update, so there's no ground-truth solution for this part. We suggest whoever want to do so test the crawler first on her/his own account, trying all behaviors to conclude some patterns. Note: rate limit applies.[15]

Due to the Twitter official API limits, the maximum amount of tweets we can crawl for each user along the timeline is around 3,200. Therefore, all relations are somewhat incomplete observations. All links we have only reflect some recent interactions between the users. One should keep the limitations of our data set in mind when using it.

2.7.1.3 Feature Preparation

We get feature from text, using users' tweets posted to generate their features. Although there has been some recent advances in NLP with transformer-based structures, such as BERT and XLNet, Sentence-BERT [125] found that BERT/XLNet embeddings are generally performing worse than taking the mean value of all words' GloVe [115] embeddings on many sentence-level tasks. Not to mention the computational cost of transformers. We therefore use GloVe mean of the words as features, Wikipedia 2014 + Gigaword 5, 300-dimensional pre-trained version. When we apply the GloVe mean embedding on tweet-level and want to tell the ideology behind the tweets, we can easily achieve about 72.84% accuracy, using a 2-layers MLP, after only 200 epochs of training.

2.7.1.4 Label Preparation

If we are to use only the 583 labels from the politicians, the evaluation will always be untrustworthy. To overcome this issue, we manually expand the labels. We first crawled the users' profiles of each $v_i \in \mathcal{P} \cup \mathcal{C}$, getting their information such as location and account description. Next, using the descriptions, searching for the words *democratic*, *republican*, *conservative*, *liberal*, their correct spell and variations, we have a large group of candidates.

[15] https://developer.twitter.com/en/docs/basics/rate-limiting

Then we do manual filtering to get rid of the uncertain users, reading their descriptions and recent tweets. We successfully included 2,976 high-quality new labels in the end. Those labels make the node-classification task significantly more stable and reliable.

2.7.2 Explanations on Weight being Feature

Starting from our layer-wise propagation formula, we have that, at the first convolutional layer (notations in Section 2.4):

$$H^{(1)} = \sigma\Big(\sum_{r \in \hat{\mathcal{R}}} \alpha_r \hat{A}_r H^{(0)} W_r^{(0)}\Big),$$

where $H^{(0)} \in N \times d^{(0)}$ is the input feature-matrix. When using one-hot embedding of features, $H^{(0)} = I$ and $d^{(0)} = N$, so the right-hand-side is equivalent with $\sigma\Big(\sum_{r \in \hat{\mathcal{R}}} \alpha_r \hat{A}_r W_r^{(0)}\Big)$. Now, $W_r^{(0)}$ on its own plays the role of $H^{(0)} W_r^{(0)}$ when $H^{(0)} \neq I$. An intuitive way of understanding is to view the relation r's propagation as aggregation of a linear transformation ($W_r^{(0)}$) done on $H^{(0)}$, from the neighborhood ($\hat{A}_r$) of each node under relation r. We propose that this procedure can also be viewed as simply the propagation of $W_r^{(0)} \in \mathbb{R}^{N \times d^{(1)}}$. In terms of what we compute, having one-hot features and weight matrix $W_r^{(0)}$ is equivalent as having input features being $\tilde{H}^{(0)} = W_r^{(0)} \in \mathbb{R}^{N \times d^{(1)}}$, and set $\tilde{W}_r^{(0)} \in \mathbb{R}^{d^{(1)} \times d^{(1)}} = I_{d^{(1)}}$ being fixed identical matrix not to be updated. That's the reason why we believe that $W_r^{(0)}$ captures the nodes' learned features under relation r, and drew Figure 2.4.

2.7.3 Baseline Hyper-Parameter and Architectural Optimizations

In this subsection, we discuss how we train our baselines on our data sets.

2.7.3.1 Applying GCN model Directly

As is discussed in Section 2.2, due to the uniqueness of the *political-centered social network data set*, most of the existing models won't work well under our problem settings. We want to examine how well can GCN do when treating all relations as the same, ignoring the heterogeneous types. Very interestingly, without much work on hyper-parameter optimization, we only increased the hidden size and added the learning rate scheduler, it works pretty well. This phenomenon can potentially be an indirect evidence that relations are correlated, in addition to the discussions in Section 2.5.

2.7.3.2 Missing-Task Completion

We compare our model's performance on each task with the baselines. Ideally, we want models working on heterogeneous information networks with both node-classification task and link-prediction task as our baselines, so that we can compare with them directly. However, the situation we faced was not as easy as such. For instance, GCN and HAN never considered applying themselves directly on link-prediction tasks. But we all know that once we have the embeddings of the nodes, link prediction is doable.

Therefore, we decided that whenever a baseline originally can't handle a task, we lend it our decoder's task-specific cells. This decision brings about some significant improvements on the link prediction performances of NTN+ and GCN+, since **TIMME-NTN** is powerful and efficient for link-prediction. Just in case, we also decide that when a node-classification task is missing, we should add a linear transformation layer with output units 2, the same as what we did, and apply a simple cross-entropy loss. From this perspective, it is no longer fair to compare them with r-GCN directly. To distinguish them from others' standard models, we add a plus sign **"+"** to the names, indicating that "we lend it our cells".

2.7.3.3 Optimizing r-GCN

The most important contribution of r-GCN is the weight-matrix decomposition methods. This mechanism would be very helpful in reducing the parameters, especially when the number of relations R is super high. However, in our case, where R is small, the weight-decomposition operation is counter-effective. The first option, basis decomposition, the number of basis b is easily being larger than R. In the second option, block-diagonal decomposition, reduces the parameter size too dramatically, and harms the model's performance.

Reviewing the experiments reported in the R-GCN paper, seeing how they chose these hyper-parameters across data sets, we found that when R is small, they often chose basis-decomposition with $b = 0$. We go by the same option, which works well in practice.

2.7.3.4 Optimizing HAN

HAN/HAN+, in general, because of the complex structure with a lot of parameters, is easily overfit. What makes things worse, its training curve is never stable, and our early tryouts on using validation set to automatically stop it at an optimal point did not work well. We had do it manually, by verifying when its best result appears on the validation set and when over-fitting starts, finding the right time to stop training. By default, we set learning rate 0.005, regularization parameter 0.001, the semantic-level attention-vector dimension 128, multi-head-attention cell's number of heads $K = 8$. We set the hyper-parameters in the **TIMME-NTN** component of HAN+ the same with ours. Optimizing HAN was a tough work to do, for it requires re-adapting every choices we made on every data set for every task. Adding more meta-path would potentially boosting its performance, but the computational cost will be overwhelming. Another observation is that, **TIMME models** are significantly better than HAN/HAN+ in handling imperfect features. When using GLOVE-mean features, **TIMME models** typically perform about 1% worse than using one-hot features, while HAN/HAN+ experience performance-drop up to around 10%.

Declarations

Funding

This work is partially supported by NSF III-1705169, NSF CAREER Award 1741634, NSF #1937599, DARPA HR00112090027, Okawa Foundation Grant, and Amazon Research Award.

Weiping Song is supported by National Key Research and Development Program of China with Grant No. 2018AAA0101900/ 2018AAA0101902 as well as the National Natural Science Foundation of China (NSFC Grant No. 61772039 and No. 91646202).

Authors' Contributions

Zhiping Xiao lead the discussions, designed the model under supervision of Prof. Yizhou Sun, and was in charge of most of the coding and writing.

Zhiping Xiao and Prof. Yizhou Sun conceived and conceptualized the study. Zhiping Xiao, Weiping Song, Haoyan Xu and Zhicheng Ren performed the analysis and wrote the initial draft of the paper. Zhiping Xiao, Weiping Song and Prof. Yizhou Sun reviewed and extensively edited the manuscript, determined what additional analysis was necessary, and produced the final version of the manuscript.

Acknowledgements

At the early stage of this work, Haoran Wang [16] contributed a lot to a nicely-implemented first version of the model, benefiting the rest of our work. Meanwhile, Zhiwen Hu [17] explored the related methods' efficiencies, and his works shed light on our way.

Our team also received some external help from Yupeng Gu. He offered us his crawler code and his old data set as references.

[16] *wanghran@gmail.com*, currently working at Snap Inc.

[17] *huzhiwen@g.ucla.edu*, currently working at PayPal.

CHAPTER 3

Constructing Polarity-Aware Embeddings Using Multi-Task Learning

In this work, [1] we aim at inferring explainable polarity scores of the user accounts, given the text contents of their posts. Ideological divisions in the United States have become increasingly prominent in daily communication. Accordingly, there has been much research on political polarization, including many recent efforts that take a computational perspective. By detecting political biases in a corpus of text, one can attempt to describe and discern the polarity of that text. Intuitively, the named entities (i.e., the nouns and the phrases that act as nouns) and hashtags in text often carry information about political views. For example, people who use the term "pro-choice" are likely to be liberal, whereas people who use the term "pro-life" are likely to be conservative. In this paper, we seek to reveal political polarities in social-media text data and to quantify these polarities by explicitly assigning a polarity score to entities and hashtags. Although this idea is straightforward, it is difficult to perform such inference in a trustworthy quantitative way. Key challenges include the small number of known labels, the continuous spectrum of political views, and the preservation of both a polarity score and a polarity-neutral semantic meaning in an embedding vector of words. To attempt to overcome these challenges, we propose the **P**olarity-aware **E**mbedding **M**ulti-task learning (**PEM**) model. This model consists of (1) a self-supervised context-preservation task, (2) an attention-based tweet-level polarity-inference task, and (3) an adversarial learning task that promotes independence between

[1]This work was published in the EPJ (i.e. European Physical Journal) – Data Science, in year 2023 [159]. Zhiping Xiao lead the discussions, designed the model under supervision of Prof. Yizhou Sun and Prof. Mason A. Porter, and was in charge of most of the coding and writing.

an embedding's polarity dimension and its semantic dimensions. Our experimental results demonstrate that our **PEM** model can successfully learn polarity-aware embeddings that perform well at tweet-level and account-level classification tasks. We examine a variety of applications — including spatial and temporal distributions of polarities and a comparison between tweets from Twitter and posts from Parler — and we thereby demonstrate the effectiveness of our **PEM** model. We also discuss important limitations of our work and encourage caution when applying the **PEM** model to real-world scenarios.

3.1 Introduction

In the United States, discourse has seemingly become very polarized politically and it often seems to be divided along ideological lines [82, 156]. This ideological division has become increasingly prominent, and it influences daily communication.

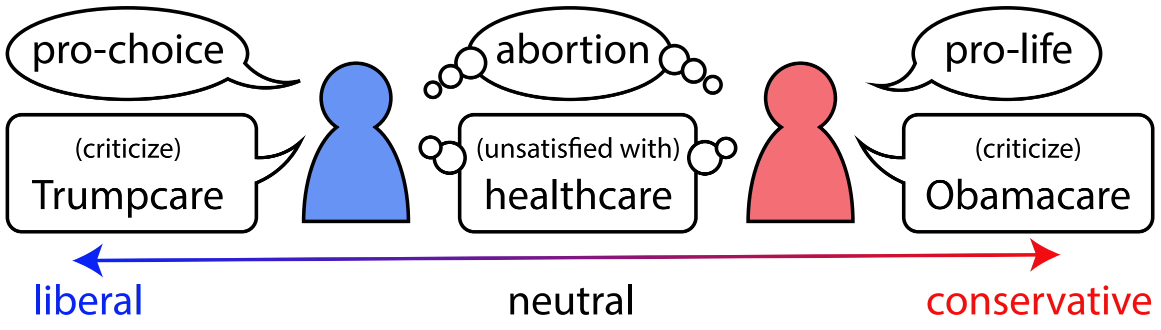


Figure 3.1: Illustration of inferring political polarities from text.

The analysis of data from social media is important for studying human discourse [21, 135]. To study the polarization of social opinions in online communication, we attempt to detect polarity biases of entities and hashtags. There are a variety of ways to model political biases; see, e.g., VoteView (see `https://voteview.com/`) [14]. A space of political opinions can include axes for social views (e.g., ranging from "conservative" to "progressive"), economic views (e.g., ranging from "socialist" to "capitalist"), views on government involvement (e.g., ranging from "libertarian" to "authoritarian"), and many others. The simplest model of a political spectrum, which we use in the present paper, is to consider a one-dimensional

(1D) political space with views that range from “liberal” to “conservative”.

By glancing at a corpus of text (such as a newspaper article or a tweet), humans can often readily recognize particular views in it without the need to analyze every word in the corpus. Many items (including named entities and hashtags) in a corpus of text are helpful for inferring political views [51], and people can quickly discern political views even in small corpora of text or in short speeches.

On Twitter, political biases are often reflected in the entities and hashtags in tweets. The entities that we use are nouns and noun phrases (i.e., phrases that act as nouns), which we identify from text corpora by using existing natural-language-processing (NLP) tools. For instance, as we illustrate in Figure 3.1, if somebody uses the term “pro-choice” to describe abortion, they may have a liberal-leaning stance on a liberal–conservative axis of political views [129]. By contrast, if somebody uses the term “pro-life”, perhaps they have a conservative-leaning stance. We propose to automate this process in an interpretable way by detecting the political biases of entities and hashtags, inferring their attention weights in tweets, and then inferring the political polarities of tweets.

The problem of inferring political polarities from text is somewhat reminiscent of “fairness-representation” problems [16, 172]. This analogy is not perfect, and these problems have different objectives. We aim to reveal polarities, whereas fairness studies are typically interested in removing polarities. The notion of fairness entails that outputs are unaffected by personal characteristics such as gender, age, and place of birth. In recent studies, Zhao et al. [172] examined how to detect and split gender bias from word embeddings and Bose and Hamilton [16] developed models to hide personal information (such as gender and age) from the embeddings of nodes in graph neural networks (GNNs). Political bias can be more subtle and change faster than other types of biases. A key challenge is the labeling of political ideologies. Unlike the inference of gender bias, where it is typically reasonable to use discrete (and well-aligned) word pairs such as “he”/“she” and “waiter”/“waitress” as a form of ground truth, political polarity includes many ambiguities [146]. Political ideology exists on a continuous spectrum, with unclear extremes, so it is very hard to determine either ground-truth polarity scores or well-aligned word pairs (e.g., “he” versus “she” is aligned with “waiter” versus “waitress”) [117].

To infer polarities, we seek to learn an embedding that can help reveal both the semantic meaning and the political biases of entities and hashtags. We propose a model, which we call the **P**olarity-Aware **E**mbedding **M**ulti-task learning (**PEM**) model, that involves three tasks: (1) preservation of the context of words; (2) preservation of corpus-level polarity information; and (3) an adversarial task to try to ensure that the semantic and polarity components of an embedding are as independent of each other as possible.

Our paper makes the following contributions:

(1) We raise the important and practical problem of studying political bias in a corpus of text, and we assemble a data set from Twitter to study this problem. Our code, the data sets of the politicians, and the embedding results of our models are available at `https://bitbucket.org/PatriciaXiao/pem/src/master/`.

(2) We propose the **PEM** model to simultaneously capture both semantic and political-polarity meanings.

(3) Our **PEM** model does not rely on word pairs to determine political polarities. Consequently, it is flexible enough to adapt to other types of biases and to use in other context-preservation strategies.

(4) Our data, source code, and embedding results are helpful for tasks such as revealing potential political polarities in a text corpus.

3.2 Related Work and Preliminary Discussions

In this section, we discuss our related works and preliminaries from five perspectives. They are, political-polarity detection, neural word embeddings, fairness of representations, sentiment analysis, and recognition of named entities.

3.2.1 Political-Polarity Detection

There are a variety of ways to formally define the notion of political polarity [14]. We consider a 1D axis of political views that range from "liberal" to "conservative". In the United States, members of the Democratic party tend to be liberal and members of the Republican party tend to be conservative [82, 88]. This prior knowledge is helpful for acquiring high-quality labeled data [158], but such data are restricted in both amount and granularity.

The detection of political polarity has been a topic of considerable interest for many years [97, 116]. Additionally, for more than a decade, social-media platforms like Twitter have simultaneously been an important source of political opinion data and have themselves impacted political opinions in various ways [4, 8]. Some researchers have attempted to infer the political views of Twitter accounts from network relationships (such as following relationships) [59, 147, 158]. Other researchers have attempted to infer polarity from tweet text [69, 80].

We seek to infer the political polarities of entities and hashtags in tweets. Gordon et al. [55] illustrated recently that word embeddings can capture information about political polarity, but their approach does not separate polarity scores from embeddings and thus cannot explicitly tell which words are biased. Most prior research has focused on tweet-level or account-level polarities [73, 150] or on case studies of specific "representative" hashtags [121]. By contrast, our **PEM** model focuses on biases at a finer granularity (specifically, entities and hashtags).

3.2.2 Neural Word Embeddings

We use the term *neural word embeddings* to describe approaches to represent tokens (e.g., words) using vectors to make them understandable by neural networks [10, 83, 87]. Words can have very different meanings under different tokenizations. In our paper, we tokenize text into entities (including nouns and noun phrases), hashtags, emoji, Twitter handles, and other words (including verbs, adjectives, and so on). One way to obtain a neural word embedding is the Skip-gram version of word2vec approaches [101], which are based on the assumption that similar words have similar local textual contexts. Another approach, which is called GloVe [115], relies on a global co-occurrence matrix of words. Other methods, such as transformers [41, 148], generate contextualized embeddings (in which a word can have different embeddings in different contexts). These models encode words, which initially take the form of a sequence of characters, into a vector space. Therefore, these models are also often called "encoders".

In contrast to all of the above studies, our **PEM** model learns an embedding that captures both the semantic meanings and the political polarities of words. Our framework is not limited to any specific embedding strategy. If desired, one can replace the embedding part (namely, Task #1) of our **PEM** model by other encoders.

3.2.3 Fairness of Representations

Many researchers have observed that word embeddings often include unwanted biases [99]. In studies of fairness, a model is considered to be "fair" if its outputs are unaffected by personal characteristics, such as gender and age; it is "biased" (i.e., "unfair") if such features influence the outputs. Models often inherit biases from training data sets, and they can exacerbate such biases [112]. Researchers have undertaken efforts to reveal biases and mitigate them [16]. For example, Zhao et al. revealed gender-bias problems using their WinoBias model [171] and attempted to generate gender-neutral representations using their GN-GloVe model [172].

Such representation-learning algorithms motivate us to separate politically-biased and politically-neutral components in embeddings (see [172]) and to use an adversarial training framework to enhance the quality of the captured polarities (see [16]). However, our work has a different focus than [172] and [16]. These works were concerned with reducing biases, whereas we seek to reveal differences between polarized groups.

3.2.4 Sentiment Analysis

Sentiment analysis aims to determine the attitude (negative, positive, or neutral) of a corpus of text [98, 136]. The use of neural word embeddings is common in statistical approaches to sentiment analysis [49, 162]. Some of these approaches account for the importance levels of entities [9, 141].

In many applications, sentiment analysis has relied on much richer labeled data sets than those that are available in political contexts [136, 144], where it is rare to find high-quality anchor words (such as good, bad, like, and dislike) [162]. In our paper, we seek to reveal polarities from textual data. Polarity is different from sentiment. For example, most entities have neutral sentiments, but these same entities can still have biased polarities.

3.2.5 Recognition of Named Entities

We focus on learning polarity scores for named entities (specifically, nouns and noun phrases) and hashtags. The terminology "named entity", which comes from NLP, refers to a noun or a noun phrase that is associated with an entity. For example, the *United States Congress* is a named entity. We use a named-entity recognition (NER) tool [84, 105] to identify the entities in our training corpus. In an NER information-extraction task, one seeks to discern and classify entities in a text corpus into predefined categories, such as person names, organizations, and locations. We use the popular tools TagMe [44] and AutoPhrase [137] for our tasks.

3.3 Problem Definition

We use "tokens" to denote the smallest word units that we obtain through tokenization of tweets. We tokenize entities, hashtags, emoji, mentioned accounts, and other words. We represent each tweet as a sequence of such tokens. We study the problem of detecting the political biases of entities and hashtags in tweets. To do this, we seek to learn (1) semantic embeddings for each token and (2) the political polarities of each entity and hashtag. We then obtain tweet-level polarity scores by calculating a weighted average of token-level polarity scores.

Definition 3.3.1. (Two-Component Polarity-Aware Embeddings) We design a two-component polarity-aware embedding $\mathbf{z} \in \mathbb{R}^{d_1+d_2}$ of each token $\mathbf{w}$. Because we seek to learn 1D polarity scores, we set $d_2 = 1$. We decompose $\mathbf{z}$ as follows:

$$\mathbf{z} = \left[\mathbf{z}^{(s)}\,,\, \mathbf{z}^{(p)}\right], \quad \mathbf{z}^{(s)} \in \mathbb{R}^{d_1}\,,\, \mathbf{z}^{(p)} \in \mathbb{R}^{d_2}\,.$$

The two components of the embedding $\mathbf{z}$ are

(1) the **polarity-neutral** semantic component $\mathbf{z}^{(s)}$ and

(2) the **polarity-aware** political-polarity component $\mathbf{z}^{(p)}$.

By forcing $\mathbf{z}^{(s)}$ to be polarity-neutral, we seek to enhance the quality of the political polarities that we capture in $\mathbf{z}^{(p)}$. We set $d_1 = d$ and $d_2 = 1$, and we use $f(\mathbf{z}^{(p)}) = z_{d+1}$ as the "polarity score" of a token. When determining tweet-level polarities, we ignore $\mathbf{z}^{(p)}$ for tokens that are neither entities nor hashtags. We expect that $z_{d+1} < 0$ when a word is liberal-leaning and that $z_{d+1} > 0$ when a word is conservative-leaning. The absolute value $|z_{d+1}|$ indicates the magnitude of a political leaning. Using our approach, we are able to infer the political polarity of a token in $\mathcal{O}(1)$ time. We are interested in the polarity scores of tokens that are either entities or hashtags. It is very common to use a 1D polarity score [14], so we do so in the present paper. However, it is straightforward to extend our **PEM** model to incorporate more polarity dimensions.

3.4 Methodology

In this section, we discuss how we design our **PEM** model. Our model consists of three parts, and could be regarded as a multi-task framework where each of the three parts handles a different task.

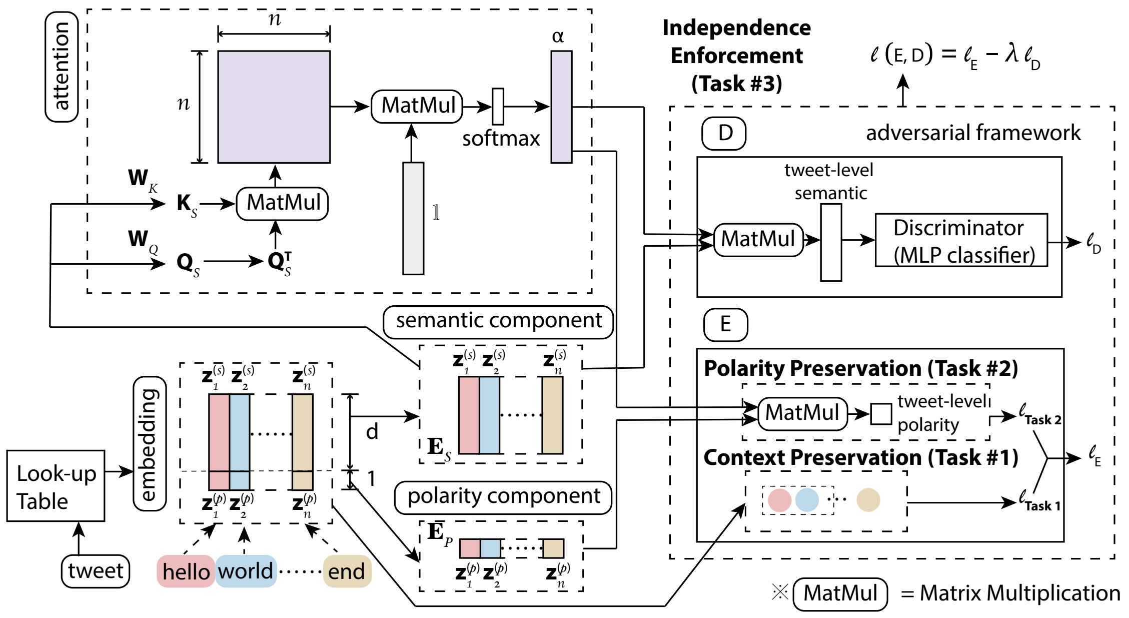


Figure 3.2: Schematic illustration of our **PEM** model. In this illustration, we consider a tweet with n tokens.

3.4.1 General Design

To generate our proposed embeddings, we infer semantic meanings, infer political polarities, and use $\mathbf{z}^{(p)}$ to capture as much political polarity as possible.

We show a schematic illustration of our model in Figure 3.2. To capture the meanings of tokens, we learn embeddings from the context of text. We thus propose Task #1 to help preserve contextual information. To infer political polarities from tokens, we propose Task #2, in which we use a weighted average of the entities' and hashtags' polarity component $\mathbf{z}^{(p)}$ to calculate a polarity score of each tweet. To further enhance the quality of the polarity component, we propose Task #3, in which we use an adversarial framework to ensure that the two components, $\mathbf{z}^{(s)}$ and $\mathbf{z}^{(p)}$, are as independent as possible.

3.4.2 Task #1: Context Preservation

We want our token-level embeddings to preserve contextual information, which has both semantic information and polarity information. A simple approach is to use SKIP-GRAM [101].

Given a document with tokens $w_1, w_2, \ldots, w_n$, we seek to maximize the mean log probability to observe tokens in a local context. Specifically, we maximize

$$\frac{1}{n} \sum_{t=1}^{n} \sum_{j \in \{-c,\ldots,c\},\, j \neq 0} \ln p(w_{t+j}|w_t)\,, \tag{3.1}$$

where c indicates the size of a sliding window and

$$p(w_{t+j}|w_t) = \frac{\exp(\mathbf{z}_t^T \mathbf{z}'_{t+j})}{\sum_{i=1}^{|\mathbf{W}|} \exp(\mathbf{z}_t^T \mathbf{z}'_i)}\,, \tag{3.2}$$

where w_i is the ith token in the document, the set $\mathbf{W}$ is the vocabulary set of all tokens, $\mathbf{z}_i$ is the target embedding of token w_i, and $\mathbf{z}'_i$ is the context embedding. When the index $t + j \notin \{1, \ldots, n\}$, we ignore it in (3.2). In Task# 1, we need both $\mathbf{z}_i$ and $\mathbf{z}'_i$ to be able to distinguish between the target and context roles of the same token [101]. In Task #2 (see Section 3.4.3) and Task #3 (see Section 3.4.4), we use only the context embedding $\mathbf{z}'_i$.

The loss function $\ell_{\text{Task 1}}$ for Task #1 is the negative-sampling objective function

$$\ell_{\text{Task 1}} = -\frac{1}{k+1} \left(\ln\left(\sigma(\mathbf{z}_t^T \mathbf{z}'_{t+j})\right) + \sum_{i=1}^{k} \mathbb{E}_{w_i \sim P_{\text{noise}}(w)} \left[\ln\left(\sigma(-\mathbf{z}_t^T \mathbf{z}'_i)\right) \right] \right)\,, \tag{3.3}$$

where k is the number of negative samples (i.e., token pairs that consist of a target token and a token from a noise distribution) per positive sample (i.e., token pairs that occur in the same sliding window), the sigmoid function σ is $\sigma(x) = \frac{1}{1+\exp(-x)}$, and $P_{\text{noise}}(\cdot)$ is a noise distribution. We obtain negative samples of word pairs from the noise distribution [101], whose name comes from the idea of noise-contrastive estimation (NCE) [61]. A good model should distinguish between data and noise. We use the same noise distribution as in Skip-Gram [101]:

$$P_{\text{noise}}(w) = \left(\frac{U(w)}{\sum_{i \in \mathbf{W}} U(i)} \right)^{3/4}\,, \tag{3.4}$$

where $U(w)$ denotes the number of appearances of a token w in the training corpus. Minimizing $\ell_{\text{Task 1}}$ approximates the maximization of the mean log probability (3.1).

In practice, when discussing political affairs, they are usually described by multiple words, namely, phrases. We use AutoPhrase [137] to detect phrases in our data sets, and treat them as tokens as well.

We refer to Task #1 as our **Baseline PEM** model, and we call it the "Skip-Gram model" when we use it on its own. We use the same hyperparameter settings as in the default settings in the original Skip-Gram model [101].

3.4.3 Task #2: Polarity Preservation

In Task #2, our goal is for the polarity component of our embeddings to capture reasonable polarity information. The finest granularity of the polarity labels that we can automatically and reliably obtain in large enough numbers are at the level of social-media accounts. We assume that every politician has consistent political views during our observation time (the years 2019 and 2020), and we assign polarity labels to their tweets based on their self-identified party affiliations. We thereby use account-level labels to guide the polarity-score learning of entities and hashtags.

A simple approach is to use the mean polarity score of all entities to estimate the polarity score of a text corpus. However, this approach does not consider the heterogeneous importance levels of entities. When considering political tendencies, some entities (e.g., "pro-choice") are more informative than others (e.g., "plan"). Therefore, we calculate a weighted average of entity polarities in each tweet, with weights that come from attention.

Suppose that we are given a sentence with n tokens (i.e., words, phrases, hashtags, mentions, emoji, and so on) that are embedded as $\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_n$, where m of the n tokens are entities or hashtags. The set of indices of the m tokens is $\mathbf{I} = \{i_1, \ldots, i_m\}$ (with $m \leq n$). The polarity dimensions of the embeddings are

$$\mathbf{E}_P = [\mathbf{z}_{i_1}^{(p)}\,;\,\mathbf{z}_{i_2}^{(p)}\,;\,\cdots\,;\,\mathbf{z}_{i_m}^{(p)}] \in \mathbb{R}^{m\times 1}\,.$$

We use a standard self-attention mechanism [66], which proceeds as follows. We represent keys, values, and queries in a vector space. Each key has a corresponding value. Upon receiving a query, we evaluate similarities between the queries and the keys. We then estimate the value of a query as a weighted average of the values that correspond to the keys [148].

We vertically concatenate the sequence of the semantic (i.e., polarity-neutral) components of the entities' and hashtags' embeddings and write

$$\mathbf{E}_S = [\mathbf{z}_{i_1}^{(s)}\,;\,\mathbf{z}_{i_2}^{(s)}\,;\,\cdots\,;\,\mathbf{z}_{i_m}^{(s)}] \in \mathbb{R}^{m\times d}\,,$$

where the key $\mathbf{K}$ and the query $\mathbf{Q}$ are different linear transformations of $\mathbf{E}_S$. That is,

$$\mathbf{K} = \text{stopgrad}\,(\mathbf{E}_S)\,\mathbf{W}_K\,, \quad \mathbf{Q} = \text{stopgrad}\,(\mathbf{E}_S)\,\mathbf{W}_Q\,,$$

where stopgrad is a stop gradient (so $\mathbf{E}_S$ is not updated by back-propagation of the attention component) and $\mathbf{W}_K, \mathbf{W}_Q \in \mathbb{R}^{d\times d}$ are weight matrices. We calculate the attention vector $\boldsymbol{\alpha} \in \mathbb{R}^{m\times 1}$, which includes an attention score for each entity in a tweet, using the standard softmax function:

$$\boldsymbol{\alpha} = \text{Att}(\mathbf{Q}, \mathbf{K}) = \text{softmax}\left(\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{m}}\right)\cdot \mathbf{1}_{m\times 1}\right)\,, \tag{3.5}$$

where the ith component of the softmax function is

$$\text{softmax}(x_i) = \frac{e^{x_i}}{\sum_{k=1}^{m} e^{x_k}}$$

and $\mathbf{1}_{m\times 1}$ is a vector of 1 entries.

Each tweet's polarity score $\tilde{\mathbf{z}}^{(p)}$ is then

$$\tilde{\mathbf{z}}^{(p)} = \boldsymbol{\alpha}^T \mathbf{E}_P \in \mathbb{R}^{1\times 1}\,. \tag{3.6}$$

Suppose that there are N tweets in total and that tweet j has the associated label $l_j \in \{-1, 1\}$, where -1 signifies that the tweet is by a politician from the Democratic party and 1 signifies that the tweet is by a politician from the Republican party. (We only consider politicians with a party affiliation.) We infer polarity scores $\{\tilde{\mathbf{z}}_1^{(p)}, \tilde{\mathbf{z}}_2^{(p)}, \ldots, \tilde{\mathbf{z}}_N^{(p)}\}$ for each tweet and then use a hinge loss with the margin parameter $\gamma > 0$ as our objective function. Specifically, we set $\gamma = 1$ and write the loss for Task #2 as

$$\ell_{\text{Task 2}} = \frac{1}{N}\sum_{j=1}^{N}\left(\max\left\{0, \gamma - l_j \tilde{\mathbf{z}}_j^{(p)}\right\}\right)\,. \tag{3.7}$$

When we use Task #1 and Task #2, we say that we are using our **Polarized PEM** model.

3.4.4 Task #3: Independence Enforcement

In Task #3, we encourage the semantic component $\mathbf{z}^{(s)}$ to be polarity-neutral, and we thereby force the political-polarity component $\mathbf{z}^{(p)}$ to capture polarity more accurately. We use an adversarial framework to achieve this goal. We alternately train two competing objectives: (1) learn a high-quality embedding $\mathbf{z}$ that preserves both context and polarity; and (2) learn a semantic embedding $\mathbf{z}^{(s)}$ that is not able to infer a tweet's polarity. Let E denote the first objective, which combines Task #1 and Task #2 and controls the quality of our embedding. The loss function ℓ_E of the first objective is

$$\ell_E = \ell_{\text{Task 1}} + \ell_{\text{Task 2}} \,. \tag{3.8}$$

Let D denote the second objective, which is a discriminator that attempts to use a semantic embedding for polarity classification. We start training by running the objective E because our discriminator makes sense only if our embedding is meaningful.

We apply the attention mechanism that we used in Task #2 (for aggregate token-level semantic embeddings) to a tweet-level semantic embedding. We use the weighted average $\tilde{\mathbf{z}}^{(s)} = \boldsymbol{\alpha}^T \mathbf{E}_S \in \mathbb{R}^d$ of the semantic dimensions of a tweet's tokens as our tweet-level semantic embedding. The $\mathbf{W}_K$ and $\mathbf{W}_Q$ functions in Task #3 are different than those in Task #2. We use the discriminator D to discern political-party labels from $\tilde{\mathbf{z}}^{(s)}$. The discriminator is a standard two-layer multilayer perceptron (MLP) classifier that infers a class label 0 for liberal-leaning tokens and a class label 1 for conservative-leaning tokens. Between these two layers, we set the number of elements in the output of each hidden layer to $d_{\text{MLP}} = 100$. We use a binary cross-entropy loss ℓ_D. The ground-truth labels of the tweets are $\mathbf{Y} = \{y_1, \ldots, y_N\} \in \{0,1\}^N$ and the inferred polarity scores are $\hat{\mathbf{Y}} = \{\hat{y}_1, \ldots, \hat{y}_N\}$. The output label of tweet i is

$$\hat{y}_i = D(\tilde{z}^{(s)}) = \sigma\left(\text{MLP}(\tilde{z}^{(s)})\right) \in [0,1] \,, \tag{3.9}$$

where σ is the sigmoid function. The discriminator loss is the binary cross entropy

$$\ell_D = -\frac{1}{N} \sum_{i=1}^{N} \left(y_i \ln(\hat{y}_i) + (1-y_i)\ln(1-\hat{y}_i)\right) \,. \tag{3.10}$$

The encoder E seeks to make ℓ_D large enough so that $\mathbf{z}^{(s)}$ tends to ignore political polarity. The discriminator D seeks to make ℓ_D small enough to be a stronger discriminator. To balance these goals, we use an adversarial framework. The training objective for all tasks together is

$$\ell_{\text{Task } 3} = \min_E \max_D \left(\ell(E, D)\right) = \min_E \max_D \left(\ell_E - \lambda \ell_D\right) \,. \tag{3.11}$$

We always train Task #3 together with Tasks #1 and #2. When we train all three tasks together, it is referred as the **Complete PEM** model.

3.4.5 Joint Training

In Algorithm 1, we present our adversarial framework for our **Complete PEM** model. An adversarial framework trains two neural networks together so that they counteract each other [27, 54]. The quantity θ_E denotes all of the parameters in Tasks #1 and #2, including all of the embedding weights $\mathbf{Z}$, the attention weights, and so on. The quantity θ_D, which we use only in Task #3, denotes the set of discriminator parameters. Each batch that we input into our **PEM** model has data from 16 tweets.

We learn all parameters in θ_E and θ_D during training, but we need to determine the hyperparameter λ. In our experiments, we examined $\lambda = 0.01$, $\lambda = 0.1$, $\lambda = 1$, and $\lambda = 10$. Of these values, our **Complete PEM** model performs the best for $\lambda = 0.1$, so we use $\lambda = 0.1$. When applying the **PEM** model to another data set, one should carefully select a suitable value of λ.

In each phase (i.e., either training θ_D or training θ_E), we stop training right after we first observe a drop in the F_1 score (which is is the harmonic mean of precision and recall) in the validation set. (Such a performance drop can be an indication of overfitting [19].) We then use the parameter values from just before the performance drop and proceed to the next phase.

Algorithm 1 Complete PEM: Learning algorithm

```
procedure LearnEmbedding(Iter)
    Z ← initialize the embeddings
    Initialize the parameter $\lambda > 0$
    for $i = 1, \ldots, \text{Iter}$ do
        while not converged do                              ▷ train $\theta_E$, fix $\theta_D$
            sample from tweets
            $\ell_E \leftarrow \ell_{\text{Task 1}} + \ell_{\text{Task 2}}$
            $\ell(E, D) \leftarrow \ell_E - \lambda \ell_D$
            update $\theta_E$ to minimize $\ell(E, D)$
        end while
        while not converged do                              ▷ train $\theta_D$, fix $\theta_E$
            sample from tweets
            $\ell_D \leftarrow$ Discriminator loss
            update $\theta_D$ to minimize $\ell_D$
        end while
    end for
    return Z                                                ▷ the learned embedding
end procedure
```

3.5 Experiments

In this study, we discuss the experimental results of our **PEM** model. Our discussion include how to construct our data sets, what do our polarity embeddings look like, and some downstream tasks that we can apply our embeddings to.

3.5.1 Data Sets

We start by collecting a list of Twitter accounts, including 585 accounts of legislators in the 115th and 116th Congresses,[2] the accounts of 8 well-known news outlets (see Table 3.1), and the accounts of President Barack Obama, President Donald Trump, and their Cabinet members at the time (3 March 2019) that we first collected the data. Our data set consists of (1) the most recent 3,200 tweets of each account that we collected on 3 March 2019 and (2) the tweets of these accounts that were posted between 1 January 2020 and 25 November 2020.

We select the news outlets from those with the most voters (i.e., participants who label the political polarity of news outlets on the AllSides Media Bias Ratings (see `https://www.allsides.com/media-bias/media-bias-ratings`). Previous studies have inferred the political polarities of news outlets from their content [21, 58], and we seek to examine whether or not our model can also reveal political polarities. The available political labels in the AllSides Media Bias Ratings are "liberal", "somewhat liberal", "neutral", "somewhat conservative", and "conservative". We use the three liberal news outlets with the most votes, the three conservative news outlets with the most votes, and the neutral news outlet with the most votes. We checked manually that the polarities of the Twitter accounts of these news outlets are consistent with the labels that we obtained from the AllSides Media Bias Ratings. When a news outlet has multiple Twitter accounts (e.g., `@cnn` and `@cnnpolitics`), we use the account with the most followers in early February 2020. On 10 February 2020, we finished collecting and sorting the media data.

[2]See `https://www.congress.gov/members`.

We split the politicians' tweets (of which there are more than 1,000,000 in total) into training, validation, and testing sets in the ratio 8:1:1. We also use the tweets of the news outlets and those of the unobserved accounts as testing sets.

We also test our embedding on three existing data sets: the Election2020 data set [22], which has 965,620,919 tweets that were collected hourly between March 2020 and December 2020; a Parler data set from 6 Jan 2021 that has 1,384,579 posts;[3] and the TIMME data set [158], which includes 2,975 Twitter accounts with location information and self-identified political-polarity labels (either Democratic or Republican). These Twitter accounts are not run by politicians and are never in a training data set. We thus refer to them as "unobserved accounts". We have access to the most recent 3,200 tweets in each Twitter account's timeline; we keep the tweets that they posted in 2020.

Table 3.1: The selected news outlets and their political polarities. The label "L" denotes a liberal-leaning outlet, "C" denotes a conservative-leaning outlet, and "N" denotes a neutral outlet. These labels come from the AllSides Media Bias Ratings (see https://www.allsides.com/media-bias/media-bias-ratings).

Twitter Account	News Outlet	Polarity
`@nytimes`	*The New York Times*	L
`@guardiannews`	*Guardian News*	L
`@cnn`	*CNN*	L
`@csmonitor`	*The Christian Science Monitor*	N
`@amspectator`	*The American Spectator*	C
`@foxnewsopinion`	*Fox News Opinion*	C
`@nro`	*National Review*	C

[3]This data set is available at the repository https://gist.github.com/wfellis/94e5695eb514bd3ad372d6bc56d6c3c8.

3.5.2 Entity Identification

We use the union of the set of entities from three main sources to identify potential entities while training.

To detect nouns, we consider all nouns and proper nouns from parts-of-speech (POS) tagging[4] to be reasonable entities.

To detect phrases that act as nouns, we use AutoPhrase (version 1.7) [137] to learn a set of phrases from all politicians' tweets in our data. We then use this set of phrases when tokenizing all employed data sets. AutoPhrase assigns a score in the interval $[0, 1]$ to each potential phrase, where a higher score indicates a greater likelihood to be a reasonable phrase. After looking at the results, we manually choose a threshold of 0.8, and we deem all multi-word noun phrases whose scores are at least this threshold to be of sufficiently high quality.

To detect special terms that represent entities that may not yet be part of standard English, we apply TagMe (version 0.1.3) [44] to our training set to include named entities that we are able to link to a Wikipedia page.

3.5.3 Results

We train our **PEM** to obtain the two-component polarity-aware embeddings of the named entities and hashtags in our data sets.

3.5.3.1 Polarity Component

We compute token-level polarity scores by examining the polarity component $\mathbf{z}^{(p)}$ of each embedding. We transform all tokens except mentions into lower-case versions. We do this because Twitter handles (i.e., user names) are case-sensitive, but upper-case and lower-case letters have the same meaning (and thus can be used as alternatives to each other) for other

[4]See `https://www.nltk.org/api/nltk.tag.html`.

entities (including hashtags).

According to our results, of the entities and hashtags that politicians used in our data (which we collected in 2019 and 2020), the ones with the strongest liberal polarities are **#trumpcare**, **#actonclimate**, **#forthepeople**, **#getcovered**, and **#goptaxscam**. The entities and hashtags with the strongest conservative polarities are **#va10**, **#utpol**, **#ia03**, **#tcot**, and **#wa04**.

Our results illustrate that hashtags that refer to electoral districts can be strongly conservative-leaning. Politicians with different political leanings may use hashtags in different ways, and examining a hashtag that is associated with an electoral district is a good way to illustrate this. Additionally, conservative politicians may use a particular non-germane hashtag for certain content more often than liberal politicians. For example, some tweets that used **#va10** contributed to a discussion of a #VA10 forum that was hosted by the Republican party in Fauquier County (`@fauquiergop`).

In Figure 3.3, we show our embedding results for the 1,000 most-frequent entities and hashtags and for a few highlighted ones that we select manually. To facilitate visualization, the vertical axis is a 1D t-distributed stochastic neighbor embedding (t-SNE) values [94]. In theory, words with particularly close semantic meanings are near each other along this axis. In our embedding results, hashtags are more likely than other tokens to capture a clear political polarity.

Some of our observations are unsurprising. For example, terms that are related to "**pro-life**" are typically conservative-leaning, whereas terms that are related to "**pro-choice**" are typically liberal-leaning.

Other observations are more nuanced. For example, liberal-leaning Twitter accounts sometimes use text that one is likely to associate more with conservative-leaning views, and vice versa. The embeddings of "**trump**" and "**obama**" give one pair of examples, and the hashtags **#trumpcare** and **#obamacare** give another. Hashtags without semantic context can also appear in tweets. Another interesting observation is that **#blacklivesmatter** and **#alllivesmatter** are both liberal-leaning. In [50], it was pointed out that **#alllivesmatter**

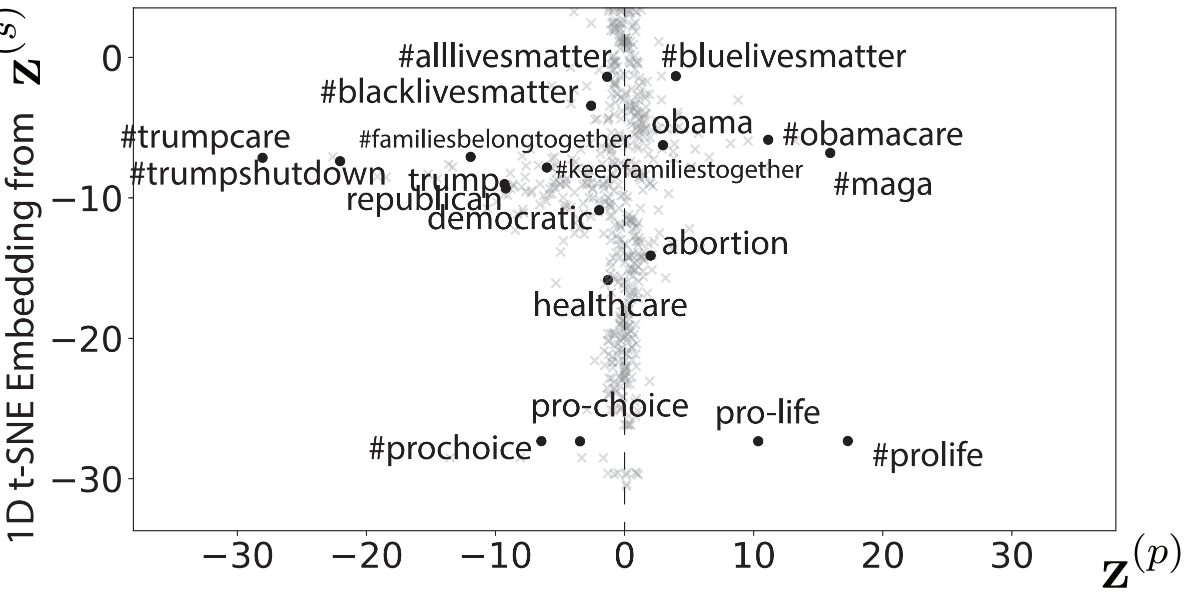
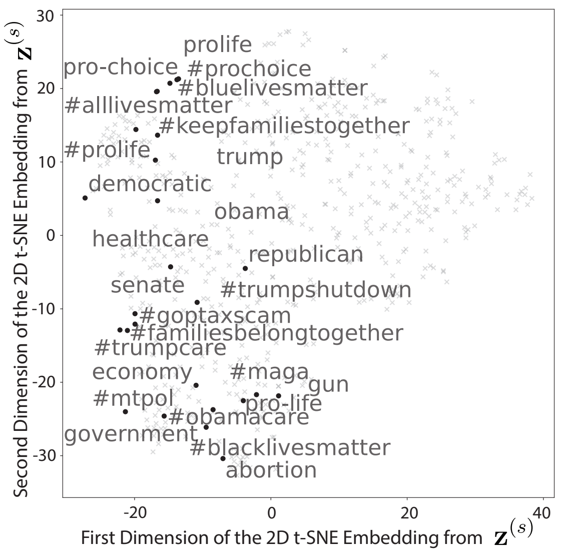


Figure 3.3: Visualization of the political polarities in our embedding results. The horizontal axis gives the values of the polarity score $\mathbf{z}^{(p)}$, and the vertical axis is a 1D t-SNE value (which we use to facilitate visualization) that we calculate from the semantic embedding $\mathbf{z}^{(s)}$.

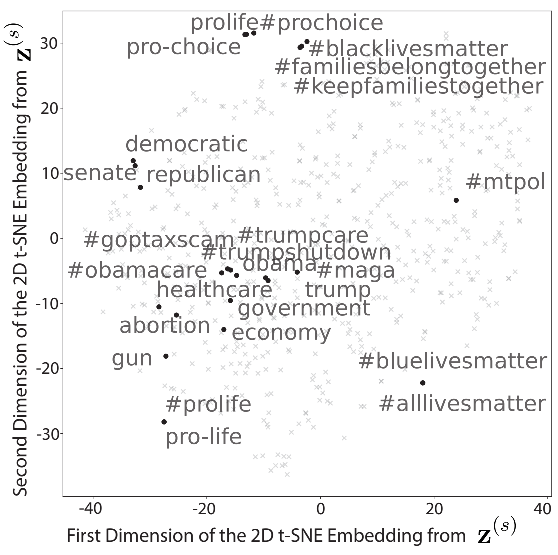
was used as a counterprotest hashtag between August 2014 and August 2015. This observation helps illustrate that the polarities of tokens can change with time. Nowadays, **#bluelivesmatter** is used more than **#alllivesmatter** as an antonym of **#blacklivesmatter** in practice (in the sense of having a similar semantic meaning but opposite political polarity). Additionally, **#alllivesmatter** now appears commonly in topics such as animal rights.

3.5.3.2 Semantic Components

To demonstrate the quality of the semantic components $\mathbf{z}^{(s)}$, we calculate the cosine similarity of the embedding vectors of the tokens. Our results appear to be reasonable. For example, we observe that the closest token to "**gun**" is "**firearm**" and that the closest token to "**healthcare**" is "**care**". The t-SNE values from our **Polarized PEM** model and **Complete PEM** model also suggest that these semantic components have reasonable quality.



(a) **Complete PEM** semantic components.



(b) **Polarized PEM** semantic components.

Figure 3.4: Visualization of the semantic components of our (a) **Complete** and (b) **Polarized PEM** embeddings. We project these components onto a plane by calculating t-SNE values. Both results are reasonable, but the **Polarized PEM** results tend to encourage semantically-related words to be closer to each other. For example, **#familiesbelongtogether** and **#keepfamiliestogether** are used similarly in practice and they are close to each other in the embedding from our **Polarized PEM** model.

In Figure 3.4a, we plot the results of calculating t-SNE values to project the semantic dimensions of the most-frequent 600 tokens and several manually-selected tokens from our **Complete PEM** embeddings onto a plane. In Figure 3.4b, we show the t-SNE values for our **Polarized PEM** embeddings. These plots illustrate similarities in the semantic meanings of these tokens. For example, we observe that **#AllLivesMatter** and **#BlueLivesMatter** have similar meanings. By comparing Figures 3.4a and 3.4b, it seems that the semantic components of our **Polarized PEM** embeddings may be slightly more reasonable than those of our **Complete PEM** embeddings.

3.5.3.3 Account-Level Case Studies

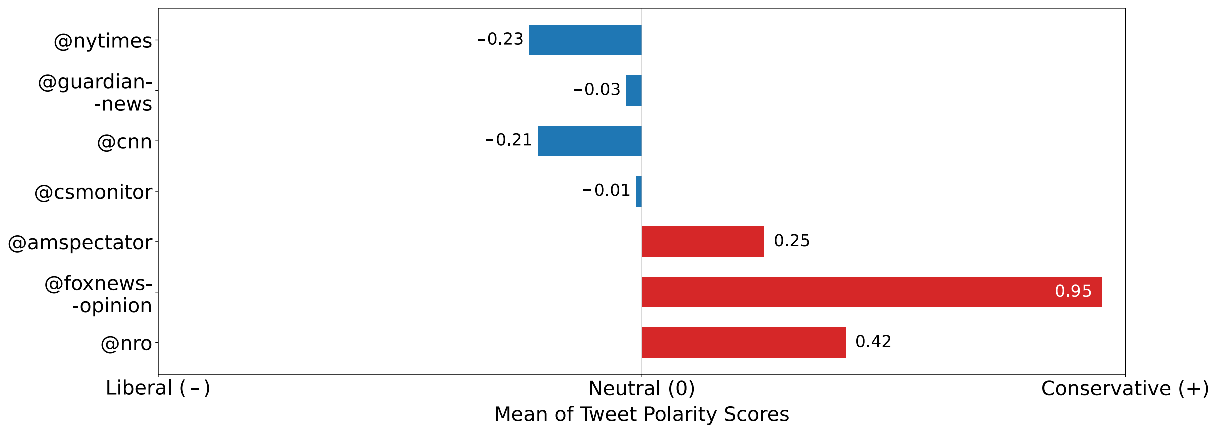


Figure 3.5: Our estimates of the political polarities of news outlets based on their most recent 3,200 tweets. We collected these tweets starting on 3 March 2019.

We compute a Twitter account's political polarity by calculating the mean of the polarity scores of all of its tweets. Suppose that an account posted N tweets. The i^{th} tweet consists of n tokens, with embeddings $\{\mathbf{z}_1, \ldots, \mathbf{z}_n\}$ and polarity scores $\{\mathbf{z}_1^{(p)}, \ldots, \mathbf{z}_n^{(p)}\}$. The tweet-level polarity score of this tweet is $b_i = (\sum_{j=1}^{n} \mathbf{z}_j^{(p)})/n$. We estimate the overall polarity score of the account to be $b = (\sum_{i=1}^{N} b_i)/N$. If $b_i < 0$, we regard account i as liberal-leaning; if $b_i > 0$, we regard it as conservative-leaning; if $b_i = 0$, we regard it as neutral. We show our results (which seem reasonable) in Figure 3.5. We plot liberal-leaning accounts in blue and conservative-leaning accounts in red.

Some previous research [59, 147] on relationships (e.g., following and retweeting relationships) between Twitter accounts has inferred clearer polarities in news outlets than what we obtain using our approach. This suggests that interactions may be more helpful than text itself at identifying the political polarities of Twitter accounts.

3.5.3.4 Illustrations of Estimating Tweet Polarities with the Attention Mechanism

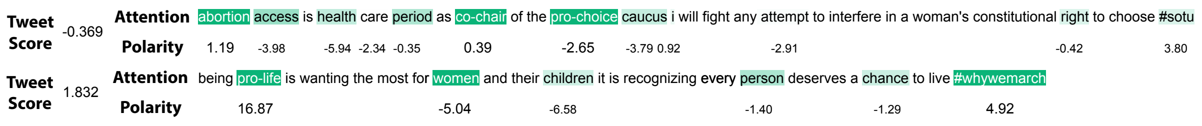

Tweet Score -0.369
Attention abortion access is health care period as co-chair of the pro-choice caucus i will fight any attempt to interfere in a woman's constitutional right to choose #sotu
Polarity 1.19 -3.98 -5.94 -2.34 -0.35 0.39 -2.65 -3.79 0.92 -2.91 -0.42 3.80
Tweet Score 1.832
Attention being pro-life is wanting the most for women and their children it is recognizing every person deserves a chance to live #whywemarch
Polarity 16.87 -5.04 -6.58 -1.40 -1.29 4.92

Figure 3.6: Illustrations of estimating tweet polarities using an attention mechanism. We show the weights from our **Complete PEM** model in green, where darker shades signify greater importance levels. We show the polarity scores underneath the entities and hashtags.

See Figure 3.6 for examples of our **Complete PEM** model's attention weights and polarity scores. Both the attention weights and the polarity scores appear to be reasonable.

3.5.3.5 An Ablation Study of the Attention Mechanism

We summarize the performance of the three versions of our **PEM** model in Table 3.2. The left column gives our classification results when we use an attention mechanism. Recall that our **Baseline** model does not use an attention mechanism. In models with an attention mechanism, we use the score that we infer from Task #2, which calculates a weighted average of the tokens' political-polarity component $\{\mathbf{z}^{(p)}\}$. In the right column, we show the accuracy and F_1 scores when we use the mean value of the elements of $\{\mathbf{z}^{(p)}\}$. Recall that we interpret tweets with negative scores as liberal and tweets with positive scores as conservative.

The results in Table 3.2 suggest that Task #2 alone can successfully capture polarity information, but introducing Task #3 to enhance the independence of the semantic and polarity components can improve inference of the political-polarity component $\mathbf{z}^{(p)}$. However, forcing $\mathbf{z}^{(s)}$ to be polarity-neutral makes it harder to preserve accurate semantic information. (See Figures 3.4a and 3.4b.) This illustrates why our **Complete PEM** model does not always outperform our **Polarized PEM** model.

3.5.4 Results on a Few Downstream Tasks

We illustrate that our embeddings are reliable and useful for several downstream tasks. These downstream tasks include estimating some accounts' political polarity, showing how public opinions change over time, detecting potential political-polarity in other real-world data sets, and so on.

3.5.4.1 Classification Results

First, we discuss the classification results of our **Polarized** and **Complete PEM** models.

We select 10% of the politicians' tweets (there are 127,143 such tweets) uniformly at random and withhold these tweets as the testing set for Table 3.2. We select another 10% of the tweets, which we also choose uniformly at random, as a validation set. We use the remaining 80% of the tweets (i.e., 1,017,137 tweets) as our training set. We train all models (see Table 3.2 and Table 3.3) on the same training set.

In Table 3.2, we show the performance of the models on the testing set. We perform the tweet-level classification task on the withheld tweets of the politicians. We never include these tweets in the training set. We perform the account-level classification task on the accounts of all politicians with tweets in the testing set. For a given account, we use its tweets in the testing set to infer its political score by calculating the mean polarity score of all of its tweets.

In Table 3.3, we show the tweet-level and account-level classification performance levels for the unobserved accounts. (See Section 3.5.1 for a description of these accounts.)

We use the Skip-Gram and GloVe embeddings as baselines. For each of these embeddings (which we do not adjust), we use the same MLP classifier that we use as a discriminator in

Table 3.2: The classification performance on the withheld tweets of politicians and on the Twitter accounts of politicians. The subscript "no attn" signifies that we use the mean value of $\{\mathbf{z}^{(p)}\}$ directly (i.e., without applying an attention mechanism). Skip-Gram (i.e., the **Baseline PEM** model) and GloVe use a pretrained embedding with the same MLP binary classifier as in our discriminator. (To train this classifier, we use a training set that includes 80% of the politicians' tweets.) In each entry, we show the accuracy followed by the F_1 score. We show the best results for each column in bold. The names of our models are also in bold.

Model	**Tweet-Level Results (accuracy; F_1)**	**Account-Level Results (accuracy; F_1)**
Skip-Gram	0.7705; 0.7736	0.8769; 0.8797
GloVe	0.7438; 0.7453	0.8578; 0.8620
BERT$_{\text{base}}$	**0.8595; 0.8603**	**0.9965; 0.9968**
BERTweet	0.8399; 0.8435	0.9844; 0.9853
Polarized PEM$_{\text{no attn}}$	0.7681; 0.7682	0.9757; 0.9758
Complete PEM$_{\text{no attn}}$	0.7991; 0.7994	0.9827; 0.9827
Polarized PEM	0.8339; 0.8337	0.9861; 0.9872
Complete PEM	0.8338; 0.8330	0.9931; 0.9936

Task #3 and train the MLP classifiers on our training set until they converge. We fine-tune the transformer classifiers BERT$_{\text{base}}$ [41] and BERTweet [106] (which uses the BERT$_{\text{base}}$ model configuration and is trained using RoBERTa-style pretraining) on our training set as baselines. We use the uncased (i.e., ignoring capitalization) version of BERT$_{\text{base}}$; the classifier BERTweet separates lower-case and upper-case letters. We use the fine-tuned transformers to classify the tweets of politicians (see Table 3.2) and the tweets of the unobserved accounts (see Table 3.3).

For the model variants that do not incorporate attention, we calculate each polarity score by computing the mean values of the polarity components $\mathbf{z}^{(p)}$ of the entities and hashtags. We compute the polarities of accounts in the same way as in our examples with news outlets (see Section 3.5.3.3).

By comparing Table 3.2 and Table 3.3, we conclude that our models perform better

Table 3.3: The classification performance on the unobserved accounts. We never include tweets from these accounts in a training data set. In each entry, we show the accuracy followed by the F_1 score. We show the best results for each column in bold. The names of our models are also in bold.

Model	**Tweet-Level Results (accuracy; F_1)**	**Account-Level Results (accuracy; F_1)**
Skip-Gram	0.5822; 0.5636	0.6660; 0.6604
GloVe	0.5680; 0.5491	0.6486; 0.6372
BERT$_{\text{base}}$	**0.6541**; 0.6280	0.7234; 0.7218
BERTweet	0.6284; 0.6486	0.7836; 0.7778
Polarized PEM$_{\text{no attn}}$	0.6066; 0.6244	0.8157; 0.8196
Complete PEM$_{\text{no attn}}$	0.6061; 0.6258	0.8494; 0.8475
Polarized PEM	0.6308; 0.6965	0.8493; 0.8758
Complete PEM	0.6479; **0.6987**	**0.8612**; **0.8870**

than the transformers (BERT$_{\text{base}}$ and BERTweet) on the unobserved accounts. Possible reasons include the following:

1. Our polarity score can take any real value, so it can highlight extremists and exploit extreme tweets that help expose an account's polarity. BERT$_{\text{base}}$ only allows polarity values between 0 and 1.

2. Models, such as the transformers, with many parameters can suffer from severe overfitting problems, especially when a training set is too small. In Section 3.6, we discuss potential drawbacks of a training data set that includes tweets only by politicians.

3.5.4.2 Classification Results using Only Semantic Components

To demonstrate that including Task #3 allows the polarity component $\mathbf{z}^{(p)}$ to capture more political information and makes the semantic components $\mathbf{z}^{(s)}$ more politically neutral, we conduct an experiment in which we use only the semantic components of the tokens for a classification task. Specifically, we examine account-level classification of the politicians' withheld tweets (see Table 3.4).

In the left column of Table 3.4, we show our account-level classification results using only $\mathbf{z}^{(s)}$. We obtain these results by training a discriminator with the same architecture as in **Task #3**. We train it on our training set (which has 80% of the politicians' tweets) until the classifier converges on our validation set (which has 10% of politicians' tweets). We then use it to classify tweets in the testing set (which has 10% of politicians' tweets).

Of our classification tasks in Section 3.5.4.1, doing account-level classification based on the politicians' tweets in the testing set is the least challenging one. For more challenging classification tasks, such as the classification of the tweets of the unobserved accounts, the accuracies that we obtain by using SKIP-GRAM (i.e., the **Baseline PEM** model), the**Polarized PEM** model, and the **Complete PEM** model are 0.5701, 0.5809, and 0.5756, respectively. Their accuracies for classifying the unobserved accounts are 0.6450, 0.6624, and 0.6551, respectively. These numerical values suggest that their performance levels are similar on these tasks.

Table 3.4: The account-level classification performance on the politicians' withheld tweets in our testing set. We never include these tweets in our training data set, but our training set does include other tweets by the accounts that posted these tweets. In each entry, we show the accuracy followed by the F_1 score. We show the best results for each column in bold. The names of our models are also in bold. The SKIP-GRAM row indicates our **Baseline PEM** results.

Model	**Results Based on $\mathbf{z}^{(s)}$ (accuracy; F_1)**	**Results Based on $\mathbf{z}^{(p)}$ (accuracy; F_1)**
SKIP-GRAM	0.8394; 0.8451	0.8457; 0.8503
Polarized PEM	**0.8994; 0.9008**	0.9861; 0.9872
Complete PEM	0.8111; 0.8204	**0.9931; 0.9936**

The results in Table 3.4 suggest that the design of our **Complete PEM** model helps encourage political information to be in the polarity component $\mathbf{z}^{(p)}$, rather than in the semantic components $\mathbf{z}^{(s)}$.

3.5.4.3 Polarity Distribution of Politicians

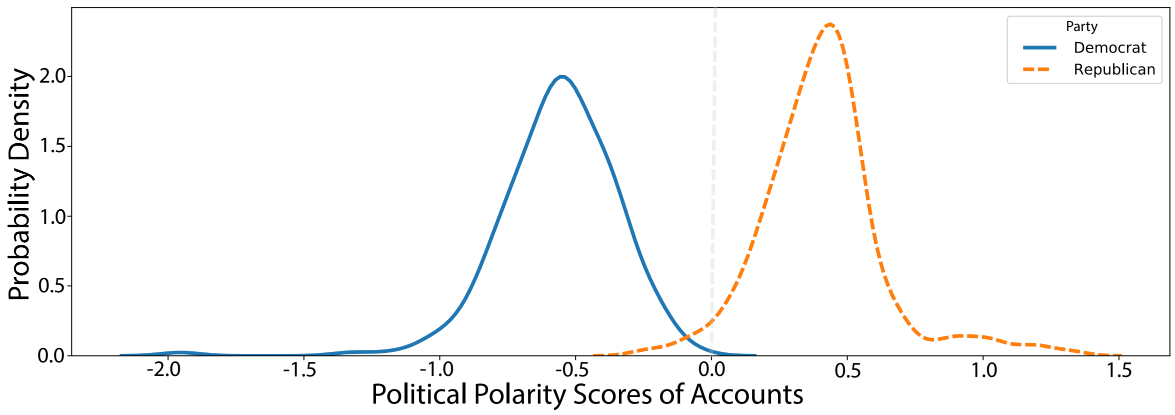


Figure 3.7: Probability densities of the polarity scores of the Twitter accounts of politicians.

We use the same approach as in Section 3.5.4.1 to estimate the polarity scores of the Twitter accounts of politicians. We plot the associated probability densities for both Democrats and Republicans in Figure 3.7, and we observe stark polarization.

3.5.4.4 Temporal Variation of Political Polarities

We now examine temporal changes in the inferred political polarities of the 49,428 Twitter accounts in the TIMME data set [158] that tweeted in 2020. To examine such temporal variation, we chunk the tweets from 2020 of each of these accounts in 7-day intervals starting from 1 January and examine trends over time. (The final interval is cut off and is hence shorter.)

We use the same approach as in Section 3.5.4.1 to infer tweet-level and account-level polarities. As we can see in Figure 3.8, our embedding results illustrate plausible trends on Twitter. Many liberal-leaning accounts were active starting in the week of the murder of George Floyd. As the week of the U.S. presidential election approached, people were using Twitter more actively, and then discussions of the election seemed to recede after it was over. Based on our results, we also suspect that there may be more liberal-leaning accounts than conservative-leaning accounts on Twitter.

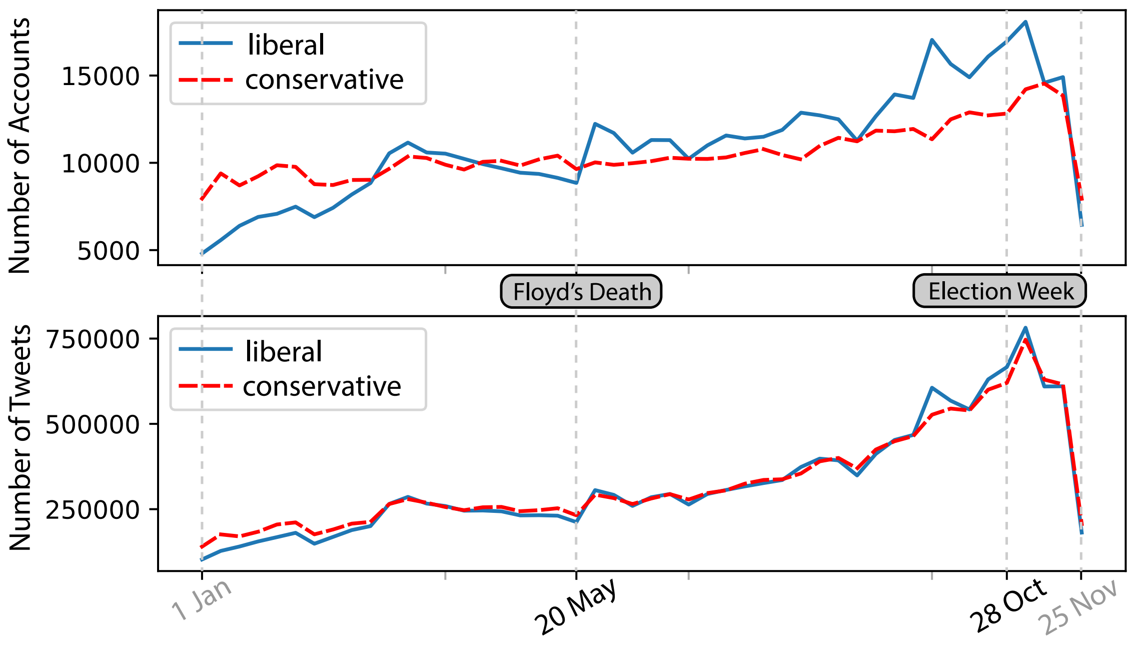


Figure 3.8: Weekly trends of liberal and conservative tweets on Twitter in 2020. We plot these trends at both (top) the account level and (bottom) the tweet level. The week of George Floyd's murder began on 20 May 2020. The week of the 2020 United States presidential election began on 28 October.

3.5.4.5 Geographic Distribution of Political Polarities

The TIMME data set [158] has 51,060 accounts with self-reported geographic locations in the United States. Using these locations, we examine the liberal versus conservative tendencies of tweets across the U.S. in 2020. We calculate the polarity of each Twitter account using the mean of the polarities of the tokens in its tweets; we show these account polarities geographically in Figure 3.9. We use the mean polarity of all accounts in a state (and the geographic regions Washington, D.C., Puerto Rico, and Guam) to calculate the state's polarity, and we then normalize the states' polarity scores $\mathbf{q} = \{q_1, \ldots, q_{53}\}$ to the interval $[-1, 1]$ by calculating $\hat{q}_i = (q_i - \frac{\sum_{j=1}^{53} q_j}{53}) / \max\{|q_1|, \ldots, |q_{53}|\}$. After this normalization, -1 is the most liberal score and $+1$ is the most conservative score. Our results are consistent with the tendencies that were reported in national polls for the 2020 U.S. election.[5]

[5]See https://www.realclearpolitics.com/epolls/2020/president/National.html.

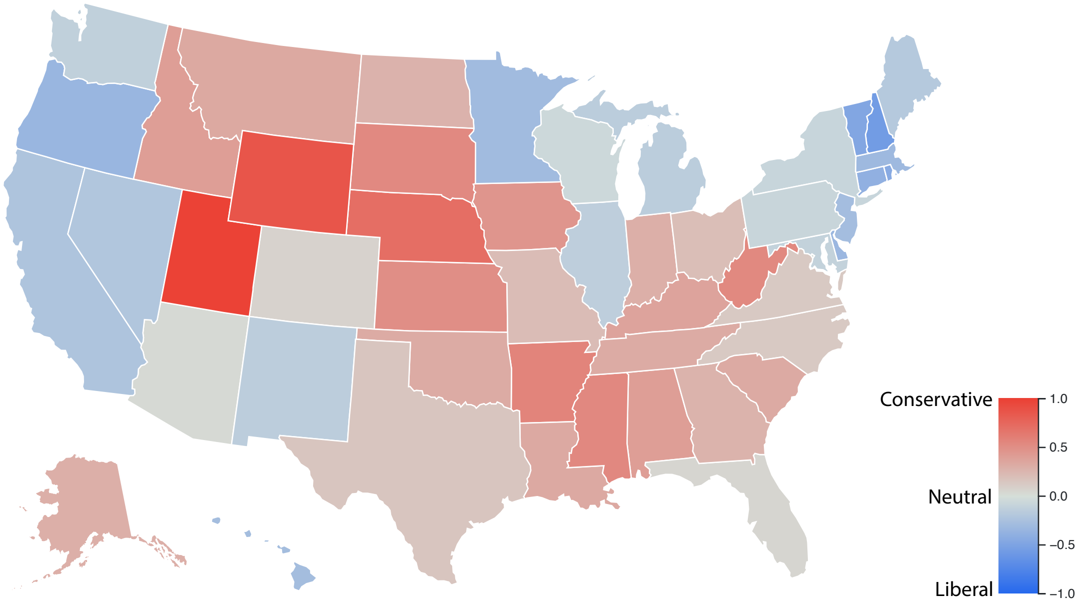


Figure 3.9: The mean polarity score of the Twitter accounts in each state (and the geographic regions Washington, D.C., Puerto Rico, and Guam) in the United States. We normalize the polarity scores to $[-1, 1]$.

3.5.4.6 Revealing Biases in Data Sets

We use the embedding results of our **Complete PEM** model to examine biases in data sets. In practice, using these results entails assuming that we can trust the polarities that we learn from the coarse-grained labels of the politicians' parties. Under this assumption, we find that the TIMME data set is politically neutral and that the ELECTION2020 data set [22] is somewhat liberal-leaning. In the ELECTION2020 data set, the mean polarity of the tweets in each week is liberal-leaning. Of the 119 keywords that were provided in Version 1 of this data set, there are 78 liberal-leaning keywords and 41 conservative-leaning keywords. Our embedding also suggests that posts on Parler tend to be more conservative than tweets on Twitter. In Figure 3.10, we plot the distributions of the polarities of the Twitter tweets and Parler posts. We compute these empirical probability densities using kernel density estimation (KDE) with a Gaussian kernel (i.e., the default setting) in the

SEABORN library [155].

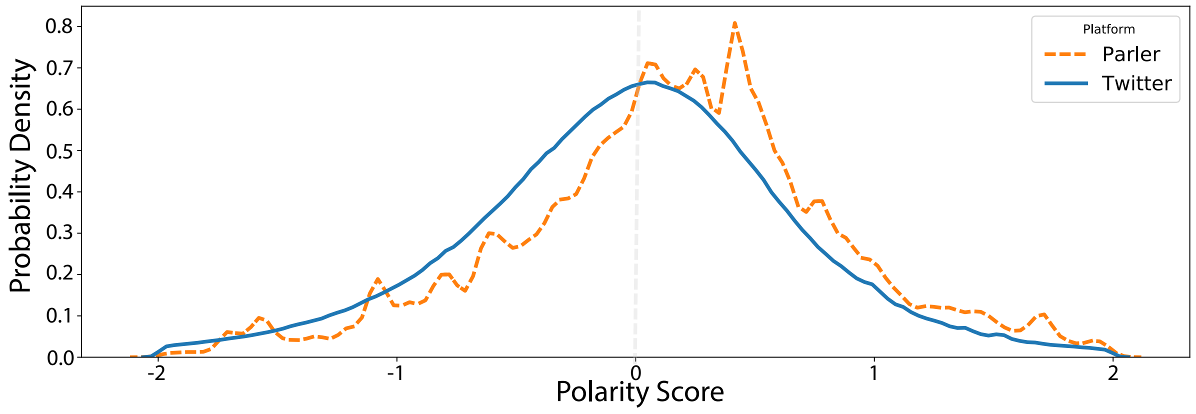


Figure 3.10: Distributions of polarity scores of Twitter tweets and Parler posts. The Twitter curve is smoother because the Twitter data set is much larger than the Parler data set.

3.5.5 Performance Robustness

In Table 3.2 and Table 3.3, we reported our best performance levels (from six different random seeds). We also want to examine the robustness of these performance levels. We use the same hyperparameter settings as before, but now we use 5-fold cross validation and different random seeds to initialize the models.

We still train the models on the politicians' tweets. However, instead of randomly using 80% of them as our training set, we now do a 5-fold cross validation. That is, we split the politicians' tweets evenly and uniformly at random into 5 sets that we select uniformly at random, and we withhold one set at a time as our validation and testing sets (with 10% each, with the tweets in them selected uniformly at random). None of the training sets are identical to the one that we used previously.

After training a model on the training set, we evaluate it on the testing data set of politicians. We then use the trained models to infer the polarities of the tweets from the unobserved accounts using the approaches in Table 3.3.

In Table 3.5, we report the means and standard deviations from our 5-fold cross validation. The results illustrate that the models' performance levels are robust, although the tweet-level

Table 3.5: The mean values and standard deviations for 5-fold cross validation of different models, which we initialize with different random seeds. We show the best results for each column in bold. The names of our models are also in bold.

Politicians' Accounts (Mean Value ± Standard Deviation)		
Model	**Tweet-Level Results (accuracy; F_1)**	**Account-Level Results (accuracy; F_1)**
Skip-Gram	0.7700 ± 0.0026 ; 0.7707 ± 0.0029	0.8833 ± 0.0113 ; 0.8996 ± 0.0100
GloVe	0.7231 ± 0.0039 ; 0.7319 ± 0.0035	0.8575 ± 0.0205 ; 0.8798 ± 0.0161
$\text{BERT}_{\text{base}}$	$\mathbf{0.8586} \pm 0.0006$; $\mathbf{0.8587} \pm 0.0006$	$\mathbf{0.9963} \pm 0.0034$; $\mathbf{0.9963} \pm 0.0034$
BERTweet	0.8337 ± 0.0010 ; 0.8327 ± 0.0010	0.9828 ± 0.0077 ; 0.9826 ± 0.0077
Polarized $\text{PEM}_{\text{no attn}}$	0.7691 ± 0.0011 ; 0.7665 ± 0.0011	0.9721 ± 0.0244 ; 0.9723 ± 0.0243
Complete $\text{PEM}_{\text{no attn}}$	0.7955 ± 0.0009 ; 0.7937 ± 0.0009	0.9805 ± 0.0169 ; 0.9811 ± 0.0167
Polarized PEM	0.8338 ± 0.0007 ; 0.8336 ± 0.0007	0.9841 ± 0.0030 ; 0.9845 ± 0.0030
Complete PEM	0.8332 ± 0.0006 ; 0.8327 ± 0.0006	0.9915 ± 0.0026 ; 0.9927 ± 0.0026
Unobserved Accounts (Mean Value ± Standard Deviation)		
Model	**Tweet-Level Results (accuracy; F_1)**	**Account-Level Results (accuracy; F_1)**
Skip-Gram	0.5822 ± 0.0007 ; 0.5635 ± 0.0008	0.6561 ± 0.0053 ; 0.6324 ± 0.0074
GloVe	0.5764 ± 0.0009 ; 0.5574 ± 0.0009	0.6387 ± 0.0073 ; 0.6222 ± 0.0099
$\text{BERT}_{\text{base}}$	0.6348 ± 0.0007 ; 0.6231 ± 0.0006	0.7182 ± 0.0078 ; 0.7149 ± 0.0072
BERTweet	0.6282 ± 0.0006 ; 0.6280 ± 0.0005	0.7752 ± 0.0176 ; 0.7695 ± 0.0173
Polarized $\text{PEM}_{\text{no attn}}$	0.6245 ± 0.0011 ; 0.6067 ± 0.0011	0.8062 ± 0.0191 ; 0.8105 ± 0.0182
Complete $\text{PEM}_{\text{no attn}}$	0.6259 ± 0.0014 ; 0.6063 ± 0.0015	0.8467 ± 0.0177 ; 0.8450 ± 0.0178
Polarized PEM	0.6284 ± 0.0023 ; 0.6865 ± 0.0020	0.8463 ± 0.0063 ; 0.8666 ± 0.0059
Complete PEM	$\mathbf{0.6472} \pm 0.0030$; $\mathbf{0.6907} \pm 0.0028$	$\mathbf{0.8550} \pm 0.0075$; $\mathbf{0.8814} \pm 0.0072$

performance levels are more robust than the account-level performance levels.

3.5.6 Bot Analysis

Our investigation does not account for the activity of automated accounts (i.e., bots). We use the verified Twitter accounts of politicians, so we assume that these are not bot accounts. However, bots are widespread on Twitter and other social media [45], We check for potential bots in our Twitter accounts and compare the inferred bot probabilities of these accounts with our inferred political polarities. We find that the probability that an account is a bot has little correlation with its political polarity.

To evaluate the probability that a Twitter account is a bot, we use Botometer (version 4) [131]. It has two options — universal and English — for the language that it employs for bot detection. The universal bot score is evaluated in a language-independent way, but the English bot score is more accurate for accounts that tweet primarily in English, so we use the English option.

There are many different types of Twitter bots (see `https://botometer.osome.iu.edu/faq`). For simplicity, we use only an overall bot score from Botomer. The score of a bot varies between 0 and 5, with larger scores signifying that an account is more likely to be a bot. In Figure 3.11, we show the probability densities of bot scores for politicians and ordinary Twitter accounts.

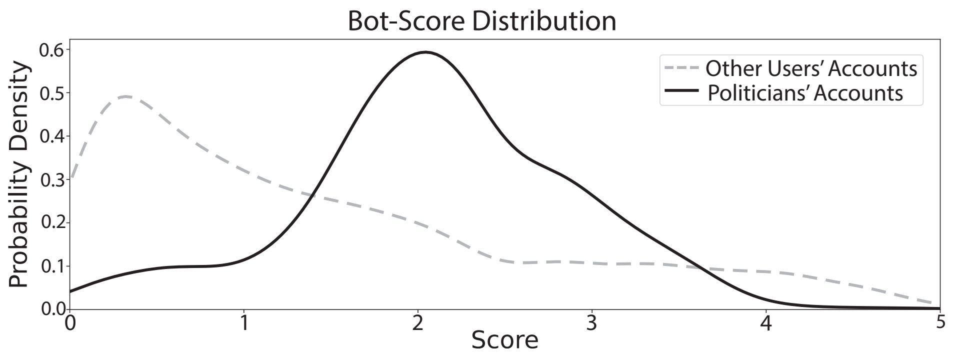
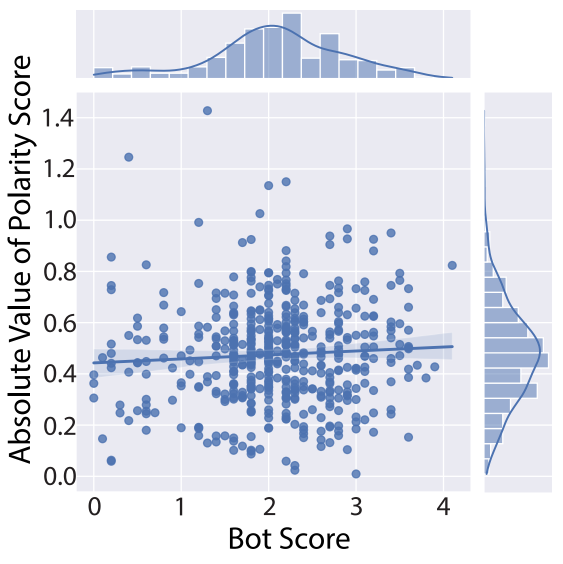


Figure 3.11: Probability densities of the bot scores of the Twitter accounts of politicians (solid curve) and all other Twitter accounts (dashed curve).

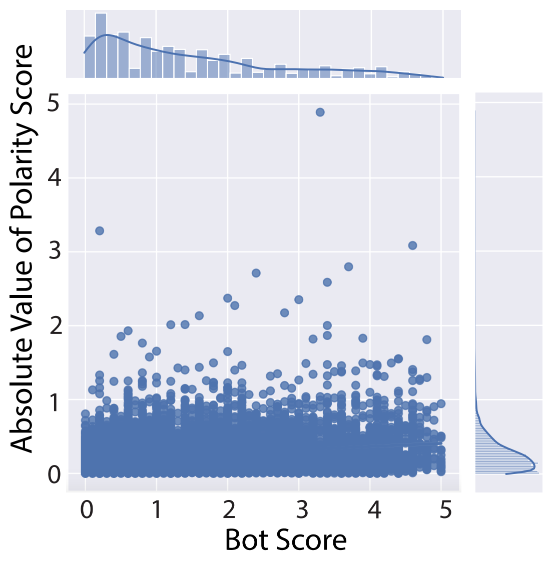
In Figure 3.12, we plot the distributions of the overall bot scores versus the absolute values of polarity scores (i.e., $|\{\mathbf{z}^{(p)}\}|$) for both politicians' Twitter accounts and ordinary Twitter accounts. The absolute values of the polarity scores indicate the extremeness of an account's content according to our **PEM** model.

3.5.7 Impact of Assigning Polarity Scores to Other Tokens

We use tokens other than hashtags and entities in our **PEM** model, but we have not assigned political polarities to them. We feel that this design decision improves the interpretability



(a)



(b)

Figure 3.12: Distribution of the overall bot score versus the absolute values of the polarity scores of the content of (a) politicians' Twitter accounts and (b) all other Twitter accounts.

of our model. For some words, such as **"a"** or **"the"**, it definitely does not make sense to assign a political polarity.

Table 3.6: The tweet-level classification performance on the politicians' withheld tweets in our testing set when we assign polarity scores to all tokens versus only assigning polarity scores to hashtags and entities. In each entry, we show the accuracy followed by the F_1 score. We show the best results for each column in bold.

Results (accuracy; F_1)	Polarized **PEM**	Complete **PEM**
Using $\mathbf{z}^{(p)}$ of All Tokens	**0.8369**; **0.8366**	0.8337; **0.8334**
Using $\mathbf{z}^{(p)}$ of Only Entities and Hashtags	0.8339; 0.8337	**0.8338**; 0.8330

As one can see in Table 3.6, assigning political polarities to tokens other than named entities and hashtags does not seem to harm our classification performance. We show it by comparing the tweet-level classification results of our **Complete PEM** model on the withheld testing set of the politicians' tweets (i.e., the same testing set that we used in

Section 3.5.4.1).

3.6 Limitations

We highlight several important limitations of our work. Naturally, our discussion is not exhaustive, and it is also relevant to think about other limitations. The limitations we mention include our data limitations and model limitations.

3.6.1 Incomplete Data

We consider only textual information. Therefore, we overlook images, videos, and other types of information.

3.6.2 Model Limitations

We designed our **PEM** model to infer political polarity scores from entities and hashtags, so it is not helpful for inferring the polarity of tweets that have no entities or hashtags. Additionally, our **PEM** model does not take time stamps into account, so it does not consider the dynamic nature of polarities.

3.6.3 Training-Set Biases and Other Issues

Our design decision of assigning political polarities to items in a training set enables one to automatically assign labels at scale. However, it can be undesirable to make such assignments a priori.

We use the tweets of politicians because their accounts are verified and they have a consistent, unambiguous, and self-identified political affiliation. However, this choice introduces biases and other potential issues. First, the size of our training data set is necessarily limited, and it is easier for models to overfit data when using small data sets than when using large ones. Second, our results may be sensitive to the time window in which we collected tweets. For example, polarization in tweets may be more apparent during elections than at other times. Third, politicians are not necessarily representative of other social-media users. Fourth, we did not train our model to handle bot or cyborg accounts. We used verified Twitter accounts in our training data set, so it presumably does not have any bots or cyborgs. (Our estimation of bot probabilities supports this presumption.) Bot accounts are very common on Twitter [45], so it is necessary to be cautious when applying our model directly to typical Twitter data sets.

The verified Twitter accounts of politicians are very different in nature from the Twitter accounts of other users. We saw ramifications of such differences in our classification results. Using $\text{BERT}_{\text{base}}$ to classify tweets from politicians versus those of other accounts yields an accuracy of 0.7590 and an F_1 score of 0.7595 on the testing set. If we partition the set of non-politician accounts into two groups that each have the tweets of 1,293 accounts (which we assign uniformly at random) and try to classify the group of each tweet, we obtain an accuracy of 0.4600 and an F_1 score of 0.6276.

3.6.4 Quantifying Political Polarity

There are many possible ways to quantify political polarity. We chose to assign labels of “liberal” and “conservative”, but other dichotomies are also relevant. Moreover, we designed our **PEM** model learn a single type of polarity. It cannot simultaneously reveal multiple types of political polarities.

3.6.5 Sarcasm and Irony

In our work, we did not analyze nuanced situations, such as sarcasm and irony, that depend heavily on context. Sarcasm plays an important role in social media [146], and it is worth generalizing our **PEM** model to be able to handle it successfully in the future.

3.7 Conclusions

We studied the problem of inferring political polarities in embeddings of entities and hashtags. To capture political-polarity information without using auxiliary word pairs, we proposed **PEM**, a multi-task learning model that employs an adversarial framework.

Our experiments illustrated the effectiveness of our **PEM** model and the usefulness of the embeddings that one can produce from it. In principle, it is possible to extend our approach to extract any type of polarity of an embedding (while attempting to minimize the effects of polarity on other components). One can also extend our **PEM** model to deploy it with a variety of embedding strategies.

3.8 Ethics Statement

There are several ethical points to consider in our work.

First, one needs to consider our data sets. The data that we used comes from publicly available sources, and our training data comes from the verified accounts of politicians. We do not store any sensitive information (such as real-time locations) from Twitter. It is important to be aware of Twitter's privacy policy (see `https://twitter.com/en/privacy`) when downloading and using data from Twitter.

There are also important ethical considerations when using the results of embeddings like ours. Our **PEM** model yields interesting and occasionally counterintuitive results. One must be cautious when using such results for subsequent tasks (e.g., when drawing conclusions about an individual's political views). Additionally, models inherit biases from training data sets, and they can exacerbate such biases [112].

The conclusions that we obtained from applying our **PEM** model are based on the existing posts of social-media accounts. One must be cautious when subsequently inferring what such accounts may post in the future and especially if one seeks to use any insights from our model to inform behavior, actions, or policy.

Declarations

List of Abbreviations

Table 3.7: List of Abbreviations

Abbreviation	**Corresponding Term**
PEM	Polarity-aware Embedding Multi-task learning
t-SNE	t-distributed stochastic neighbor embedding
BERT	Bidirectional Encoder Representations from Transformers
GloVe	Global Vectors for Word Representation
TIMME	Twitter Ideology-detection via Multi-task Multi-relational Embedding
NCE	noise-contrastive estimation
KDE	kernel density estimation

In Table 3.7, we list the most important abbreviations in our paper.

Ethics Approval and Consent to Participate

Not applicable.

Consent for Publication

Not applicable.

Availability of Data and Materials

Our code, the data sets of the politicians, and the embedding results of our models are available at `https://bitbucket.org/PatriciaXiao/pem/src/master/`.

Competing interests

Not applicable.

Funding

This research was supported by the National Science Foundation (through grants III-1705169, NSF 1937599, NSF 2119643, and 1922952), an Okawa Foundation Grant, Amazon Research Awards, Cisco research grant USA000EP280889, Picsart Gifts, and Snapchat Gifts.

Authors' Contributions

Zhiping Xiao, Pei Zhou, Prof. Mason A. Porter, and Prof. Yizhou Sun conceived and conceptualized the study. Zhiping Xiao, Jeffrey Zhu, Yining Wang, and Wen Hong Lam performed the analysis and wrote the initial draft of the paper. Zhiping Xiao, Prof. Mason A. Porter, and Prof. Yizhou Sun reviewed and extensively edited the manuscript, determined what additional analysis was necessary, and produced the final version of the manuscript.

Acknowledgements

We thank Yupeng Gu and Zhicheng Ren for helpful discussions.

CHAPTER 4

A Social Dynamical System for Twitter Analysis

Public opinions on social phenomena often change with time. Twitter (which has been rebranded as 𝕏), as one of the most popular social-media platforms in the world, offers an opportunity to study opinions without having to collect them through surveys. We propose to model the constantly changing stances of Twitter accounts as the constantly changing status of nodes in a graph with fixed network structure and time-dependent node features.

To model a graph structure efficiently and effectively, many researchers use graph neural networks (GNNs) [72]. In recent years, GNNs have played an increasingly important role in many applications, including social-network analysis [11, 86]. However, most such models are designed for time-independent network data. To model a dynamical social system using a GNN, we need to use a GNN that is appropriate for time-dependent data.

Many challenges exist from the data perspective. There are only a few data sets that are both time-dependent and multimodal. It is already hard to have a data set with both relationship and text information. It is ever rarer to find such data sets that also have temporal information. Simulated social-network data has its own limitations. We can never expect a synthetic social network to reflect all aspects of a real-world social network. Most synthetic social networks reveal only certain aspects of a social network's patterns. Therefore, we will collect real-world data and build a new data set from scratch.

We propose to model changes of opinions in real-world data sets. The difficulty of data collection, the scarcity of observations, the massive amount of noise in data, and the lack of ground-truth labels make it very challenging to prepare the data sets and build a reliable and stable model.

We propose a **L**atent **S**ocial **D**ynamical-**S**ystem framework (**LSDS**) that learns to model the dynamical latent opinions of social-media accounts from the text in their posts. It incorporates continuous time, uses a neural-ODE framework,allows one to consider the irregularly-sampled observations, and follows the training pipeline of a variational-autoencoder (VAE) framework. We create our own dynamic data sets from data that we collected on Twitter. Our **LSDS** model consists of three major components: (1) a temporal encoder, which takes in all observations before the starting time and yields an initial hidden state (i.e., initial latent opinion embedding) of any node at the starting time; (2) an ODE solver, which encodes the dynamic opinion-updating rules by optimizing a GNN, which serves as the ODE function, so that we can use the initial hidden state to predict an account's latent opinion at any future time; and (3) a temporal decoder, where we use the predicted future latent opinions for several downstream tasks.

To the best of our knowledge, we are the first to propose such a dynamic social-network framework that targets at latent opinions of accounts. Our experiments reveal that the **LSDS** framework can help reveal the opinion changes of nodes in social networks.

4.1 Introduction

Over the past few decades, there has been rapid growth in internet usage. People from across the world now share their opinions online and interact with each other. One can obtain behavioral data of people from their posts on social-media platforms, such as $\mathbb{X}$ (which used to be known as Twitter). This process of collecting opinions, although it comes with other types of data bias [46], is much more convenient than traditional methods such as surveys and interviews.

To capture opinion dynamics in a social-media network, we build a machine-learning model to predict future observations from past observations by learning the updating rules of the accounts' latent opinions. As we show in the schematic illustration in Figure 4.1, accounts' opinions are influenced by other accounts that interact with them. For example,

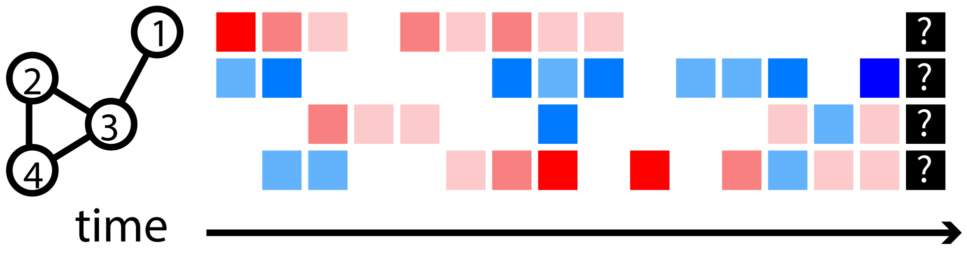


Figure 4.1: Illustration of changes in node opinions as they interact with each other in a network. The colors indicate observed opinions, such as on a liberal–conservative spectrum. The shade of a color indicates the extremeness of an opinion. Therefore, the darkest blue and darkest red indicate the most extreme opinions. The white color signifies that we do not observe the opinion of a node at the associated time.

in the depicted scenario, after hearing some red-sided opinions from its neighbor node 3, node 4 changes from a blue opinion to a red opinion. However, node 3 agrees partly with both the blue side and the red side, so later node 3 and node 2 both broadcast some blue-sided opinions, turning node 4 from extreme red to a more moderate red opinion. If our framework is capable of understanding the opinion-updating rules that caused these changes, it should also be good at forecasting future opinions.

To design our framework, we investigated related works and found inspirations from different perspectives. It is very common to model a graph-structured data set using a GNN [62, 77, 78, 160]. However, dynamic GNNs have received much less attention than time-independent GNNs [138]. Recently, some computer scientists have verified that, in carefully designed experiments, neural networks are capable of learning the update rules in the traditional opinion models [111], such as the DeGroot [2] and Friedkin–Johnsen (FJ) [47] models. Many models of opinion dynamics have been studied by social scientists, physicists, and others over many decades [17, 18, 35, 92, 108, 109, 130]. Although recent explorations have established a connection between neural-network models and the traditional opinion models, there remain many important problems to pursue in using neural networks to study opinion dynamics. First, most existing works focus on either simulated data or labor-intensive manual labeling, and it is thus unclear how successful they can be in real-world scenarios at scale. The existing models assume that opinions are explicitly known at each observation, whereas we hold a different point of view. From our earlier exploration [158], we conclude that the text content in accounts' posts often does does accurately reflect their opinions (see Chapter 2). Therefore, we propose to model their opinions as latent representations to be learned from observations. Second, many existing opinion models rely on discrete time or observations that are sampled at constant time intervals. We seek to formulate a model that can predict node opinions at any future time t, where t is any real number (i.e., time is continuous) within a reasonable horizon. Third, there has not yet been a framework where traditional rule-based opinion-update rules and learnable neural-network-based update rules can be used alternatively and compared to each other. We propose to build a framework that is able to handle real-world data observations, make predictions on a continuous timeline, and is flexible enough to encompass different opinion-model variants.

Social-media networks provide multimodal data, such as text in accounts' posts and interaction behaviors between accounts. Our direct observations include the observation of interactions of the accounts and the observation of accounts' features (such as text). Edges that reflect follower–followee relationships remain consistent because following relationships do not change often within the time range (i.e., approximately a year) of our data. We treat the temporal relationships (specifically, retweet, mention, and like) between nodes as temporal edges. They provide ground-truth signals for our downstream tasks.

We design a **L**atent **S**ocial **D**ynamical-**S**ystem (**LSDS**) framework. Inspired by LATENT GRAPH ODE (LG-ODE) [67] and NEURAL RELATIONAL INFERENCE (NRI) [77], our social dynamical system uses a NEURAL ODE framework [25], with an ODE function defined using GNN function and the whole framework is trained end-to-end (i.e., from raw input to final output and do not need any intermediate steps) with a VARIATIONAL AUTOENCODER (VAE) pipeline [165]. To project the text features into a continuous vector space, we use a pretrained SENTENCE-BERT model [125] to generate tweet-level text embeddings from raw text content.

Our experimental results (see Section 4.5) demonstrate that our framework is useful for modeling the changes of accounts' opinions in a social network. Our ablation studies suggest that, for our tasks and within our model variants, currently the best encoder is a temporal GNN model and that the best ODE function is an NRI model, as their performance are the best on most of our downstream tasks. Of the few decoder tasks that we have tried (text-embedding reconstruction, node-polarity prediction, and interaction prediction), interaction prediction is the most challenging one. Nonetheless, one can readily propose an alternative encoder component, ODE function component, or other decoder for a new downstream task. These new components can be added to our framework. From this perspective, our framework has great extensibility.

Our main contributions are as follows:

1. To the best of our knowledge, we are the first to propose a neural-ODE dynamical-systems framework that is capable of handling multimodal real-world data sets.

2. We collected and processed temporal multimodal data sets from $\mathbb{X}$ (i.e., Twitter). We expect that our data set will also help researchers with future studies.

3. Our experimental results suggest that our **LSDS** model is able to capture the update rules of a social dynamical system. Additionally, by comparing different variants of opinion models in case studies and ablation studies, we demonstrate that each component of **LSDS** is important.

4. Our **LSDS** framework is compatible with a variety of encoder and decoder architectures and a large variety of selection of ODE functions. It thus has great potential to be adjusted for use in many other tasks.

5. In addition, we propose an automated data-preprocessing method, which does not need labor-intensive manual-labeling.

4.2 Related Work

In this section, we discuss our related works from four perspectives. They are, graph neural networks, neural ordinary differential equations, models of opinion dynamics, and variational encoders.

4.2.1 Graph Neural Networks

Graph neural networks (GNNs) are neural-network models that are designed specifically for graphical data that consists of nodes and the edges between them [72]. GNNs are able to take advantage of neural networks and solve graph-based problems. They have succeeded in many fields and have thus attracted much attention [1, 123, 157, 170, 174].

Based on model architecture and the graph-based tasks that they are targeting, GNNs are roughly classified into the following categories: recurrent-based GNNs, convolution-based GNNs, spatiotemporal GNNs, graph autoencoders (GAEs), graph adversarial networks, and graph reinforcement-learning models [123]. These categories overlap with each other, so one type of GNN can belong to multiple classes. In this chapter, we focus exclusively on convolution-based GNNs.

Convolutions on graph data [6] can be either in the spectral domain [65] or in the spatial domain [175]. Spectral convolutional GNNs use ideas from spectral graph theory. One of the key differences between spectral convolutional GNNs and spatial convolutional GNNs is that the latter consider only the neighborhood of a target node, whereas the former require access to all nodes. For instance, GCN [78] is a spectral model and normalizes the adjacency matrix using the symmetrically normalized Laplacian of an entire graph, whereas NRI [77], GIN [160], GraphSAGE [62], and many other convolutional GNNs are spatial models [5, 168]. Spectral convolutional GNN models typically possess some useful properties that can be proved theoretically [48, 128, 176]. However, spatial convolutional GNNs tend to be more efficient and thus less costly to train. Some spatiotemporal GNNs combine GNN components and recurrent-neural-network (RNN) components for sequence prediction.

These models have performed well in applications such as traffic forecasting [60, 173].

Despite the popularity and the demonstrated effectiveness of GNNs, they have not been used very much to analyze dynamic data on graphs [138]. Exploring how GNNs can be used to help analyze dynamical real-world graph data, such as opinions in real-world social networks, remains an open and challenging direction.

4.2.2 Neural Ordinary Differential Equations

Solving ordinary differential equations (ODE) is a traditional field that researchers have been exploring for hundreds of years [63, 68, 102]. In mathematics, the term "ordinary" is used in contrast with "partial" to emphasize that only one independent variable is involved. For example, with the dependent variable y and the independent variable x, the only possible derivative term in an ODE is $\frac{dy}{dx}$. In a partial differential equation (PDE), there are more than one independent variables, say x and w, so a PDE can include both $\frac{\partial y}{\partial x}$ and $\frac{\partial y}{\partial w}$. An ODE takes the form

$$\frac{dy}{dx} = f(x, y)\,, \tag{4.1}$$

where x is the independent variable and y is the dependent variable. We can also denote the y value with a given x_i value by $y(x_i)$. The function $f(x, y)$ determines how y changes instantaneously with respect to x. Additionally, The ODE (4.1) is subject to an initial condition $y(x_0) = y_0$ or a boundary condition $y(x_a) = y_a$, where y_0 or y_a are known values. Initial and boundary conditions provide essential information that specifies the particular solution(s) of the ODE among all possible solutions. In an ODE, both time and space are continuous. There are numerous numerical approaches to solve ODEs [104].

The Neural-ODE [25] model combines a neural-network architecture with an ODE solver by using a neural network to implement the ODE function $f(x, y)$. This design is helpful to bring neural networks from discrete space to continuous space. They allow one to approximate derivatives in a continuous manner. The contributions of the Neural-ODE model are twofold:

1. The Neural-ODE model re-implemented traditional ODE solvers (e.g., a Runge–Kutta solver) so that they can work in deep-learning settings.

2. By using neural networks instead of algebraic formulas as the ODE functions, Neural-ODE is able to enhance the expressiveness of a model. It thereby improves its flexibility

For example, ODE solvers help in the construction of a continuous version of a neural-network layer-wise gradient update. A neural ODE can also be used to model changes in nodes' representations in continuous time. One example is the Latent Graph ODE (LG-ODE) model [67], where time-dependent physical system graphs (e.g., predicting the movement of bouncing balls) were successfully analyzed. Our work follows this line of research. In our case, however, we are interested in opinion dynamics on social networks.

4.2.3 Models of Opinion Dynamics

Social scientists have long been very interested in modeling how people's opinions change as they interact with each other [17, 35, 92, 108]. Models of opinion dynamics include the DeGroot model [2], the Friedkin–Johnsen (FJ) model [47], the Hegselmann–Krause model (HK, a type of bounded-confidence model) [124], and many others. Each of these models relies on various underlying assumptions.

Recently, the method Sociologically-Informed Neural Network (SINN) [111] used neural-network architectures to learn the derivatives of opinion variables at each time step of a discrete-time dynamical system. They used an opinion model to give ground-truth update rules of the opinion value, and they then trained a neural network to learn the opinion-update rules automatically. Experiments in [111] demonstrated that neural networks are capable of learning to act as a traditional opinion model. Their work demonstrates the power of neural-network models and illustrates their potential to help in studies of opinion dynamics.

4.2.4 Variational Autoencoder

A variational autoencoder (VAE) [76] is a framework where variational inference is applied for training. It is set up as an autoencoder, which makes it differentiable for backpropagation. It includes both an encoder and a decoder. The encoder takes the inputs and learns a corresponding latent-space variable distribution, and the decoder draws samples from the latent space as the hidden representation and reconstructs the input [52].

In many existing works (such as NRI [77] and LG-ODE [67]) that seek to capture the changes of the states of the nodes in a network, VAE is used as the training framework. We adopt a similar training framework as a standard, unsupervised VAE. However, because our data set is partially labeled and thus allows a semisupervised approach, our training pipeline is no longer precisely a VAE framework. We keep the encoder and latent space, and the decoder is able to learn many different objectives, including but not limited to the input-reconstruction task.

4.3 Problem Definition

We learn to model how the opinions of social-network accounts affect each other and change with time. In this section, we discuss how we define a dynamical social graph, and from which signals we can estimate the accounts' opinions.

4.3.1 Dynamic Social Graph

There are many different types of dynamic networks [138]. Some have time-independent sets of nodes, whereas others have time-dependent sets of nodes (i.e., nodes can appear and disappear). Some networks with time-dependent edges allow edges to appear and disappear, and others only allow changes in edge weights.

Social networks are commonly modeled as graphs. When we model a social network, each account is a node, and we model each follower–followee relationship as an edge.

In our scenario, due to the nature of a social network, the follower–followee relationships change at a relatively low frequency. Opinions change much faster, so we regard follower–followee relationships as time-independent. Accordingly, we study dynamical systems on a time-independent network [120].

We focus on the changing opinions of each account in a social dynamical system. We model our social dynamical system as a multi-agent dynamical system. We describe a social network as a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{v_1, v_2, \ldots, v_N\}$ is the set of N nodes that each represents an account, which interact with the other accounts. The set $\mathcal{E}$ encodes the accounts' follower–followee relations, where $\langle v_i, v_j \rangle \in \mathcal{E}$ indicates that account v_i follows account v_j. The graph $\mathcal{G}$ has $M = |\mathcal{E}|$ edges.

We observe the accounts' opinions through the text content from their posts. For each account v_i, we have a series of observations $o_i = \{\mathbf{o}_i^t\}$. For any account v_i and any time t, the observation $\mathbf{o}_i^t \in \mathbb{R}^D$ is the vector-space representation of the corresponding text content at time t. In practice, we generate $\mathbf{o}_i^t$ from the account's raw tweet text using the pretrained SENTENCE-BERT model [125]. If account v_i posted multiple tweets at time t, we use the mean value of all tweets' embeddings to represent the account v_i's corresponding observation $\mathbf{o}_i^t$ of account v_i.

As one can expect, in our data sets, each account posts content at their unique frequency, so that each account v_i has a unique observation time sequence $\{t_i^j\}_{j=0}^{T_i}$. Let $\mathbf{z}_i^t$ denote the hidden representation of account v_i at time t; the quantity $\mathbf{z}_i^t$ is account v_i's latent opinion. Our goal is to model how $\mathbf{z}_i^t$ changes according to time t.

4.3.2 Opinion Evaluation

We use two different ways to evaluate our model's performance of estimating future opinions. Each of these ways corresponds to a different decoder task.

4.3.2.1 Interaction Prediction

We can evaluate the quality of the hidden representation $\mathbf{z}_i^t$ and $\mathbf{z}_j^t$ of accounts v_i and v_j by using these representations to infer the interactions between them at time t.

Using our underlying time-independent graph architecture that is defined by "follow" relationships, we treat all other types of interactions (specifically, mention, retweet, reply, and like) as time-dependent edges to be inferred. That is, we have multiple types of time-dependent edges: $\{\mathcal{E}_{\text{mention}}, \mathcal{E}_{\text{retweet}}, \mathcal{E}_{\text{reply}}, \mathcal{E}_{\text{like}}\}$. The time-dependent edge $\langle i, j, t\rangle \in \mathcal{E}_{\text{mention}}$ signifies that account v_i mentioned account v_j in a their post at time t.

To evaluate how well our model captures the update rules of opinions, we can calculate an edge-prediction score. For example, for $\langle i, j, t\rangle \in \mathcal{E}_{\text{R}}$ with the relation $\text{R} \in \{$mention, retweet, reply, like$\}$, we measure

$$\text{Score}(i, j, t, \text{R}) = f_{\text{R}}(\mathbf{z}_i^t, \mathbf{z}_j^t)\,,$$

where f_{R} (with $\text{R} \in \{\text{mention}, \text{retweet}, \text{reply}, \text{like}\}$) is a function to compute a score that reflects how likely there is an edge between the given nodes according to $\mathbf{z}_i^t$ and $\mathbf{z}_j^t$ (i.e., their latent opinions at time t).

4.3.2.2 Polarity Inference

One of the key challenges of real-world social-network data sets is the lack of ground-truth labels. In our case, the data set is political in nature, so it is reasonable to assign a political-polarity label to each tweet.

Once we have political-polarity labels for observed tweet, we can learn to recover the polarity score p_i^t from the corresponding hidden representation $\mathbf{z}_i^t$. The more precise our estimation, the more powerful our model is.

Using the pretrained text-embedding model from our PEM model [159] (see Chapter 3), where the last dimension of the token embedding is a political-polarity score, we are able to assign a polarity score $p_i^t \in \mathbb{R}$ to each tweet.

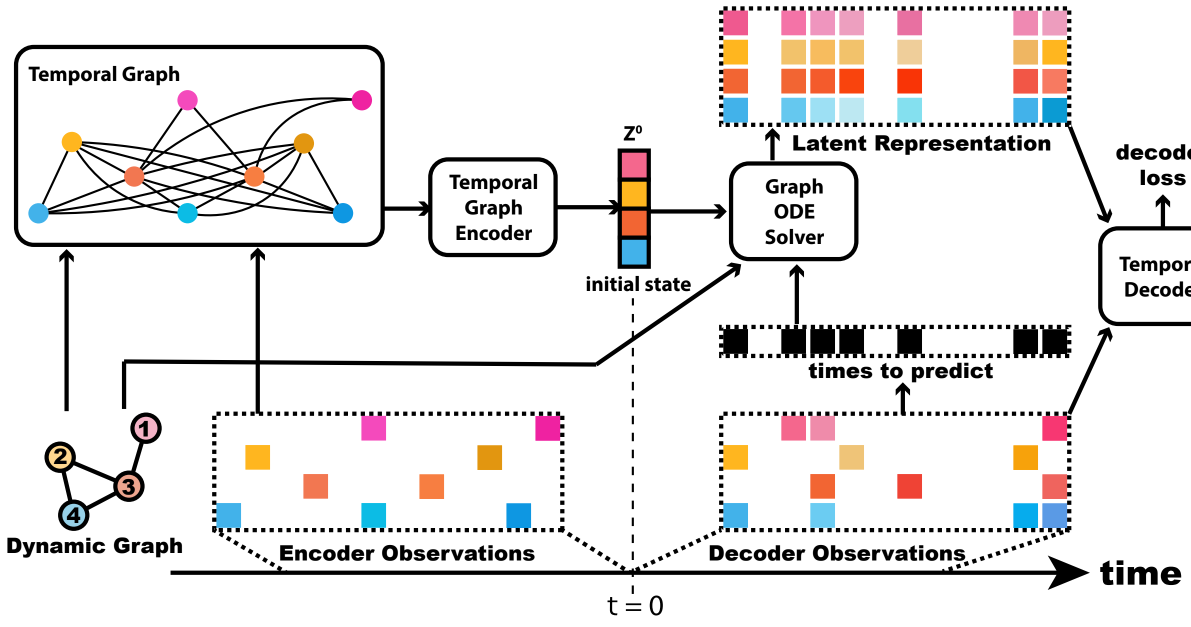


Figure 4.2: A schematic overview of our model's architecture. We implement the graph ODE function g_i as a GNN function, where all edges in the network depicts the follower–followee relationships in the original graph (i.e., the graph on the bottom-left corner of this Figure).

4.4 Methodology

Our **LSDS** model consists of three main components: a time-dependent encoder, an ODE solver that handles a GNN ODE function, and a time-dependent decoder. In Figure 4.2, we give a schematic overview of how these components work together in our model.

The encoder transforms the observations of accounts into the initial state of their latent opinion representations. It is time-dependent because it considers historical observations before the starting time $t = 0$ by using both observed content and the time stamps. It aims to encode the hidden space of the initial state $\mathbf{z}_i^0$ from all observations $\mathbf{o}_i^{t'}$ before the starting time (i.e., for all observations with $t' < 0$). To achieve this goal, we first construct a temporal graph $\mathcal{G}_{\text{temporal}}$ from these observations. In this temporal graph, each node represents an observation. For example, in Figure 4.2, we show how our framework works with an example of 4 accounts (social network is shown on the bottom left corner, and we denote the accounts' nodes as node 1, 2, 3 and 4). There are 2 observations on node 1 before

$t = 0$ and 2 observations on node 3. Therefore, node 1 in the social network results in 2 nodes $\{o_{1a}, o_{1b}\}$ in the temporal graph, and similarly node 3 corresponds to 2 other temporal-graph nodes $\{o_{3a}, o_{3b}\}$. An edge in a temporal graph represents a follower–followee relationship between the corresponding accounts. For example, in Figure 4.2, node 1 is adjacent to node 3. The temporal graph then has the edges: $\{(o_{1a}, o_{3a}), (o_{1a}, o_{3b}), (o_{1b}, o_{3a}), (o_{1b}, o_{3b})\}$. After constructing a temporal graph, we encode it with a temporal-graph encoder, whose output will allow us to generate the initial latent states $\mathbf{z}_i^0$ of all accounts $(i = 1, 2, \ldots, N)$. Ideally, the latent representation $\mathbf{z}_i^t$ captures the latent opinions of account v_i at time t. This representation is related to the observation o_i but is not necessarily identical to the direct observation.

An ODE solver allows us to numerically integrate an ODE with a GNN ODE functon on the right-hand side. The GNN is designed as a message-passing framework, so traditional opinion models such as DeGroot can also fit into this framework (see Section 4.4.2). It aims to provide $\mathbf{z}_i^t$ from the initial state $\mathbf{z}_i^0$, the target time t, and the graph-ODE function.

The decoder uses a hidden representation such as $\mathbf{z}_i^t$ to do downstream tasks. We can evaluate our model's effectiveness from its performance on the downstream tasks. The decoder is also time-dependent because the downstream tasks use time-dependent hidden representations as their inputs.

4.4.1 The Time-Dependent Encoder

In this subsection, we introduce the architecture of our temporal-graph encoder, and another variant of time-dependent encoder that also works in our framework, serving as a baseline encoder (see Section 4.4.1.2).

4.4.1.1 The Temporal Graph Encoder

We use a temporal-graph encoder to encode the observation sequences $o_i = \{\mathbf{o}_i^{t'} | t' < 0\}$ before the starting time $t = 0$ for each account v_i (with $i \in \{1, 2, \ldots, N\}$) into a factorized distribution of initial hidden states:

$$q_\phi(\mathbf{Z}^0) = \prod_{i=1}^{N} q_\phi(\mathbf{z}_i^0 | o_1, o_2, \ldots, o_N)\,. \tag{4.2}$$

Because this multi-agent dynamical system consists of many coupled components, similar to the design of LG-ODE [67], we choose to use a model that captures interactions between the nodes, instead of simply modeling each node's observation sequence separately using RNNs [126].

We use a similar encoder architecture as in the LG-ODE model. Our encoder has two phases: (1) a dynamic-node-representation learning (DNRL) phase, in which we learn a function f_{embed} that learns a structural contextualized embedding $\mathbf{h}_i^t$ for each observation $\mathbf{o}_i^t$ of node i at time t; (2) a temporal self-attention (TSA) phase, in which we learn a function f_{readout} to represent the sequence $\{\mathbf{h}_i^t | t < 0\}$ of each account v_i as a fixed-length vector $\mathbf{u}_i$, which we then use to determine the posterior distribution of the initial state $\mathbf{Z}^0$ (i.e., the matrix who stacks N vectors $\{\mathbf{z}_i^0 | i = 1, 2, \ldots, N\}$ together).

In the DNRL phase, we learn the function f_{embed}, which is a spatiotemporal GNN model. In this model, we first construct a temporal graph $\mathcal{G}_{\text{temporal}} = (O, E)$, where each node in O represents an observation $\mathbf{o}_i^t$. An edge exists between the nodes representing $\mathbf{o}_i^{t_1}$ and $\mathbf{o}_j^{t_2}$ if and only if node v_i and v_j are adjacent in the input graph $\mathcal{G}$. We also consider the impact of previous observations of the same account, so nodes with observations $\mathbf{o}_i^{t_1}$ and $\mathbf{o}_j^{t_2}$ are adjacent when $i = j$.

For each layer l, we represent the hidden representation of $o_{\text{src}} \in O$ as $\mathbf{h}_{\text{src}}^{(l)}$ and $o_{\text{tgt}} \in O$ as $\mathbf{h}_{\text{tgt}}^{(l)}$, respectively. The nodes learn from their neighborhood via a standard message-passing and aggregation paradigm [64], where we use attended message-passing and use a summation operation as its aggregation function. That is,

$$\mathbf{h}_{\text{tgt}}^{(l)} = \mathbf{h}_{\text{tgt}}^{(l-1)} + \sigma\Big(\sum_{\text{src} \in \mathcal{N}_{\text{tgt}}} \Big(\text{Attention}(\mathbf{h}_{\text{src}}^{(l-1)}, \mathbf{h}_{\text{tgt}}^{(l-1)}) \cdot \text{Message}(\mathbf{h}_{\text{src}}^{(l-1)}, \Delta t(\text{src}, \text{tgt})) \Big) \Big)\,, \tag{4.3}$$

where $\mathcal{N}_{\text{tgt}}$ is the neighborhood of node o_{tgt} (i.e., all nodes that are adjacent to o_{tgt}). We assume that any source node o_{src} is adjacent to the target node o_{tgt}, denoted as $\text{src} \in \mathcal{N}_{\text{tgt}}$.

The quantity $\Delta t(\mathrm{src}, \mathrm{tgt})$ is the temporal gap between the two observations src and tgt. The time-dependent message function Message$(\cdot)$ is

$$\begin{aligned} &\mathrm{Message}(\mathbf{h}_{\mathrm{src}}^{(l-1)}, \Delta t(\mathrm{src}, \mathrm{tgt})) = \mathbf{W}_{\mathrm{val}} \hat{\mathbf{h}}_{\mathrm{src}}^{(l-1)}\,, \\ &\text{where } \hat{\mathbf{h}}_{\mathrm{src}}^{(l-1)} = \sigma(\mathbf{W}_{\mathrm{temp}}[\mathbf{h}_{\mathrm{src}}^{(l-1)} \parallel \Delta t(\mathrm{src}, \mathrm{tgt})]) + \mathrm{TE}(\Delta t(\mathrm{src}, \mathrm{tgt}))\,, \end{aligned} \tag{4.4}$$

the linear transformations $\mathbf{W}_{\mathrm{temp}}$ and $\mathbf{W}_{\mathrm{val}}$ ensure that the output dimension is the same as the input dimension, TE is a projection function that projects the temporal observation gap $\Delta t(\mathrm{src}, \mathrm{tgt})$ onto a vector with the same dimensionality as the hidden space (i.e., the same dimensionality as $\mathbf{h}_{\mathrm{src}}^{(l-1)}$). We separately project its odd and even dimensions:

$$\mathrm{TE}(\Delta t)_{2i+1} = \cos(\Delta t / 10000^{2i/d})\,, \quad \mathrm{TE}(\Delta t)_{2i} = \sin(\Delta t / 10000^{2i/d})\,.$$

The attention function Attention$(\cdot)$ is based on a self-attention mechanism [148]. We introduce the linear transformations $\mathbf{W}_{\mathrm{key}}$ and $\mathbf{W}_{\mathrm{que}}$, which have the same dimensionality as $\mathbf{W}_{\mathrm{val}}$. The transformations $\mathbf{W}_{\mathrm{key}}$, $\mathbf{W}_{\mathrm{que}}$, and $\mathbf{W}_{\mathrm{val}}$ together project the input node representations $\hat{\mathbf{h}}_{\mathrm{src}}^{(l-1)}$ into values $\mathbf{W}_{\mathrm{val}} \hat{\mathbf{h}}_{\mathrm{src}}^{(l-1)}$, keys $\mathbf{W}_{\mathrm{key}} \hat{\mathbf{h}}_{\mathrm{src}}^{(l-1)}$, and queries $\mathbf{W}_{\mathrm{que}} \mathbf{h}_{\mathrm{tgt}}^{(l-1)}$. We then define the attention function Attention$(\cdot)$ as

$$\mathrm{Attention}(\mathbf{h}_{\mathrm{src}}^{(l-1)}, \mathbf{h}_{\mathrm{tgt}}^{(l-1)}) = (\mathbf{W}_{\mathrm{key}} \hat{\mathbf{h}}_{\mathrm{src}}^{(l-1)})^T (\mathbf{W}_{\mathrm{que}} \mathbf{h}_{\mathrm{tgt}}^{(l-1)}) \cdot \frac{1}{\sqrt{d}}\,, \tag{4.5}$$

where d is the hidden dimensionality. Finally, we stack the $L = 2$ layers of the spatiotemporal convolutional layers to obtain the representation $\mathbf{h}_i^t = \mathbf{h}_{\mathrm{tgt}}^{(2)}$, where the corresponding node with observation $\mathbf{o}_i^t$ in $\mathcal{G}_{\mathrm{temporal}} = (O, E)$ is o_{tgt}.

We have a spatiotemporal embedding $\mathbf{h}_i^t$ of each $\mathbf{o}_i^t$. However, each individual account v_i can have an observation sequence o_i of variable length. To encode the sequence as a fixed-dimensional vector $\mathbf{u}_i$ and then derive the hidden-representation distribution's mean and standard deviation $\boldsymbol{\mu}_i, \boldsymbol{\sigma}_i \in \mathbb{R}^d$ from it, we use the TSA phase. In the TSA phase, we learn f_{readout}. That is,

$$[\boldsymbol{\mu}_i \parallel \boldsymbol{\sigma}_i] = f_{\mathrm{readout}}(\{\mathbf{h}_i^t | t \in \{t_i^j\}_{j=0}^{T_i}, t < 0\})\,,$$

where f_{readout} outputs a matrix $[\boldsymbol{\mu}_i \parallel \boldsymbol{\sigma}_i]$, which denotes the concatenation of $\boldsymbol{\mu}_i$ and $\boldsymbol{\sigma}_i$. We use $[\cdot \parallel \cdot]$ to represent the concatenation operation.

We incorporate temporal information into the embedding $\mathbf{h}_i^t$ and write

$$\hat{\mathbf{h}}_i^t = \sigma(\mathbf{W}_{\text{temp}}[\mathbf{h}_i^t \parallel \Delta t]) + \text{TE}(\Delta t)\,,$$

where $\Delta t = t - t_{\text{start}}$ is the difference between the current observation and the earliest observation. This step is very similar to how we compute Message$(\cdot)$.

We compute a global sequence vector $\mathbf{a}_i \in \mathbb{R}^d$ and a linear transformation $\mathbf{W}_a \in \mathbb{R}^{d\times d}$ by calculating

$$\mathbf{a}_i = \tanh\left(\left(\frac{1}{|\{o_i|t<0\}|}\sum_t \hat{\mathbf{h}}_i^t\right)\mathbf{W}_a\right)\,, \qquad \mathbf{u}_i = \frac{1}{|\{o_i|t<0\}|}\sum_t \sigma(\mathbf{a}_i^T\hat{\mathbf{h}}_i^t)\hat{\mathbf{h}}_i^t\,,$$

where $|\{o_i|t<0\}|$ is the observed sequence length that is accessible by the encoder. We then have another transformation layer:

$$[\boldsymbol{\mu}_i \parallel \boldsymbol{\sigma}_i] = f_{\text{posterior}}(\mathbf{u}_i)\,. \tag{4.6}$$

In practice, we implement $f_{\text{posterior}}$ as a linear-transformation layer with input dimension d and output dimension $2d$.

With our approximate posterior distribution and our hidden-space distribution parameters $\boldsymbol{\mu}_i$ and $\boldsymbol{\sigma}_i$, we can sample the initial state $\mathbf{z}_i^0$ from the normal distribution with mean $\boldsymbol{\mu}_i$ and standard deviation $\boldsymbol{\sigma}_i$. That is,

$$\mathbf{z}_i^0 \sim q_\phi(\mathbf{z}_i^0|o_1, o_2, \ldots, o_N) = \mathcal{N}(\boldsymbol{\mu}_i, \boldsymbol{\sigma}_i)\,. \tag{4.7}$$

4.4.1.2 GCN Encoder with Time-Dependent Input Features

Another option of our framework's time-dependent encoder component is a standard Graph Convolutional Network (GCN) [78] encoder. This encoder has the identical GNN model architecture as in the original GCN paper and code.[1] This version of the encoder contains a 2-layer graph-convolutional architecture, with the layer-wise update rule

$$\mathbf{h}_i^{(l)} = \sigma\Big(\sum_{v_j\in\mathcal{N}_i}\frac{1}{c_{ij}}\mathbf{h}_j^{(l-1)}\mathbf{W}^{(l-1)}\Big)\,, \tag{4.8}$$

[1]See https://github.com/tkipf/pygcn for GCN code.

where $\mathbf{h}_i^{(l)}$ is the hidden representation of node v_i at layer l. Parameters in the weight matrix $\mathbf{W}^{(l-1)}$ are learnable parameters. The quantity c_{ij} is a normalization factor for the edge between v_i and v_j. We compute c_{ij} during the data-preprocessing step after we add a self-edge to each node. It is given by

$$c_{ij} = \sqrt{|\mathcal{N}_i| \cdot |\mathcal{N}_j|}\,,$$

where $\mathcal{N}_i$ is the neighborhood of node v_i's. The quantity $|\mathcal{N}_i|$ is the size of node v_i's neighborhood.

In the GCN encoder, time information is embedded into the node features. Suppose that we have an observed sequence $\{o_i|t < 0\}$ of node v_i with sequence length SeqEnc_i, we compute a weighted sum of all the observations in the sequence. The weight of the earliest observation is $\frac{1}{2^{\text{SeqEnc}_i}}$, the next observation has weight $\frac{1}{2^{\text{SeqEnc}_i - 1}}$, and so on. The latest observation right before the starting time $t = 0$ has weight $1/2$. The weighted sum of all $\{\mathbf{o}_i^t|t < 0\}$ is the input $\mathbf{h}_i^{(0)}$ to the GCN model.

The last layer of the GCN is followed by a linear-transformation layer with the same architecture as $f_{\text{posterior}}$ (see Equation (4.6)). We then compute the mean $\boldsymbol{\mu}_i$ and the standard deviation $\boldsymbol{\sigma}_i$ of the GCN encoder the same way as in Equation (4.6).

4.4.2 The Graph ODE

A social-network data set with N accounts, together with update rules of their opinions and the initial states of their opinions, can be viewed as a continuous multi-agent dynamical system.The opinion evolution of $\mathbf{z}_i^t$ is governed by a set of coupled first-order ODEs with vector fields g_i that update their values in infinitesimal time steps.

We model our ODE function in a straightforward way using a GNN framework g_i by writing

$$\frac{\mathrm{d}\mathbf{z}_i^t}{\mathrm{d}t} = g_i(\mathbf{z}_1^t, \mathbf{z}_2^t, \ldots, \mathbf{z}_N^t)\,, \tag{4.9}$$

where g_i consists of L_{ODE} layers of message-passing convolutional layers:

$$\mathbf{v}_j^{(l+1)} = f_{\text{aggregate}}^{(l)}\Big(\{f_{\text{message}}^{(l)}(\mathbf{v}_i^{(l)}, \mathbf{v}_j^{(l)})|v_i \in \mathcal{N}_j\}\Big)\,. \tag{4.10}$$

The input is the nodes' representations $\mathbf{v}_i^{(0)} = \mathbf{z}_i^t$ and the output is the derivative $\frac{\mathrm{d}\mathbf{z}_i^t}{\mathrm{d}t} = \mathbf{v}_i^{(L_{\text{ODE}})}$.

With the initial state $\mathbf{z}_i^0 \in \mathbb{R}^d$ for each account v_i, the solution $\mathbf{z}_i^t$ satisfies an initial-value problem (IVP), which we express in the integral form

$$\mathbf{z}_i^t = \mathbf{z}_i^0 + \int_{t'=0}^{t} g_i(\mathbf{z}_1^{t'}, \mathbf{z}_2^{t'}, \dots, \mathbf{z}_N^{t'})\mathrm{d}t' \,. \tag{4.11}$$

There are many numerical ODE solvers to obtain $\mathbf{z}_i^t$. A prominent example is RUNGE–KUTTA methods [134]. The Euler's method we refer to in our code is a particular type of Runge–Kutta methods [56].

The vector field g_i defines the dynamics of the latent state of the ith agent (i.e., account v_i). The coupled dynamics of the agents constitute a multi-agent dynamical system.

We have a GNN-based ODE function in our **LSDS** framework by default (see Section 4.4.2.1), and we also adapt some ODE functions from opinion models to produce other variants (see Section 4.4.2.2).

4.4.2.1 Graph Neural-ODE Function

Our model's ODE function uses a similar updating algorithm as in the NRI model, a type of GNN that has been used successfully to model discrete-time multi-agent dynamical systems [77]. In each layer of an ordinary NRI model, information is passed from nodes to edges and then from edges to nodes:

$$\mathbf{e}_{(i,j)}^{(l)} = f_e^{(l)}([\mathbf{v}_i^{(l)} \parallel \mathbf{v}_j^{(l)}]) \,, \quad \mathbf{v}_j^{(l+1)} = f_v^{(l)}(\sum_{i \neq j} \mathbf{e}_{(i,j)}^{(l)}) \,,$$

where $\mathbf{e}_{(i,j)}^{(l)}$ is the edge embedding between node pair (v_i, v_j) in layer l and the vector $\mathbf{v}_j^{(l)}$ is the node embedding of node v_j at layer l. The functions $f_e^{(l)}$ and $f_v^{(l)}$ are multi-layer perceptrons (MLPs). In NRI [77], Kpif et al. considered a fully connected graph and discrete time steps. We are interested in dynamics on social networks, so we need to consider real-world social-network structure. Therefore, we adapt the standard NRI node-update rules. For each layer,

$$\begin{aligned}\mathbf{e}_{(i,j)}^{(l)} &= f_{\text{message}}^{(l)}(\mathbf{v}_i^{(l)}, \mathbf{v}_j^{(l)}) = f_e^{(l)}([\mathbf{v}_i^{(l)} \parallel \mathbf{v}_j^{(l)}])\,, \\ \mathbf{v}_j^{(l+1)} &= f_{\text{aggregate}}^{(l)}(\{\mathbf{e}_{(i,j)}^{(l)} | v_i \in \mathcal{N}_j\}) = f_v^{(l)}(\sum_{v_i \in \mathcal{N}_j} [\mathbf{v}_i^{(l)} \parallel \mathbf{e}_{(i,j)}^{(l)}]) + \mathbf{v}_j^{(l)}\,.\end{aligned} \tag{4.12}$$

Following settings that have worked well in similar models [67, 77], we set the number of layers in our ODE function L_{ODE} to 1 as a default. When we change it to other values, we find the model performance degrades.

4.4.2.2 Baseline Opinion Models

In some other variants of **LSDS**, we adapt the right-hand sides of our ODE functions from opinion models to GNN models. In our discussion, we use the SINN [111] baseline models because they are already neural-network models and worked well on some simulated data sets. These baselines include the DeGroot model [2], the Friedkin–Johnsen (FJ) model [47], and a Hegselmann–Krause (HK) bounded-confidence model [124].

- Among all opinion models we have translated into GNN model, the DeGroot model is the simplest one with the fewest constraints and requires the least modification from SINN model to GNN model. Denote latent opinion of an account v_i at time t by $\mathbf{z}_i^t \in \mathbb{R}^d$. Its discrete-time update rule on a network is

  $$\mathbf{z}_j^{t+1} = \mathbf{z}_j^t + \sum_{v_i \in \mathcal{N}_j} a_{ij} \mathbf{z}_i^t\,,$$

  where $\mathcal{N}_j$ is the neighborhood of node v_j and $a_{ij} > 0$ is the strength of the edge (i.e., the follower–followee relationship) between two adjacent accounts v_i and v_j. The SINN model uses an ODE as a continuous-time analogue of the DeGroot model in the form

  $$\frac{\mathrm{d}\mathbf{z}_i^t}{\mathrm{d}t} = \sum_{v_i \in \mathcal{N}_j} a_{ij} \mathbf{z}_i^t = \sum_{v_i \in \mathcal{N}_j} \mathbf{m}_i^T \mathbf{q}_j \mathbf{z}_i^t\,,$$

  where $\mathbf{m}_i, \mathbf{q}_j \in \mathbb{R}^K$ are the ith and jth columns of $\mathbf{M}, \mathbf{Q} \in \mathbb{R}^{N \times K}$.The number of nodes of the network is N, and we use $\mathbf{M}$ and $\mathbf{Q}$ to decompose the $N \times N$ amount of parameters of a_{ij} with the expression $\mathbf{a} = \mathbf{M}\mathbf{Q}^T$. The SINN model sets the hyperparameter $K \ll N$, which dramatically reduces the number of parameters from $\mathcal{O}(N^2)$ to $\mathcal{O}(KN)$. Translating this formula into a GNN model yields

$$\begin{aligned} \mathbf{e}^{(l)}_{(i,j)} &= f^{(l)}_{\text{message}}(\mathbf{v}^{(l)}_i, \mathbf{v}^{(l)}_j) = \mathbf{m}_i^T \mathbf{q}_j \mathbf{v}^{(l)}_i \,, \\ \mathbf{v}^{(l+1)}_j &= f^{(l)}_{\text{aggregate}}(\{\mathbf{e}^{(l)}_{(i,j)} | v_i \in \mathcal{N}_j\}) = \sum_{v_i \in \mathcal{N}_j} \mathbf{e}^{(l)}_{(i,j)} \,. \end{aligned} \tag{4.13}$$

When $L_{\text{ODE}} = 1$, our graph-ODE function is identical to the SINN DeGroot ODE function.

- The Friedkin–Johnsen (FJ) model is an opinion model that considers accounts with different susceptibilities to interpersonal influence. Some researchers also use the term "stubborness" to describe FJ model's incorporation of nodes with hesitance to change their opinions [153]. The discrete-time FJ update rule is

$$\mathbf{z}^{t+1}_j = (1 - s_j)\mathbf{z}^0_j + s_j \sum_{v_i \in \mathcal{N}_j} \mathbf{z}^t_i \,,$$

where $s_j \in [0, 1]$ is the susceptibility to persuasion. A smaller value of s_j signifies that account v_j is harder for others to influence. All neighbors in a network have the same importance. Okawa and Iwata [111] proposed an ODE that is similar to the FJ model. It is

$$\frac{\mathrm{d}\mathbf{z}^t_i}{\mathrm{d}t} = s_j \sum_{v_i \in \mathcal{N}_j} \mathbf{z}^t_i + (1 - s_j)\mathbf{z}^0_j - \mathbf{z}^t_j \,.$$

When $L_{\text{ODE}} = 1$, by extending the dimensionality of $\mathbf{z}^t_j$ from one dimension (1D) to d dimensions (dD), we obtain a GNN version of FJ ODE. We do not include the decay term $-\mathbf{z}^t_j$. This yields our GNN FJ model

$$\begin{aligned} \mathbf{e}^{(l)}_{(i,j)} &= f^{(l)}_{\text{message}}(\mathbf{v}^{(l)}_i, \mathbf{v}^{(l)}_j) = \mathbf{s}_j \odot \mathbf{v}^{(l)}_i \,, \\ \mathbf{v}^{(l+1)}_j &= f^{(l)}_{\text{aggregate}}(\{\mathbf{e}^{(l)}_{(i,j)} | v_i \in \mathcal{N}_j\}) = \sum_{v_i \in \mathcal{N}_j} \mathbf{e}^{(l)}_{(i,j)} + (\mathbb{1} - \mathbf{s}_j) \odot \mathbf{v}^{(0)}_j \,, \end{aligned} \tag{4.14}$$

where $\mathbf{x} \odot \mathbf{y}$ denotes element-wise multiplication. For any node v_j, the susceptibility value and the hidden value of the node (in each layer of the GNN function) have the same dimensionality (i.e., $\mathbf{s}_j, \mathbf{v}^{(l)}_j \in \mathbb{R}^d$).

- The Hegselmann–Krause (HK) model is a bounded-confidence model (BCM) [12]. Bounded-confidence models consider "confirmation bias" and "selective exposure", which entails that agents tend to pay more attention to information that confirms their preconceptions [29, 133]. The discrete-time HK model is

$$\mathbf{z}_j^{t+1} = \mathbf{z}_j^t + \frac{1}{|N_j(t)|} \sum_{i \in \Gamma_j(t)} (\mathbf{z}_i^t - \mathbf{z}_j^t)\,,$$

  where $\Gamma_j(t) = \{v_i \in \mathcal{N}_j \mid |\mathbf{z}_i^t - \mathbf{z}_j^t| \leq \delta\}$ is the set of neighboring nodes whose opinions are within the confidence bound of the target node j (i.e., $|\mathbf{z}_i^t - \mathbf{z}_j^t| \leq \delta$). Okawa and Iwata [111] used a continuous-time HK model in the form of the ODE

$$\frac{\mathrm{d}\mathbf{z}_i^t}{\mathrm{d}t} = \sum_{v_i \in \mathcal{N}_j} \sigma\Big(\delta - |\mathbf{z}_i^t - \mathbf{z}_j^t|\Big)\Big(\mathbf{z}_i^t - \mathbf{z}_j^t\Big)\,,$$

  where $\mathbf{z}_j^t \in \mathbb{R}^1$ and $\sigma(z) = 1/(1 + e^{\gamma z})$. In our scenario, we extend the agent opinions from 1D to dD, with $\mathbf{z}_j^t \in \mathbb{R}^d$, and assume that an agent's attention to all different topics (with one topic in each dimension of the d-dimensional opinion space) is limited. Therefore, we use the softmax smooth function and write

$$\begin{aligned} \mathbf{e}_{(i,j)}^{(l)} &= f_{\text{message}}^{(l)}(\mathbf{v}_i^{(l)}, \mathbf{v}_j^{(l)}) = \boldsymbol{\gamma} \odot \text{Softmax}\Big(\boldsymbol{\xi} \odot (\boldsymbol{\delta} - |\mathbf{v}_i^t - \mathbf{v}_j^t|)\Big) \odot \Big(\mathbf{v}_i^t - \mathbf{v}_j^t\Big)\,, \\ \mathbf{v}_j^{(l+1)} &= f_{\text{aggregate}}^{(l)}(\{\mathbf{e}_{(i,j)}^{(l)} | v_i \in \mathcal{N}_j\}) = \sum_{v_i \in \mathcal{N}_j} \mathbf{e}_{(i,j)}^{(l)}\,, \end{aligned} \tag{4.15}$$

  where $\odot$ represents element-wise multiplication and $\boldsymbol{\xi}, \boldsymbol{\delta}, \boldsymbol{\gamma} \in \mathbb{R}^d$, which are the same for all agents, are parameters to learn. In practice, on our real-world data sets, we found softmax smoothing function performs significantly better than the sigmoid function that SINN used for 1D opinion updates.

To demonstrate the importance of the ODE functions, we also use a naive baseline ODE model, which we call No-Update. This model is

$$\begin{aligned} \mathbf{e}_{(i,j)}^{(l)} &= f_{\text{message}}^{(l)}(\mathbf{v}_i^{(l)}, \mathbf{v}_j^{(l)}) = \mathbf{v}_j^{(l)}\,, \\ \mathbf{v}_j^{(l+1)} &= f_{\text{aggregate}}^{(l)}(\{\mathbf{e}_{(i,j)}^{(l)} | v_i \in \mathcal{N}_j\}) = \frac{1}{|\mathcal{N}_j|} \sum_{v_i \in \mathcal{N}_j} \mathbf{e}_{(i,j)}^{(l)}\,. \end{aligned} \tag{4.16}$$

In this case, the ODE has the same opinion prediction as the initial opinion $\mathbf{v}_j^0$ for each node v_j for all times t.

4.4.3 The Time-Dependent Decoder

We consider two tasks: (1) interaction inference (i.e., "prediction") and (2) polarity inference (i.e., "prediction"). As we discussed in Section 4.3.2, each of the downstream tasks has a different decoder design and different objectives.

Previous neural dynamical systems [67, 77] jointly train the encoder, a reconstruction decoder, and an ODE by maximizing the evidence lower bound (ELBO)

$$\mathrm{ELBO}(\theta, \phi) = \mathbb{E}_{\mathbf{Z^0} \sim q_\phi(\mathbf{Z^0}|o_1, o_2, \ldots, o_N)}[\log p_\theta(o_1, o_2, \ldots, o_N)] - \mathrm{KL}[q_\phi(\mathbf{Z^0}|o_1, o_2, \ldots, o_N) \parallel p(\mathbf{Z^0})] \,, \tag{4.17}$$

where the prior $p(\mathbf{Z^0})$ is the standard normal distribution $\mathcal{N}(\mathbb{0}, \mathbb{1})$. The vectors $\mathbb{0}$ and $\mathbb{1}$ are all-zero and all-one vectors, respectively. The quantity $\mathrm{KL}[q_\phi \parallel p]$ is the Kullback–Leibler (KL) divergence between the distribution q_ϕ and the distribution p [169]. Previous neural dynamical systems have considered an observation-reconstruction task [67, 77]. However, real-world social networks involve many uncertainty. It is unreasonable to suppose that we can predict exactly what the accounts post. Therefore, we seek different decoder tasks than observation–reconstruction.

We use a similar framework as in Equation (4.17). We break the loss value into two parts. The first part estimates the quality of the decoder's prediction, and the second part is a regularization term. The loss value is

$$\mathcal{L} = \mathcal{L}_{\mathrm{dec}} + \lambda \mathcal{L}_{\mathrm{reg}} \,, \tag{4.18}$$

where $\mathcal{L}_{\mathrm{reg}} = \mathrm{KL}[q_\phi(\mathbf{Z^0}|o_1, o_2, \ldots, o_N) \parallel p(\mathbf{Z^0})]$. The term $\mathcal{L}_{\mathrm{dec}}$ is different for different tasks.

4.4.3.1 Interaction Prediction

We model the interaction-prediction task as a time-dependent edge-prediction problem. We assign an index $R \in \{1, 2, 3, 4\}$ to the 4 types of possible interactions: reply, mention, retweet, and like. For a temporal edge $\langle i, j, t \rangle$ with relation $r \in \{1, 2, 3, 4\}$, node indices $i, j \in \{1, 2, \ldots, N\}$, and time $t \in [0, T]$ (where T is the ending time of the observations), we measure the score of the edge's existence with the function

$$\text{edge\_score}(i, j, t, r) = \mathbf{z}_i^t \mathbf{W}_r \mathbf{z}_j^t + \mathbf{W}_v \begin{bmatrix} \mathbf{z}_i^t \\ \mathbf{z}_j^t \end{bmatrix} + b\,. \tag{4.19}$$

This choice extends the TIMME-NTN module [158] (see Section 2.4.2) for edge prediction to temporal edges. In Equation 4.19, $\mathbf{W}_r \in \mathbb{R}^{d \times d}$, $\mathbf{W}_v \in \mathbb{R}^{2d}$, and $b \in \mathbb{R}$ are trainable parameters of the decoder. The matrix $\mathbf{W}_r$ is diagonal.

When the temporal edge $\langle i, j, t \rangle$ exists, the ground-truth value of the edge score is 1; when it does not exist, the ground-truth value is 0. We optimize edge_score(i, j, t, r) using the ground-truth values by minimizing the binary-cross-entropy loss, which we thus use for $\mathcal{L}_{\text{dec}}$. We evaluate the models' performance on the test data set by using the ROC-AUC and by the mean precision. The mean precision is the area under the curve (AUC) of the precision–recall (PR) curve, so we also refer to it as PR-AUC.

4.4.3.2 Polarity Prediction

We model the polarity-prediction task as a regression problem. We label the ground-truth political-polarity score using the polarity scores that we compute with PEM (see Chapter 3) [159].

We feed x_i^t into the pipeline of the pretrained PEM model sentence by sentence. We calculate the mean value of the polarity score of all tokens in a sentence as a sentence-level polarity score. We regard the mean value of all sentences' polarity scores as a ground-truth polarity score p_i^t.

For any text-content observation x_i^t of account v_i at time $t > 0$, the corresponding latent opinion observation $\mathbf{z}_i^t$ is available. We then use a 2-layer MLP network and output a 1D score

$$\hat{p}_i^t = \text{MLP}(\mathbf{z}_i^t)$$

as the predicted polarity value. We compute the loss component $\mathcal{L}_{\text{dec}}$ as a negative Gaussian log-likelihood between the prediction $\hat{p}_i^t \in \mathbb{R}$ and the ground truth $p_i^t \in \mathbb{R}$. We evaluate the quality of our prediction by calculating the mean-square error (MSE) and mean absolute-percentage error (MAPE) between all pairs of $\hat{p}_i^t$ and p_i^t.

4.5 Experiments

In this section, we talk about how we designed our experiments in details, and discuss the experimental results of our **LSDS** framework.

4.5.1 Data Sets

We collected our data set from Twitter using the Twitter (currently named $\mathbb{X}$) API.[2] We collected the data by the end of the year 2020.[3] Back then, the Twitter API allowed access to the most recent 3,200 tweets in each account's timeline.

Although some accounts' records can be traced back to 2019 or even earlier, we align the starting time and the ending time of their timelines and keep only the tweets that were posted in 2020.

4.5.1.1 Selecting the Subset of Accounts

We use the TIMME data set's collection of politicians' accounts [158], which consists of Congress members, cabinet members, and president candidates at that time. After filtering out the accounts that were no longer available at the time that we collected them and removing the accounts that have posted fewer than two tweets during the year 2020, we keep 513 politician accounts.

The 513 politician accounts are the nodes in our **T**witter 20**20** **D**ynamical data of **Politicians** (**T20D-Politicians**) data set. For our comparisons, we set the number of accounts in all other data sets to 513.

[2] See `https://developer.twitter.com/en/docs/twitter-api` for more information.

[3] The last valid record of post was on November 26th.

We select 256 politicians' accounts (approximately 50% of the politicians' accounts) uniformly at random and include some of their neighbors according to the follower–followee relationships in the TIMME data set. The total number of nodes that we include in every data set is always 513, with 256 politicians and 257 other accounts. We used different criteria to select the followers and followees. We order the followers and followees of the selected politicians' accounts in the TIMME data set by how many tweets they posted during the year 2020.

We keep a list of the selected politicians' followers and followees as the candidate accounts. We select the most active (i.e., those that have posted the most tweets) candidate accounts to make the **T20D-Active** data set (i.e., **T**witter 20**20** **D**ynamical data of **Active** accounts). We select the most inactive (i.e., those that have posted the fewest tweets) candidate accounts to make the **T20D-Inactive** data set. We also select the candidate accounts from the TIMME data set uniformly at random to form the **T20D-Random** data set.

4.5.1.2 From Observations to Representations

We can observe only an account's tweets. Instead of having a ground-truth representation $\mathbf{o}_i^t$, we observe the text x_i^t that an account v_i posts on Twitter at time t. For our tasks, we use the SENTENCE-BERT model [125]. We use the "all-MiniLM-L6-v2" version of its pretrained parameters to translate text sentences into vector embeddings. The SENTENCE-BERT model has been tested on benchmarks such as SENTEVAL [31], and it has performed better on sentence-level embeddings [125] than other popular language models such as BERT [41] and ROBERTA [90].

We transform the observations $\tilde{X}_i^t = \{x_i^{\tilde{t}} | \tilde{t} \in [t_{\text{start}}, t_{\text{end}}]\}$ of account v_i's tweets on a certain day into vector-space representations $\tilde{O}_i^t = \{\tilde{\mathbf{o}}_i^{\tilde{t}} | \tilde{t} \in [t_{\text{start}}, t_{\text{end}}]\}$, and we then take a mean of these representations. This yields our vector-space representation of account v_i at day t. It is given by

$$\mathbf{o}_i^t = \frac{1}{|\tilde{O}_i^t|} \sum_{\tilde{t} \in [t_{\text{start}}, t_{\text{end}}]} \tilde{\mathbf{o}}_i^{\tilde{t}} \,. \tag{4.20}$$

In practice, we assume that humans are unlikely to change their opinions very frequently. There is no need to measure an account's opinion at every second. Accordingly, we use a temporal granularity of one day.

4.5.1.3 Constructing A Dynamic Network

We assume that the follower–followee relationships of each **T20D** node set (see Section 4.5.1.1) never change. However, the nodes' latent opinions continuously change.

To train our model effectively, we need multiple trajectories of samples, so we need to construct multiple dynamic graphs for each data set. We select 50 different sequences of observations from the 2020 tweets. The time window of different sequences that we select can overlap with each other, and each sequence's time range includes at least 20 observations time points. We select these numbers based on experience in other neural-ODE projects [67].

For each tweet, we record the accounts that retweet this tweet, which accounts are mentioned by this tweet, which accounts (if any) the tweets are replying to, and which accounts like the tweet. We keep only accounts that are included in our node set (e.g., **T20D-Active** node set). When we combine all observations from a particular day, we merge all of the interactions of all tweets on that day into the observed interactions.

We split all data sets into training and testing sets with an 8:2 ratio. To reduce data leakage, we use the earlier observations in a timeline as a training sequence and the later ones as testing sequences. The different data sets have different numbers of follower–followee relationships (see Table 4.1).

We see in Table 4.1 that the testing set of **T20D-Active** is significantly denser than its training set.[4] The reason is that we use earlier times for the training sets and later times for the testing sets. Additionally, in 2020, Twitter allowed access to only the most recent 3,200 tweets of each account's timeline. If some accounts are extremely active (i.e., they post many tweets per day), then we may only have their tweets during a short time period

[4]Similar problem exists on other data sets (i.e., **T20D-Inactive** and **T20D-Random**) but typically less severe.

Table 4.1: Overview of the size of our training and testing data sets. The quantity "# Seq" denotes the number of sequences of a dynamic network. The quantity "Max N" denotes the maximum number of candidate nodes in this network. The quantity "Mean N" refers to the mean number of nodes in a network, and "Mean M" refers to its mean number of edges.

Data set name	Split	# Seq	Max N	Mean N	Mean M	Feature dimensions
T20D-Politicians	train	40	513	462.35	239,450.73	768
	test	10	513	467.90	223,263.10	768
T20D-Active	train	40	513	334.83	72,565.30	768
	test	10	513	478.30	102,114.60	768
T20D-Random	train	40	513	341.03	56,198.83	768
	test	10	513	395.50	61,431.80	768
T20D-Inactive	train	40	513	183.55	42,475.68	768
	test	10	513	185.90	40,093.60	768

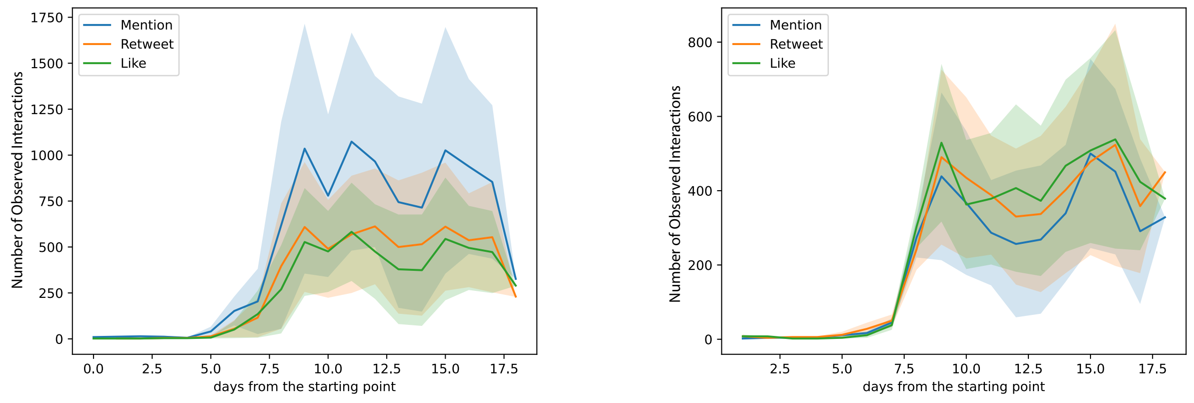
(e.g., perhaps a few months or less). This size imbalance between training sets and testing sets is negligible in data sets (e.g., **T20D-Inactive**) in which we select the accounts that do not post tweets very frequently.

4.5.1.4 Limitations

Because of Twitter API's accessibility, the most recent behaviors are preserved better than the earlier behaviors. We examine the numbers of observations each day for mentions, retweets, and likes. We plot their mean values and the standard deviations in Figure 4.3.

4.5.2 Model Performance

We test the different variants of our **LSDS** framework and compare their performance. For each setting, we use 3 different random seeds and take the mean and standard deviation of the results. Table 4.2 shows the results of our temporal-relationship prediction task, and Table 4.3 shows the results of our political-polarity regression task. Our data sets are denser than many other data sets, such as knowledge graphs (e.g., the FB15K data set has as many as 14,951 nodes but only 592,213 edges [15]) and chemical graphs (e.g., the MUTAG data



(a) **T20D-Politicians** data set (b) **T20D-Random** data set

Figure 4.3: The number of observed interactions after different numbers of days. The solid curves indicate the mean values, and the filled areas indicate the standard deviations.

Table 4.2: Performance on our temporal interaction-prediction task (we predict the next 18 days). We show mean results and their standard deviations. By default, most researchers use GCN models with 2 convolution layers (i.e., $L_{\text{ENC}} = 2$). We show the best results in bold.

Model Variants	**T20D-Politicians**		**T20D-Active**		**T20D-Random**		**T20D-Inactive**	
	ROC-AUC	AVG-PR	ROC-AUC	AVG-PR	ROC-AUC	AVG-PR	ROC-AUC	AVG-PR
LSDS with other encoder variants								
GCN ($L_{\text{ENC}} = 2$)	$.7400 \pm .0330$	$.7190 \pm .0342$	$.7219 \pm .0100$	$.7196 \pm .0108$	$.6614 \pm .0093$	$.6459 \pm .0120$	$.7266 \pm .0158$	$.7117 \pm .0205$
GCN ($L_{\text{ENC}} = 3$)	$.7308 \pm .0172$	$.7033 \pm .0175$	$.7274 \pm .0110$	$.7260 \pm .0136$	$.7576 \pm .0071$	$.7622 \pm .0054$	$.8326 \pm .0317$	$.8163 \pm .0271$
LSDS with other ODE function variants								
DEGROOT	$.8354 \pm .0170$	$.8203 \pm .0198$	$.8797 \pm .0042$	$.8843 \pm .0016$	$.8849 \pm .0088$	$.8768 \pm .0118$	$.8726 \pm .0123$	$.8704 \pm .0088$
FJ	**.8621** $\pm .0069$	**.8502** $\pm .0127$	$.8736 \pm .0041$	$.8736 \pm .0036$	$.8564 \pm .0066$	$.8587 \pm .0045$	$.8845 \pm .0063$	**.8872** $\pm .0055$
BCM (HK)	$.8302 \pm .0217$	$.8190 \pm .0235$	$.8768 \pm .0104$	$.8827 \pm .0075$	$.8792 \pm .0049$	$.8715 \pm .0065$	$.8646 \pm .0041$	$.8615 \pm .0103$
NO-UPDATE	$.7546 \pm .0721$	$.7526 \pm .0565$	$.8392 \pm .0148$	$8357 \pm .0155$	$.7855 \pm .0657$	$.7673 \pm .0726$	$.8091 \pm .0685$	$.8078 \pm .0636$
LSDS	$.8558 \pm .0136$	$.8427 \pm .0118$	**.8849** $\pm .0022$	**.8881** $\pm .0052$	**.8987** $\pm .0015$	**.8925** $\pm .0016$	**.8888** $\pm .0062$	$.8856 \pm .0059$

set has 17.93 nodes on average but only 19.79 edges on average [38, 163]). Therefore, instead of assuming that there is no interaction between two nodes that we sample uniformly at random and then use the edge as negative sample, we select node pairs that do not interact with each other according to a specified relationship (e.g., retweet) at a particular time to construct our negative samples. We use the negative edge samples to train the relation prediction task. We call this process “removing false-negative interactions”. This treatment is both reasonable and essential, but it roughly doubles the computation time.

As we can see in Table 4.2, for the temporal-edge prediction task, our **LSDS** model performs the best on almost all data sets, except for the **T20D-Politicians** data set. On this data set, using the FJ ODE function achieves the best performance. We suspect that the FJ model’s takes the agents’ susceptibilities into consideration makes it predict more precisely on the politicians’ behaviors.

Researchers typically use GCNs with two convolution layers (i.e., $L_{\text{ENC}} = 2$). However, we find that a 3-layer GCN ($L_{\text{ENC}} = 3$) performs well on the temporal edge-prediction task, so we show results for this variant. We do not obtain a similar finding for the other decoder tasks. That is, for our other two tasks, the default 2-layer GCN outperforms a 3-layer GCN.

Table 4.3: Performance on our temporal polarity-prediction task (we predict the next 18 days). We show mean results and their standard deviations. We show the best results in bold.

Model Variants	**T20D-Politicians**		**T20D-Active**		**T20D-Random**		**T20D-Inactive**	
	MSE	MAPE	MSE	MAPE	MSE	MAPE	MSE	MAPE
LSDS with other encoder variants								
GCN	0.371 ± 0.008	1.33 ± 0.09	0.243 ± 0.003	1.95 ± 0.14	0.383 ± 0.020	$\mathbf{1.35} \pm 0.06$	0.361 ± 0.009	1.27 ± 0.09
LSDS with other ODE function variants								
DeGroot	0.338 ± 0.008	1.35 ± 0.03	0.195 ± 0.011	2.21 ± 0.10	0.331 ± 0.002	1.39 ± 0.03	0.331 ± 0.002	$\mathbf{1.20} \pm 0.02$
FJ	0.320 ± 0.002	1.65 ± 0.08	0.205 ± 0.007	2.17 ± 0.10	0.331 ± 0.002	1.58 ± 0.08	0.326 ± 0.002	1.34 ± 0.07
BCM (HK)	0.365 ± 0.013	$\mathbf{1.25} \pm 0.05$	0.195 ± 0.003	1.77 ± 0.32	0.349 ± 0.001	1.42 ± 0.02	0.348 ± 0.005	$\mathbf{1.20} \pm 0.00$
No-Update	0.361 ± 0.011	1.75 ± 0.11	0.200 ± 0.004	3.39 ± 0.63	0.351 ± 0.002	1.88 ± 0.17	0.350 ± 0.003	1.45 ± 0.07
LSDS	$\mathbf{0.305} \pm 0.001$	1.82 ± 0.37	$\mathbf{0.168} \pm 0.001$	$\mathbf{1.64} \pm 0.05$	$\mathbf{0.318} \pm 0.005$	1.60 ± 0.30	$\mathbf{0.299} \pm 0.005$	1.57 ± 0.08

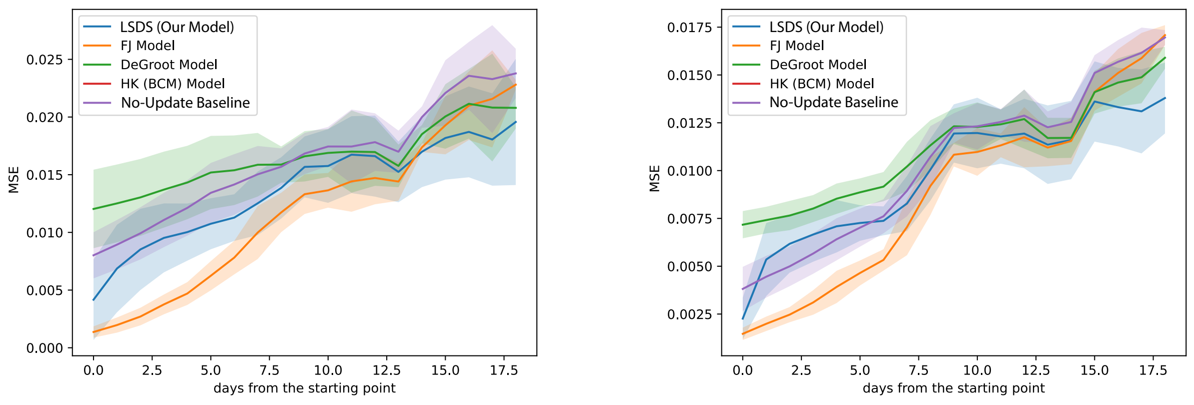
In Table 4.3, we show the overall performance on the political-polarity regression task. Although our model performs well in general, all models perform similarly to each other. This finding is consistent with the fact that individuals tend to maintain consistent opinions along a liberal–conservative axis.

4.5.3 Case Studies

Thus far, the temporal interaction-prediction task has been the most discriminative decoder task. However, treating all treating all testing-set samples equally is not enough in practice. We are also interested in exploring how well each model can look into the near future versus the far future.

Additionally, although we optimized scores on all types of relations together due to the sparsity of some types of relations, it is interesting to examine whether or not we observe different interaction-prediction performance for different types of relationships over time.

4.5.3.1 Which Model Performs Better in the Long Term?

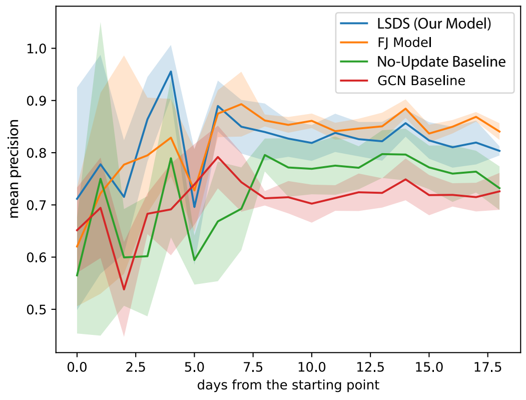


(a) **T20D-Politicians** Data Set (b) **T20D-Random** Data Set

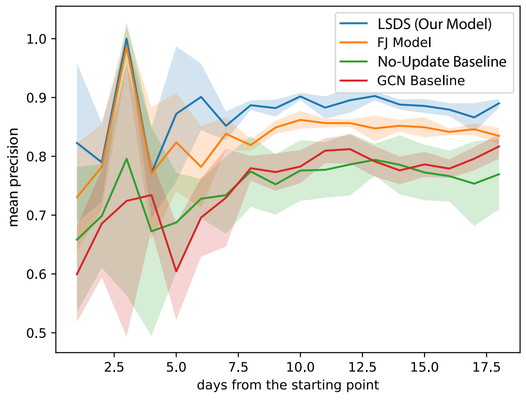
Figure 4.4: The mean-squared errors of the predictions of political-polarity scores after different numbers of days. The solid curves indicate the mean values, and the filled areas indicate the standard deviations.

Although all neural-ODE functions perform similarly on our polarity-prediction task (see Table 4.3), we observe that our **LSDS** model is almost always better at making longer-term predictions (see Figure 4.4). If we generate a data set to include longer time ranges, perhaps our **LSDS** model will perform better. This is an interesting experiment to perform in the future. See Section 4.6 for a discussion of this idea and other future work.

In Figure 4.4, we see from the performance of the No-Update baseline that if we assume that everybody always retains the same opinion, then our models make more mistakes as we consider progressively longer time horizons. This trend is more prominent in the No-Update model than in the other models.



(a) **T20D-Politicians** data set



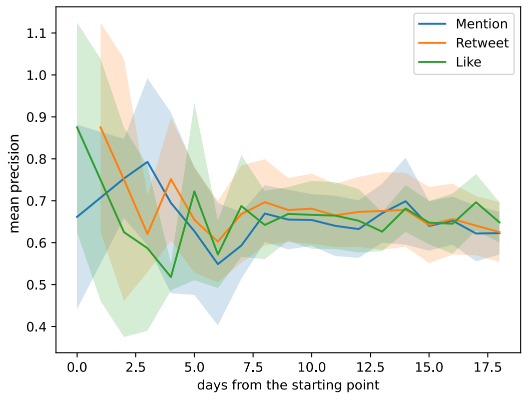
(b) **T20D-Random** data set

Figure 4.5: The mean precisions (i.e., PR-AUCs) of the predictions of temporal interactions after different numbers of days. The solid curves indicate the mean values, and the filled areas indicate the standard deviations.

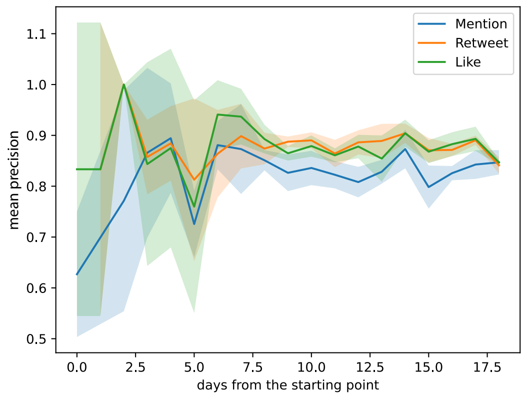
In Figure 4.5, we show the performance on our temporal-interaction prediction task for different time horizons. In this comparison, we include our **LSDS** model, the FJ model (which performs unexpectedly well, especially on the **T20D-Politicians** data set), the No-Update baseline, and an ordinary GCN.

From these figures, we find that unlike the political-polarity score, which is significantly harder to predict in the long-term than in the short-term, most of the models' performances on predicting the interactions between accounts stabilizes after about 10 days. It may be that different tweets have very different tweet contents, even if they comes from the same account, hence that it is less likely for a feature to stabilize, whereas behavior in a social network does not tend to change much in a short period of time (e.g., in one month). For example, perhaps people retweet and mention many of the same accounts in a short period of time.

However, from the poor performance of the NO-UPDATE baseline, we conclude that there is still some pattern of changes in the temporal interactions. In general, capturing long-term behaviors is more challenging than capturing short-term behaviors. We believe that the models' performance on long-term predictions depends on how well they can capture the more consistent interaction-patterns between accounts. As one can see in the significant difference between Figure 4.5a and Figure 4.5b, we suspect that politicians and other types of accounts have different interaction patterns. Additionally, from Figure 4.5, we suspect that our data sets are not challenging enough for our models. They all seem to do too well at predicting future dynamics. We need to test them more stringently in future works.



(a) **LSDS** model

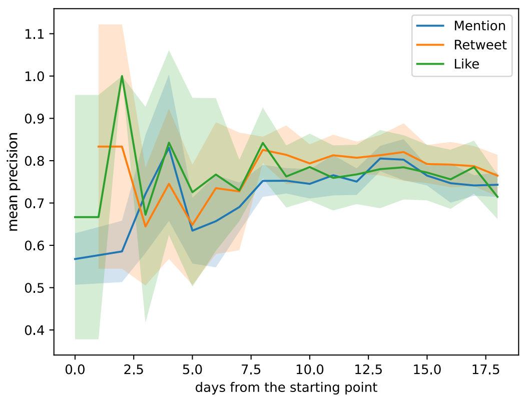


(b) FJ model

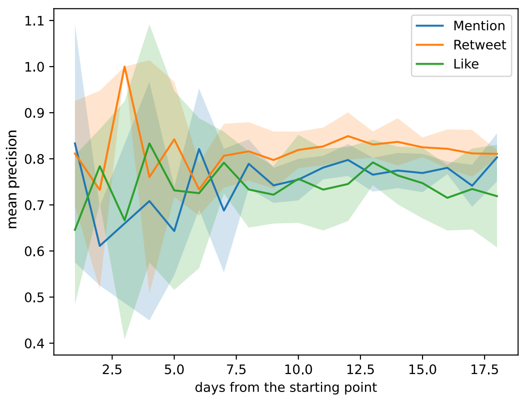
Figure 4.6: The mean precisions (i.e. PR-AUCs) of the predictions of different types of temporal interactions in the **T20D-Politicians** data set after different numbers of days. The solid curves indicate the mean values, and the filled areas indicate the standard deviations.

One may ask why our Figures 4.5 (and also Figure 4.6 discussed in the next subsection) are so wiggly in the first few days, even though it should be easier generically to predict the near future than farther in the future. We suspect that this observation arises from the construction of our data set. As we mention in Section 4.5.1.4, the distribution of our observations is uneven in time. This feature of our data set can cause the predictions in the first few days to be less reliable than later predictions.

Intuitively, one expects model performance to worsen as we consider times that are farther away from the initial time $t = 0$. However, this is not the case for our edge-prediction results. We suspect that this is due to the powerful temporal GNN encoder component. As we show in Figure 4.7, even the No-Update model, in which there is essentially no ODE and we let $\mathbf{z}_i^t = \mathbf{z}_i^0$ for all nodes v_i, the performance on our interaction-prediction task is somewhat stable.



(a) **T20D-Politicians** data set



(b) **T20D-Random** data set

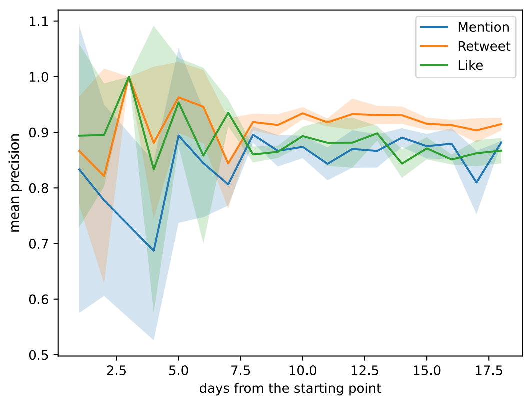
Figure 4.7: The mean precisions (i.e. PR-AUCs) of the predictions of temporal interactions using the No-Update baseline. The performance of No-Update baseline relies heavily on the quality of the encoder component.

4.5.3.2 Which Relationship is the Easiest One to Predict?

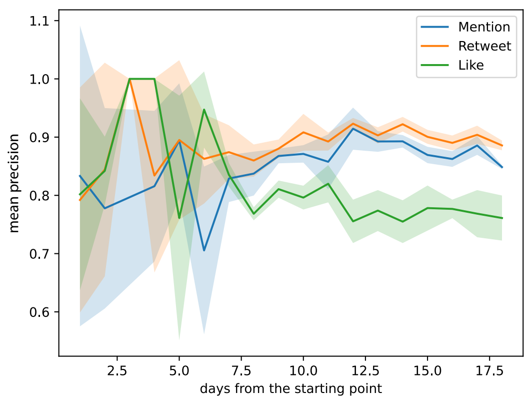
In the relationship set {reply, mention, retweet, like}, the hardest type of relationship to predict is replies. Replies are also the least reliable relationship, given how sparse it is. Most of the time, we are not even able to plot the performance on predicting replies because of the data sparsity.

All other types of relationships have enough data to use for reasonable predictions. The sparsity of the reply-relationship data is also why we measure prediction success using PR-AUC instead of ROC-AUC. PR-AUC is better than ROC-AUC on handling imbalanced data sets [36].

According to Figure 4.6, the FJ model consistently performs better than our **LSDS** model for predicting all types of interactions. All other models perform similarly well for predictions of all relationships between the accounts of politicians. A model that performs well at predicting mentions is likely to perform similarly at predicting retweets. The FJ model performs slightly better than the others, but it achieves different performances for different relationships. It seems that, when using the FJ model, it is easiest to predict the politicians' likes and retweets and that it is hardest to predict their mentions.



(a) **LSDS** model



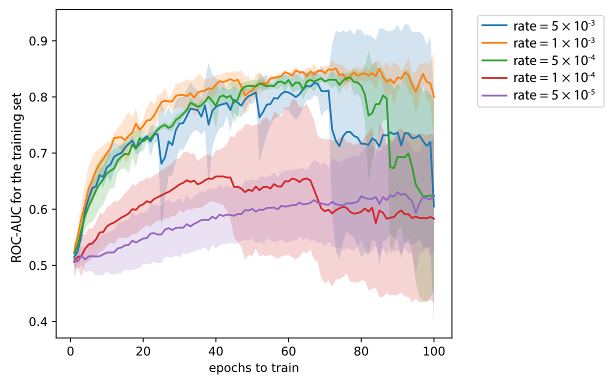
(b) FJ model

Figure 4.8: The mean precisions (i.e., PR-AUCs) of the predictions of temporal-interactions for the **T20D-Random** data set after different numbers of days for different types of interactions. The solid curves indicate the mean values, and the filled areas indicate the standard deviations.

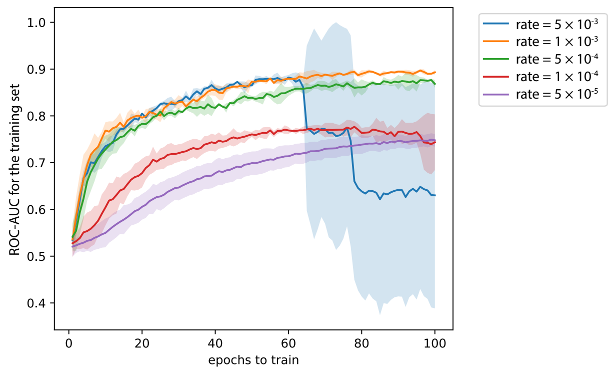
We obtain different results for other sets of accounts (e.g., **T20D-Random**. In Figure 4.8, we see that **LSDS** is successful and stable at predicting retweet interactions in a set of accounts that consists of 50% politicians and 50% of ordinary followees and followers of politicians. An advantage of our **LSDS** model is that it achieves a similar performance for all relationships. For the FJ model (see Figure 4.8b), it is significantly harder to predict like interactions than to predict the other types of interactions.

4.5.4 Training

It is very tricky to train our **LSDS** model. We gradually increase the KL coefficient λ (see Section 4.4.3 and Equation (4.21)) as the training proceeds, there exist many local optima, so the training performance can easily drop.



(a) **T20D-Politicians** data set



(b) **T20D-Random** data set

Figure 4.9: The ROC-AUCs of our **LSDS** model for the training set after each epoch for various learning rates. We observe similar trends for the testing set. Larger loss values indicate worse performance.

As we see in Figure 4.9, the most suitable learning rate lr when we train for our temporal-relationship-prediction task for 100 epochs is $lr = 1 \times 10^{-3}$. The learning rate influences whether or not our model becomes trapped in a bad local optimum. Additionally, the training curve depends on the data set, as we can see by comparing Figures 4.9a and 4.9b.

In Figure 4.9a, we see that the learning curve for the learning rate $lr = 1 \times 10^{-4}$ experiences some instability at about epoch 40, but the optimization performance becomes better at about epoch 60. Later, its performance drops again and then does not improve. This phenomenon is caused by our KL-coefficient scheduler. We compute the KL coefficient

λ using the formula

$$\lambda = \begin{cases} 0\,, & \text{epoch} \leq 10 \\ 1 - 0.99^{\,\text{epoch}-10}\,, & \text{epoch} > 10 \text{ and epoch} \leq 90 \\ 0.6\,, & \text{epoch} > 90\,. \end{cases} \tag{4.21}$$

We adapt this learn-rate schedule for λ from the schedule in LG-ODE [67]. LG-ODE was designed initially for physical systems that typically require fewer than 100 epochs of training, and we add an upper bound of 0.6 to λ so that our model is less likely to experience severe performance drops after 100 epochs.

To stabilize the training curve and enable training for many more epochs without harming the performance, many neural-ODE implementations [24, 42] have used the Cosine Annealing learning-rate scheduler [93]. Additionally, some previous works on VAEs have suggested that the KL coefficient λ can be scheduled in other ways [113]. We have not added the Cosine Annealing learning rate scheduler. We believe that adding this will improve the performance of our **LSDS** model. We find that the learning rate $lr = 1 \times 10^{-4}$ most benefits temporal GNN encoder for our temporal-relation-prediction task. With the baseline GCN encoder, sometimes the best learning rate is 5×10^{-5}. For the political-polarity prediction task (see Table 4.3), the political-polarity classification task (see Table 4.4), and the text-embedding reconstruction task (see Table 4.5), the 5×10^{-5} and 1×10^{-4} learning rates yield similar performance.

4.5.5 Ablation Study

In this subsection, we try different combinations of hyperparameter settings on our **LSDS** framework, and then conduct ablation studies.

4.5.5.1 Hidden-Dimension Sizes

Adding more GNN layers to a graph ODE function does not improve the performance of **LSDS** at our tasks. The original NRI model uses a single-layer architecture to capture the dynamics of a physical system [77]. We have done experiments with a 2-layer GNN in the ODE function, but it does not improve the performance of **LSDS**. For example, for the temporal-interaction-prediction task on the **T20D-Politicians** data set, the ROC-AUC drops slightly to about 85.6% and the mean precision (i.e., PR-AUC) drops to about 84.5%.

Hidden-dimension sizes affect the performance of our **LSDS** model, but they are not as important as other model features. The hidden-dimension size that we use for the graph ODE function is 128 by default. If we increase the hidden-dimension size of the graph ODE function from 128 to 256, for the temporal-interaction prediction task on the **T20D-Politicians** data set, then the ROC-AUC drops to about 83.2% and the mean precision (i.e., PR-AUC) drops to about 81.9%. If we decrease the hidden-dimension size from 128 to 64, the ROC-AUC performance drops to about 84.4% and the mean-precision score (i.e., PR-AUC) drops to about 83.1%. The hidden size we use for the temporal graph encoders is 64 by default. If we increase the hidden size of **LSDS**'s temporal GNN encoder from 64 to 128 and consider the **T20D-Politicians** data set, the ROC-AUC drops to about 85.1% and the mean precision (i.e., PR-AUC) drops to about 83.9%. Decreasing the encoder's hidden-dimension size from 64 to 32 yields a ROC-AUC of about 85.1% and a mean precision (i.e., PR-AUC) of about 84.2%.

Increasing the dimensionality of the latent-opinion representation $\mathbf{z}_i^t \in \mathbb{R}^d$ increases the expressiveness of **LSDS** in theory. However, we need to consider the memory costs and the risk of over-fitting. We set the size of $\mathbf{z}_i^t$ to $d = 16$. With $d = 32$, the performance is about the same (it increases by only 0.01%), with the standard deviation 5–10 times larger than those for $d = 16$. With $d = 64$, overfitting becomes severe. There is an approximately 2% training performance increase, but the testing performance decreases by about 2%. To make our comparisons fair, we use $d = 16$ for all models in our main experiments.

4.5.5.2 The Encoder and Decoder

From Table 4.2, by comparing the performance of the GCN baseline and other baselines, we see for our tasks that having a good encoder is more important than having a good ODE function. It is also important to have a reliable decoder with a reasonable objective function.

The tasks are extremely important when we evaluate the quality of a model. For example, as an extreme case, suppose that we treat the political-polarity-prediction task as a classification task instead of as a regression tasktask. To do this, we write

$$\tilde{p}_i^t = \begin{cases} 1\,, & \text{if } p_i^t > 0 \\ 0\,, & \text{otherwise}\,, \end{cases}$$

where $\tilde{p}_i^t$ is the ground-truth class label. We then change the MLP decoder's output unit to 2 and use a softmax layer as a class selector. We can use the negative log-likelihood function as our loss and evaluate model performance by computing the accuracy and the F_1-score.

Table 4.4: Performance on our temporal polarity-classification task (we predict the next 18 days). We show mean results and their standard deviations. We show the best results in bold.

Model Variants	**T20D-Politicians**		**T20D-Active**		**T20D-Random**		**T20D-Inactive**	
	Accuracy	F_1-Score	Accuracy	F_1-Score	Accuracy	F_1-Score	Accuracy	F_1-Score
LSDS with other encoder variants								
GCN	$.8288 \pm .0093$	$.8269 \pm .0093$	$.7143 \pm .0044$	$.6855 \pm .0050$	$.7114 \pm .0020$	$.7112 \pm .0019$	$.8434 \pm .0041$	$.8427 \pm .0040$
LSDS with other ODE function variants								
DeGroot	$.8595 \pm .0017$	$.8576 \pm .0017$	$\mathbf{.8303} \pm .0010$	$\mathbf{.8256} \pm .0013$	$.7848 \pm .0021$	$.7846 \pm .0022$	$.8639 \pm .0010$	$.8630 \pm .0010$
FJ	$\mathbf{.8600} \pm .0000$	$\mathbf{.8582} \pm .0000$	$.8270 \pm .0014$	$.8224 \pm .0019$	$.7744 \pm .0031$	$.7741 \pm .0031$	$.8582 \pm .0015$	$.8572 \pm .0014$
BCM (HK)	$.8567 \pm .0011$	$.8548 \pm .0011$	$.8285 \pm .0014$	$.8243 \pm .0017$	$.7629 \pm .0078$	$.7626 \pm .0079$	$.8550 \pm .0026$	$.8538 \pm .0026$
No-Update	$.8570 \pm .0009$	$.8552 \pm .0009$	$.8260 \pm .0022$	$.8209 \pm .0027$	$.7847 \pm .0037$	$.7843 \pm .0037$	$.8616 \pm .0013$	$.8606 \pm .0013$
LSDS	$.8582 \pm .0012$	$.8562 \pm .0011$	$.8278 \pm .0010$	$.8232 \pm .0011$	$\mathbf{.7876} \pm .0017$	$\mathbf{.7872} \pm .0018$	$\mathbf{.8644} \pm .0005$	$\mathbf{.8634} \pm .0005$

As we show in Table 4.4, it is hard to distinguish between the performances of the different models. Even the No-Update baseline, which simply sets $\mathbf{z}_i^t = \mathbf{z}_i^0$ for all i and all t, performs very well on our political-polarity-classification task. Although all models perform similarly, the FJ and DeGroot models sometimes outperform the others by a negligible margin.

If we consider an unsupervised setting, where the nodes do not have any training labels, we can use the text-embedding reconstruction task as our decoder task, which is much more similar to the decoder task of the original version of the VAEs, compared to all the other tasks we have. Using the ground-truth future posts of account i at time t, we know its ground-truth future text-embedding observation $\mathbf{o}_i^t$. We can train a decoder to reconstruct $\tilde{\mathbf{o}}_i^t$ from the hidden representation $\mathbf{z}_i^t$. The more $\tilde{\mathbf{o}}_i^t$ and $\mathbf{o}_i^t$ align with each other, the better our social dynamical system is.

The text-embedding reconstruction task is self-supervised, so we do not need node labels for it. By passing $\mathbf{z}_i^t$ through a single linear layer, which performs a linear transformation that adapts the output dimension to the size of $\mathbf{o}_i^t$, we predict the observation

$$\hat{\mathbf{o}}_i^t = \text{Linear}(\mathbf{z}_i^t) \,.$$

We then use the negative Gaussian log-likelihood between the prediction $\hat{\mathbf{o}}_i^t$ and the ground truth $\mathbf{o}_i^t$ as the objective function $\mathcal{L}_{\text{dec}}$. We evaluate the quality of the reconstruction process by taking the mean values of the MSEs and MAPEs between all pairs of $\hat{\mathbf{o}}_i^t$ and $\mathbf{o}_i^t$.

Table 4.5: Performance on our text-embedding-reconstruction task (we predict the next 18 days). We show mean results and their standard deviations. We show the best results in bold.

Model Variants	**T20D-Politicians**		**T20D-Active**		**T20D-Random**		**T20D-Inactive**	
	MSE	MAPE	MSE	MAPE	MSE	MAPE	MSE	MAPE
LSDS with other encoder variants								
GCN	0.034 ± 0.002	5.12 ± 0.51	0.019 ± 0.001	4.19 ± 0.46	0.022 ± 0.002	3.42 ± 0.22	0.025 ± 0.004	3.81 ± 0.36
LSDS with other ODE function variants								
DeGroot	$\mathbf{0.016} \pm 0.000$	$\mathbf{3.15} \pm 0.06$	$\mathbf{0.011} \pm 0.000$	$\mathbf{3.39} \pm 0.19$	$\mathbf{0.016} \pm 0.000$	$\mathbf{3.15} \pm 0.14$	$\mathbf{0.017} \pm 0.000$	$\mathbf{3.29} \pm 0.07$
FJ	0.017 ± 0.000	3.88 ± 0.04	0.012 ± 0.000	3.80 ± 0.37	$\mathbf{0.016} \pm 0.000$	3.67 ± 0.10	$\mathbf{0.017} \pm 0.000$	3.87 ± 0.05
BCM (HK)	0.018 ± 0.000	4.46 ± 0.11	0.012 ± 0.000	3.88 ± 0.20	$\mathbf{0.016} \pm 0.000$	3.44 ± 0.18	$\mathbf{0.017} \pm 0.000$	3.66 ± 0.15
No-Update	0.025 ± 0.009	5.78 ± 0.23	0.018 ± 0.006	6.49 ± 0.46	0.025 ± 0.012	5.80 ± 0.16	0.027 ± 0.008	6.18 ± 0.62
LSDS	$.0164 \pm .0006$	3.45 ± 0.11	$.0111 \pm .0001$	4.10 ± 0.38	$.0150 \pm .0005$	3.06 ± 0.26	$.0167 \pm .0004$	3.49 ± 0.34

In Table 4.5, we see that all models perform very similarly to each other on the text-embedding-reconstruction task. However, the DeGroot model, which was designed to study consensus in multi-agent systems [40], slightly outperforms the rest. The fact that the No-Update baseline performs well on the text-embedding-reconstruction task may help explain why we found it hard to reconstruct text content from the predicted embeddings. A bottleneck of our **LSDS** framework is that its pipeline relies heavily on the quality of the text embedding. With a better strategy than using Sentence-BERT embedding for post-content representation, our framework has the potential to better model a social dynamical system. However, currently, none of the text embedding models that are more powerful than Sentence-BERT are affordable to us.

The above results also demonstrate that the three components of our model — the encoder, the ODE solver, and the decoder — have different impacts on performance. An expressive encoder that is able to generate a proper initial state $\mathbf{z}_i^0$ is essential to achieve good performance. Selecting a reasonable decoder is also important. For different data sets and different target tasks, different choices of ODE solvers can be appropriate. The text-embedding quality also affects the model's quality.

4.5.6 Model Efficiency

Our model has reasonable time and memory efficiency both in theory and in practice. We ran our experiments on a Linux server with the Ubuntu 20.04 operating system, with a 16G RTX A4000 GPU, 32GB memory, and 12 virtual CPUs built on Intel(R) Xeon(R) Gold 5320 CPUs of 2.20GHz clock speed.

We use the standard adjoint method for the neural-ODE solver [178], so the time complexity of the ODE solver is $\mathcal{O}(N_z N_f (N_t + N_r))$, where N_z is the size of each state $\mathbf{z}_i^t$, N_f is the number of hidden layers (i.e., L_{ODE}) in the neural-ODE function, and N_t is the number of function evaluations in the forward pass (one can estimate it roughly as the number of time points at which we make predictions), the quantity N_r is the number of evaluations in the backward pass [179]. To give an example of run time with one of our tasks, running our **LSDS** model for the temporal-edge prediction task on the **T20D-Politicians** data set takes approximately 20–30 seconds per epoch.

The number of parameters depends on the choice of the encoder, the ODE function, and the decoder. For the encoders, our temporal GNN model has 86,592 parameters and the GCN encoder (with $L_{\text{ENC}} = 2$) has 30,944 parameters. In Table 4.6, we show the number of parameters in the ODE function. In Table 4.7 , we show the numbers of parameters in the decoders.

Table 4.6: The numbers of parameters in the ODE functions.

ODE function type	GNN (NRI)	DeGroot	FJ	BCM (HK)	No-Update
Number of parameters	288,752	16,416	41,040	240	0

Table 4.7: The numbers of parameters in the decoders. The label “Rel Pred” stands for relationship-prediction decoder, “Pol Pred” stands for polarity-prediction (i.e., regression) decoder, “Pol Class” stands for polarity-classification decoder, and “Emb Recon” stands for text-embedding-reconstruction decoder.

Decoder type	Rel Pred	Pol Pred	Pol Class	Emb Recon
Number of parameters	196	289	306	6,545

The memory cost of running the neural-ODE solver using the standard adjoint method is $\mathcal{O}(N_z N_f)$, where N_z denotes the parameter size of each state and N_f denotes the number of hidden layers in the neural-ODE function [167, 179]. In practice, the memory cost of our model on our data sets while we run our **LSDS** model is usually less than 5 GB, and the peak memory cost is under 10 GB.

4.6 Conclusion and Future Work

We developed a **L**atent **S**ocial **D**ynamical-**S**ystem (**LSDS**) framework that is able to capture the dynamics of multi-agent dynamical systems on real-world social networks.

Using data from Twitter, we analyzed our model’s performance on a series of experiments on temporal-interaction prediction and political-polarity prediction tasks. **LSDS** performed well on these tasks. Additionally, it is flexible enough to use other encoder, decoder, and ODE-function components, so it can be used on other downstream tasks and data sets.

There are many interesting future research directions to pursue.

Because of the limitations of the collected data, our social-dynamical-system trajectories cover less than one month. As we discussed in Section 4.5.3, we suspect that our model is more accurate at predicting over long time periods than over short time periods, so it is important to examine the performance of **LSDS** on data samples that cover such periods.

In our current encoder framework, we consider only information in a social-media platform. However, in the real world, it is also important to consider news, television shows, and offline discussions. It is a challenging but interesting task to generalize our **LSDS** model to account for such platforms.

It is also desirable to generalize our **LSDS** model to study *adaptive networks* [13], in which nodes and edges can appear, disappear, and change their weights with time. Additionally, although the NRI model provided a successful neural-ODE function for most of our tasks, it is not designed specifically to study social dynamical systems. Other GNNs may be better suited to studies of opinion dynamics and other social phenomena.

It is possible improve the **LSDS** model's training process by improving the learning rate schedule and KL-coefficient schedule (see Section 4.5.4). It may also be useful to use different learning rates for different model components.

After observing that the FJ model does have some advantage over other models at modeling the opinions of politicians' accounts, we believe that it is also promising to combine the update rules of the FJ model and **LSDS** neural-ODE function.

It is also important to consider other decoder tasks. A key example is text reconstruction, which aims to predict the text content that a particular account will produce at a specific future time. Given the flexibility of our **LSDS** model, we also hope to extend it to a multi-task model so it can simultaneously consider multiple downstream tasks.

It is necessary to take ethical concerns seriously when applying our framework and its variants to any task on real-world data sets. For example, as with any other neural network models, our **LSDS** model can inherit biases from training data, such as gender bias. Additionally, even though the **LSDS** model achieves good performance on a few tasks that we have tried, whatever it predicts should never be treated as ground truth about individuals.

Declarations

Funding

This research was supported by the National Science Foundation (NSF 2303037 and NSF 2312501), and NASA Grant (NASA M2300680).

Authors' Contributions

Zhiping Xiao, Prof. Mason A. Porter and Prof. Yizhou Sun conceived and conceptualized the study. Zhiping Xiao, Zijie Huang, Xiao Luo, Yifang Qin, Xinyu Wang performed the analysis. Zhiping Xiao wrote the initial draft of the paper. Zhiping Xiao, Prof. Mason A. Porter, and Prof. Yizhou Sun reviewed and extensively edited the manuscript, determined what additional analysis was necessary, and produced the final version of the manuscript.

Acknowledgements

Thanks to Fang Sun for his insightful suggestions on our work, especially on organizing the related works.

CHAPTER 5

Conclusions and Future Works

In this chapter, we conclude our previous works and discuss our future works.

5.1 Conclusions

This thesis summarizes our works on the Twitter data (i.e., $\mathbb{X}$ data). We have studied the relations modeled as heterogeneous types of links (see Chapter 2), how to extract the political polarity from tweet content (see Chapter 3)

In the first project, we proposed the **TIMME** model [158] to study how we can infer an account's ideology from its interaction with other accounts. In the second project, we proposed the **PEM** model [159] to study how we can infer the political polarity from the accounts' posts. In the third project, we proposed the **LSDS** model to learn the dynamics of a multi-agent social network system.

In fact, the methods we have proposed are not restricted to analyzing political-related data. We can easily migrate their usage to other problems and other data sets with neglectable changes. Our works are universally suitable for analyzing social opinions from any perspectives. Besides, it is not even necessarily limited to the use on analyzing social media data. For example, it is possible that **TIMME/TIMME-hierarchical** [158] (see Chapter 2) be directly applied to other classification tasks on other data sets.

Additionally, we should always keep in mind that all data sets contain biases. In our case, all observations come from the social-media platform Twitter. In other words, even if the limited amount of user accounts we analyze are representative enough as Twitter users, they have no chance of representing those who never use social medias. Conclusions from Twitter data are often different than what one observes with other lenses on the real world [103]. We need to be careful when we draw conclusions according to our data and our methods.

5.2 Future Work

However, despite the insights we get from the previous works, the reasons why such interactions take place still remain mysteries.

We do not have any insight into the actual impact of a certain influential factor yet. For instance, we do not know how to model the impact of retweet-behavior from a certain account. The next project along this line could possibly focus on the study of causality, solving the fourth research problem we propose:

> By learning from the observed link, text, and temporal information, can we reveal the underlying causal relationships among the accounts' posts and behaviors?

How to reveal causal inference on a social network is one of the questions that we are highly interested in. That is a possible future work following our current projects.

One of the key challenges in studying this problem is the lack of ground-truth observations. In other words, lack of counterfactual data. Conducting counter-factor studies online by conducting planned behaviors with all other conditions controlled is neither ethical nor practical.

However, studying causal effect on social network is worthwhile. There have been many existing works revealing the severity of polarization in the real world [110]. On the other hand, most of the efforts on proposing methods to mediate such polarization are more or less on the theoretical side, solving a mathematical problem instead of applying to real-world data sets [96]. If we are able to reveal what is the cause of polarity from the data, it will be super helpful in mediating the ever-growing polarization.

Previous studies have already combined causality inferences and graphical models, being widely viewed as cross-disciplinary studies of computer science and philosophy [37, 53, 81, 139]. Some researchers have pointed out that counterfactual is no longer necessary with help of probabilistic models [37].

BIBLIOGRAPHY

[1] Sergi Abadal, Akshay Jain, Robert Guirado, Jorge López-Alonso, and Eduard Alarcón. Computing Graph Neural Networks: A Survey from Algorithms to Accelerators. *ACM Computing Surveys (CSUR)*, 54(9):191, Oct 2021.

[2] Robert P. Abelson. Mathematical Models of the Distribution of Attitudes under Controversy. In N. Fredericksen and H. Gullicksen, editors, *Contributions to Mathematical Psychology*, pages 147–160. Holt, Rinehart & Winston, 1964.

[3] Christopher H. Achen. Mass Political Attitudes and the Survey Response. *American Political Science Review (APSR)*, 69(4):1218–1231, 1975.

[4] Christopher A. Bail, Lisa P. Argyle, Taylor W. Brown, John P. Bumpus, Haohan Chen, M. B. Fallin Hunzaker, Jaemin Lee, Marcus Mann, Friedolin Merhout, and Alexander Volfovsky. Exposure to Opposing Views on Social Media can Increase Political Polarization. In *Proceedings of the National Academy of Sciences of the United States of America (PNAS)*, volume 115, pages 9216–9221. National Acad Sciences, 2018.

[5] Muhammet Balcilar, Renton Guillaume, Pierre Héroux, Benoit Gaüzère, Sébastien Adam, and Paul Honeine. Analyzing the Expressive Power of Graph Neural Networks in a Spectral Perspective. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021.

[6] Muhammet Balcilar, Guillaume Renton, Pierre Héroux, Benoit Gauzere, Sebastien Adam, and Paul Honeine. Bridging the Gap Between Spectral and Spatial Domains in Graph Neural Networks. *arXiv preprint arXiv:2003.11702*, 2020.

[7] Ramy Baly, Georgi Karadzhov, Abdelrhman Saleh, James Glass, and Preslav Nakov. Multi-Task Ordinal Regression for Jointly Predicting the Trustworthiness and the Leading Political Ideology of News Media. In *Proceedings of the Conference of the*

North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT), volume 1, pages 2109–2116, 2019. (Volume 1: Long and Short Papers).

[8] Pablo Barberá. How Social Media Reduces Mass Political Polarization. Evidence from Germany, Spain, and the U.S. *Job Market Paper, New York University*, pages 1–46, 2014. (Available at http://pablobarbera.com/static/barbera_polarization_APSA.pdf).

[9] Siddharth Batra and D.T.V Dharmajee Rao. Entity Based Sentiment Analysis on Twitter. 2010. class report, Computer Science 224, Stanford University (Available at https://nlp.stanford.edu/courses/cs224n/2010/reports/drao-sidbatra.pdf).

[10] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A Neural Probabilistic Language Model. *Journal of Machine Learning Research (JMLR)*, 3:1137–1155, 2003.

[11] Samy Benslimane, Jérôme Azé, Sandra Bringay, Maximilien Servajean, and Caroline Mollevi. A Text and GNN based Controversy Detection Method on Social Media. *World Wide Web Journal (WWWJ)*, 26(2):799–825, 2023.

[12] Carmela Bernardo, Claudio Altafini, Anton Proskurnikov, and Francesco Vasca. Bounded Confidence Opinion Dynamics: A Survey. *Automatica*, 159:111302, 2024.

[13] Rico Berner, Thilo Gross, Christian Kuehn, Jürgen Kurths, and Serhiy Yanchuk. Adaptive Dynamical Networks. *Physics Reports*, 1031:1–59, 2023.

[14] Adam Boche, Jeffrey B. Lewis, Aaron Rudkin, and Luke Sonnet. The new Voteview.com: Preserving and Continuing Keith Poole's Infrastructure for Scholars, Students and Observers of Congress. *Public Choice*, 176(1–2):17–32, 2018.

[15] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating Embeddings for Modeling Multi-relational Data. In C. J.

Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, volume 26, 2013.

[16] Avishek Joey Bose and William L. Hamilton. Compositional Fairness Constraints for Graph Embeddings. In *Proceedings of the International Conference on Machine Learning (ICML)*, volume 97 of *Proceedings of Machine Learning Research (PMLR)*, pages 715–724, 09–15 Jun 2019.

[17] Heather Z. Brooks, Philip S. Chodrow, and Mason A. Porter. Emergence of polarization in a sigmoidal bounded-confidence model of opinion dynamics. *arXiv preprint arXiv:2209.07004*, 2022. (SIAM Journal on Applied Dynamical Systems, in press).

[18] Heather Z. Brooks and Mason A. Porter. A Model for the Influence of Media on the Ideology of Content in Online Social Networks. *Physical Review Research*, 2:023041, 2020.

[19] Gavin C. Cawley and Nicola L. C. Talbot. On Over-fitting in Model Selection and Subsequent Selection Bias in Performance Evaluation. *Journal of Machine Learning Research (JMLR)*, 11:2079–2107, 2010.

[20] Yukuo Cen, Xu Zou, Jianwei Zhang, Hongxia Yang, Jingren Zhou, and Jie Tang. Representation Learning for Attributed Multiplex Heterogeneous Network. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 1358–1368, 2019.

[21] Zehan Chao, Denali Molitor, Deanna Needell, and Mason A. Porter. Inference of Media Bias and Content Quality Using Natural-Language Processing. *arXiv preprint arXiv:2212.00237*, 2022.

[22] Emily Chen, Ashok Deb, and Emilio Ferrara. # Election2020: The First Public Twitter Dataset on the 2020 US Presidential Election. *Journal of Computational Social Science (JCSS)*, 5(1):1–18, 2022.

[23] Jie Chen, Tengfei Ma, and Cao Xiao. FastGCN: Fast Learning with Graph Convolutional Networks via Importance Sampling. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2018.

[24] Ricky TQ Chen, Brandon Amos, and Maximilian Nickel. Learning Neural Event Functions for Ordinary Differential Equations. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2020.

[25] Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K. Duvenaud. Neural Ordinary Differential Equations. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, volume 31, 2018.

[26] Wei Chen, Xiao Zhang, Tengjiao Wang, Bishan Yang, and Yi Li. Opinion-aware Knowledge Graph for Political Ideology Detection. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 3647–3653, 2017.

[27] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, pages 2172–2180, 2016.

[28] Wei-Lin Chiang, Xuanqing Liu, Si Si, Yang Li, Samy Bengio, and Cho-Jui Hsieh. Cluster-GCN: An Efficient Algorithm for Training Deep and Large Graph Convolutional Networks. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 257–266, 2019.

[29] Uthsav Chitra and Christopher Musco. Analyzing the Impact of Filter Bubbles on Social Network Polarization. In *Proceedings of the International Conference on Web Search and Data Mining (WSDM)*, pages 115–123, 2020.

[30] Joshua Clinton, Simon Jackman, and Douglas Rivers. The Statistical Analysis of Roll Call Data. *American Political Science Review (APSR)*, 98(2):355–370, 2004.

[31] Alexis Conneau and Douwe Kiela. SentEval: An Evaluation Toolkit for Universal Sentence Representations. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC)*, 2018.

[32] Michael D. Conover, Bruno Gonçalves, Jacob Ratkiewicz, Alessandro Flammini, and Filippo Menczer. Predicting the Political Alignment of Twitter Users. In *the IEEE International Conference on Privacy, Security, Risk and Trust (PASSAT) and IEEE Third International Conference on Social Computing (SocialCom)*, pages 192–199, 2011.

[33] Michael D. Conover, Jacob Ratkiewicz, Matthew Francisco, Bruno Gonçalves, Filippo Menczer, and Alessandro Flammini. Political Polarization on Twitter. In *Proceedings of the International AAAI Conference on Weblogs and Social Media (ICWSM)*, 2011.

[34] Aron Culotta, Nirmal Ravi Kumar, and Jennifer Cutler. Predicting the Demographics of Twitter Users from Website Traffic Data. In *Proceedings of the Association for the Advancement of Artificial Intelligence Conference on Artificial Intelligence (AAAI)*, volume 15, pages 72–8, 2015.

[35] Abhimanyu Das, Sreenivas Gollapudi, and Kamesh Munagala. Modeling Opinion Dynamics in Social Networks. In *Proceedings of the ACM International Conference on Web Search and Data Mining (WSDM)*, pages 403–412, 2014.

[36] Jesse Davis and Mark Goadrich. The Relationship Between Precision-Recall and ROC Curves. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 233–240. Association for Computing Machinery (ACM), 2006.

[37] A. Philip Dawid. Causal Inference without Counterfactuals. *Journal of the American statistical Association (JASA)*, 95(450):407–424, 2000.

[38] Asim Kumar Debnath, Rosa L. Lopez de Compadre, Gargi Debnath, Alan J. Shusterman, and Corwin Hansch. Structure-Activity Relationship of Mutagenic

Aromatic and Heteroaromatic Nitro Compounds. Correlation with Molecular Orbital Energies and Hydrophobicity. *Journal of Medicinal Chemistry*, 34(2):786–797, 1991.

[39] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, pages 3844–3852, 2016.

[40] Morris H. DeGroot. Reaching a Consensus. *Journal of the American Statistical Association (JASA)*, 69(345):118–121, 1974.

[41] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, volume 1, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. (Volume 1: Long and Short Papers).

[42] Chengyu Dong, Liyuan Liu, Zichao Li, and Jingbo Shang. Towards Adaptive Residual Network Training: A Neural-ODE Perspective. In *Proceedings of the International Conference on Machine Learning (ICML)*, volume 119 of *Proceedings of Machine Learning Research (OMLR)*, pages 2616–2626, 13–18 Jul 2020.

[43] Adam Enders, Christina Farhart, Joanne Miller, Joseph Uscinski, Kyle Saunders, and Hugo Drochon. Are Republicans and Conservatives More Likely to Believe Conspiracy Theories? *Political Behavior*, 45(4):2001–2024, 2023.

[44] Paolo Ferragina and Ugo Scaiella. TagMe: On-the-fly Annotation of Short Text Fragments (by Wikipedia Entities). In *Proceedings of the ACM International Conference on Information and Knowledge Management (CIKM)*, pages 1625–1628, 2010.

[45] Emilio Ferrara, Onur Varol, Clayton Davis, Filippo Menczer, and Alessandro Flammini. The Rise of Social Bots. *Communications of the ACM*, 59(7):96–104, 2016.

[46] James Flamino, Alessandro Galeazzi, Stuart Feldman, Michael W. Macy, Brendan Cross, Zhenkun Zhou, Matteo Serafino, Alexandre Bovet, Hernán A. Makse, and Boleslaw K. Szymanski. Political Polarization of News Media and Influencers on Twitter in the 2016 and 2020 US Presidential Elections. *Nature Human Behaviour*, 7(6):904–916, 2023.

[47] Noah E. Friedkin and Eugene C. Johnsen. Social Influence and Opinions. *Journal of Mathematical Sociology*, 15(3-4):193–206, 1990.

[48] Guoji Fu, Peilin Zhao, and Yatao Bian. p-Laplacian Based Graph Neural Networks. In *Proceedings of the International Conference on Machine Learning (ICML)*, volume 162 of *Proceedings of Machine Learning Research (PMLR)*, pages 6878–6917. PMLR, 17–23 Jul 2022.

[49] Peng Fu, Zheng Lin, Fengcheng Yuan, Weiping Wang, and Dan Meng. Learning Sentiment-Specific Word Embedding via Global Sentiment Representation. In *Proceedings of the Association for the Advancement of Artificial Intelligence Conference on Artificial Intelligence (AAAI)*, volume 32, pages 4808–4815, 2018.

[50] Ryan J. Gallagher, Andrew J. Reagan, Christopher M. Danforth, and Peter Sheridan Dodds. Divergent Discourse between Protests and Counter-Protests:# BlackLivesMatter and# AllLivesMatter. *PLOS One*, 13(4):e0195644, 2018.

[51] Matthew Gentzkow and Jesse M. Shapiro. What Drives Media Slant? Evidence From U.S. Daily Newspapers. *Econometrica*, 78(1):35–71, 2010.

[52] Benyamin Ghojogh, Ali Ghodsi, Fakhri Karray, and Mark Crowley. Factor Analysis, Probabilistic Principal Component Analysis, Variational Inference, and Variational Autoencoder: Tutorial and Survey. *arXiv preprint arXiv:2101.00734*, 2021.

[53] Clark Glymour. Bayes Nets as Psychological Models. *Explanation and Cognition*, pages 169–197, 2000.

[54] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Nets. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, pages 2672–2680, 2014.

[55] Joshua Gordon, Marzieh Babaeianjelodar, and Jeanna Matthews. Studying Political Bias via Word Embeddings. In *Companion Proceedings of the Web Conference (TheWebConf/WWW)*, pages 760–764, 2020.

[56] David F. Griffiths, Desmond J. Higham, David F. Griffiths, and Desmond J. Higham. Euler's Method. *Numerical Methods for Ordinary Differential Equations: Initial Value Problems*, 5:19–31, 2010.

[57] Aditya Grover and Jure Leskovec. node2vec: Scalable Feature Learning for Networks. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 855–864, 2016.

[58] Maurício Gruppi, Panayiotis Smeros, Sibel Adalı, Carlos Castillo, and Karl Aberer. SciLander: Mapping the Scientific News Landscape. *arXiv preprint arXiv:2205.07970*, 2022.

[59] Yupeng Gu, Ting Chen, Yizhou Sun, and Bingyu Wang. Ideology Detection for Twitter Users with Heterogeneous Types of Links. *arXiv:1612.08207*, 2016.

[60] Mingyu Guan, Anand Padmanabha Iyer, and Taesoo Kim. DynaGraph: Dynamic Graph Neural Networks at Scale. In *Proceedings of the ACM SIGMOD Joint International Workshop on Graph Data Management Experiences & Systems (GRADES) and Network Data Analytics (NDA)*, page 6, 2022.

[61] Michael Gutmann and Aapo Hyvärinen. Noise-contrastive Estimation: A new Estimation Principle for Unnormalized Statistical Models. In *Proceedings of the*

International Conference on Artificial Intelligence and Statistics (AISTATS), pages 297–304, 2010.

[62] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive Representation Learning on Large Graphs. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, pages 1024–1034, 2017.

[63] Philip Hartman. *Ordinary Differential Equations*. SIAM, 2002.

[64] Hengtao He, Xianghao Yu, Jun Zhang, Shenghui Song, and Khaled B. Letaief. Message Passing Meets Graph Neural Networks: A New Paradigm for Massive MIMO Systems. *IEEE Transactions on Wireless Communications*, 23(5):4709–4723, 2024.

[65] Mikael Henaff, Joan Bruna, and Yann LeCun. Deep Convolutional Networks on Graph-Structured Data. *arXiv preprint arXiv:1506.05163*, 2015.

[66] Dichao Hu. An Introductory Survey on Attention Mechanisms in NLP Problems. In *Proceedings of the SAI Intelligent Systems Conference (IntelliSys)*, pages 432–448, 2019.

[67] Zijie Huang, Yizhou Sun, and Wei Wang. Learning Continuous System Dynamics from Irregularly-Sampled Partial Observations. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pages 16177–16187. Curran Associates, Inc., 2020.

[68] Edward L. Ince. *Ordinary Differential Equations*. Courier Corporation, 1956.

[69] Mohit Iyyer, Peter Enns, Jordan Boyd-Graber, and Philip Resnik. Political Ideology Detection Using Recursive Neural Networks. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, volume 1, pages 1113–1122, 2014. (Volume 1: Long Papers).

[70] Julie Jiang, Xiang Ren, and Emilio Ferrara. Retweet-BERT: Political Leaning Detection Using Language Features and Information Diffusion on Social Networks. In *Proceedings of the International AAAI Conference on Web and Social Media (ICWSM)*, volume 17, pages 459–469, 2023.

[71] Kristen Johnson and Dan Goldwasser. Identifying Stance by Analyzing Political Discourse on Twitter. In *Proceedings of the First Workshop on NLP and Computational Social Science (NLP+CSS)*, pages 66–75, 2016.

[72] Wei Ju, Zheng Fang, Yiyang Gu, Zequn Liu, Qingqing Long, Ziyue Qiao, Yifang Qin, Jianhao Shen, Fang Sun, Zhiping Xiao, Junwei Yang, Jingyang Yuan, Yusheng Zhao, Yifan Wang, Xiao Luo, and Ming Zhang. A Comprehensive Survey on Deep Graph Representation Learning. *Neural Networks*, 173:106207, 2024.

[73] Andreas Jungherr. Twitter use in election campaigns: A systematic literature review. *Journal of Information Technology & Politics*, 13(1):72–91, 2016.

[74] Sandeepa Kannangara. Mining Twitter for Fine-Grained Political Opinion Polarity Classification, Ideology Detection and Sarcasm Detection. In *Proceedings of the ACM International Conference on Web Search and Data Mining (WSDM)*, pages 751–752, 2018.

[75] Alex Kendall, Yarin Gal, and Roberto Cipolla. Multi-Task Learning Using Uncertainty to Weigh Losses for Scene Geometry and Semantics. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7482–7491, 2018.

[76] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[77] Thomas Kipf, Ethan Fetaya, Kuan-Chieh Wang, Max Welling, and Richard Zemel. Neural Relational Inference for Interacting Systems. In *Proceedings of the International*

Conference on Machine Learning (ICML), Proceedings of Machine Learning Research (PMLR), pages 2688–2697, 10–15 Jul 2018.

[78] Thomas N Kipf and Max Welling. Semi-Supervised Classification with Graph Convolutional Networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2016.

[79] Theresa Kuhn and Aaron Kamm. The National Boundaries of Solidarity: A Survey Experiment on Solidarity with Unemployed People in the European Union. *European Political Science Review (EPSR)*, 11(2):179–195, 2019.

[80] Mirko Lai, Marcella Tambuscio, Viviana Patti, Giancarlo Ruffo, and Paolo Rosso. Stance Polarity in Political Debates: A Diachronic Perspective of Network Homophily and Conversations on Twitter. *Data & Knowledge Engineering (DKE)*, 124:101738, 2019.

[81] Steffen L. Lauritzen. Causal inference from graphical models. *Monographs on Statistics and Applied Probability*, 87:63–108, 2001.

[82] Matthew Levendusky. *The Partisan Sort: How Liberals Became Democrats and Conservatives Became Republicans.* University of Chicago Press, Chicago, IL, USA, 2009.

[83] Omer Levy and Yoav Goldberg. Neural Word Embedding as Implicit Matrix Factorization. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, pages 2177–2185, 2014.

[84] Jing Li, Aixin Sun, Jianglei Han, and Chenliang Li. A Survey on Deep Learning for Named Entity Recognition. In *Proceedings of the International Conference on Computational Linguistics (COLING)*, pages 2145–2158. Association for Computational Linguistics, 2018.

[85] Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper Insights into Graph Convolutional Networks for Semi-Supervised Learning. In *Proceedings of the Association for the*

Advancement of Artificial Intelligence Conference on Artificial Intelligence (AAAI), pages 3538–3545, 2018.

[86] Xiao Li, Li Sun, Mengjie Ling, and Yan Peng. A Survey of Graph Neural Network based Recommendation in Social Networks. *Neurocomputing*, 549:126441, 2023.

[87] Yitan Li, Linli Xu, Fei Tian, Liang Jiang, Xiaowei Zhong, and Enhong Chen. Word Embedding Revisited: A New Representation Learning and Explicit Matrix Factorization Perspective. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 3650–3656, 2015.

[88] Robert Lieberman, Suzanne Mettler, Thomas B. Pepinsky, Kenneth M. Roberts, and Richard Valelly. Trumpism and American Democracy: History, Comparison, and the Predicament of Liberal Democracy in the United States. *Comparison, and the Predicament of Liberal Democracy in the United States*, 29 Aug 2017.

[89] Jie Liu, Shaowei Chen, Bingquan Wang, Jiaxin Zhang, Na Li, and Tong Xu. Attention as Relation: Learning Supervised Multi-head Self-Attention for Relation Extraction. In *Proceedings of the twenty-ninth international conference on international joint conferences on artificial intelligence*, pages 3787–3793, 2021.

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

[91] Ziqi Liu, Chaochao Chen, Xinxing Yang, Jun Zhou, Xiaolong Li, and Le Song. Heterogeneous Graph Neural Networks for Malicious Account Detection. In *Proceedings of the ACM International Conference on Information and Knowledge Management (CIKM)*, pages 2077–2085, 2018.

[92] Jan Lorenz. Continuous Opinion Dynamics under Bounded Confidence: A Survey. *International Journal of Modern Physics C*, 18(12):1819–1838, 2007.

[93] Ilya Loshchilov and Frank Hutter. SGDR: Stochastic Gradient Descent with Warm Restarts. *arXiv preprint arXiv:1608.03983*, 2016.

[94] Laurens van der Maaten and Geoffrey Hinton. Visualizing Data using t-SNE. *Journal of Machine Learning Research (JMLR)*, 9:2579–2605, 2008.

[95] Sergio Martini and Mariano Torcal. Trust Across Political Conflicts: Evidence from a Survey Experiment in Divided Societies. *Party Politics*, 25(2):126–139, 2019.

[96] Seth A. Marvel, Hyunsuk Hong, Anna Papush, and Steven H. Strogatz. Encouraging Moderation: Clues from a Simple Model of Ideological Conflict. *Physical Review Letters*, 109(11):118702, 2012.

[97] Diana Maynard and Adam Funk. Automatic Detection of Political Opinions in Tweets. In *Proceedings of the Extended Semantic Web Conference (ESWC)*, pages 88–99, 2011.

[98] Walaa Medhat, Ahmed Hassan, and Hoda Korashy. Sentiment Analysis Algorithms and Applications: A survey. *Ain Shams Engineering Journal*, 5(4):1093–1113, 2014.

[99] Ninareh Mehrabi, Fred Morstatter, Nripsuta Saxena, Kristina Lerman, and Aram Galstyan. A Survey on Bias and Fairness in Machine Learning. *ACM Computing Surveys (CSUR)*, 54(6):1–35, 2021.

[100] Sneha Mehta, Huzefa Rangwala, and Naren Ramakrishnan. Low Rank Factorization for Compact Multi-Head Self-Attention. *arXiv preprint arXiv:1912.00835*, 2019.

[101] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. Distributed Representations of Words and Phrases and their Compositionality. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, pages 3111–3119, 2013.

[102] Richard K. Miller and Anthony N. Michel. *Ordinary Differential Equations.* Academic Press, 2014.

[103] Fred Morstatter and Huan Liu. Discovering, Assessing, and Mitigating Data Bias in Social Media. *Online Social Networks and Media*, 1:1–13, 2017.

[104] Mohammad Rafiq Muqri. A Study of Differential Equation Solver Suites and Real-world Applications Using Python, Maple, and Matlab. In *Proceedings of the ASEE Virtual Annual Conference Content Access*, 26 Jul 2021.

[105] David Nadeau and Satoshi Sekine. A Survey of Named Entity Recognition and Classification. *Lingvisticae Investigationes*, 30(1):3–26, 2007.

[106] Dat Quoc Nguyen, Thanh Vu, and Anh-Tuan Nguyen. BERTweet: A Pre-trained Language Model for English Tweets. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing: System Demonstrations (EMNLP)*, pages 9–14, 2020.

[107] Viet-An Nguyen, Jordan Boyd-Graber, Philip Resnik, and Kristina Miler. Tea Party in the House: A Hierarchical Ideal Point Topic Model and Its Application to Republican Legislators in the 112th Congress. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL) and the International Joint Conference on Natural Language Processing (IJCNLP)*, volume 1, pages 1438–1448, Beijing, China, July 2015. (Volume 1: Long Papers).

[108] Hossein Noorazar. Recent Advances in Opinion Propagation Dynamics: A 2020 Survey. *The European Physical Journal Plus*, 135:521, 2020.

[109] Hossein Noorazar, Kevin R. Vixie, Arghavan Talebanpour, and Yunfeng Hu. From Classical to Modern Opinion Dynamics. *International Journal of Modern Physics C*, 31(07):2050101, 2020.

[110] Ann M. Oberhauser, Daniel Krier, and Abdi M. Kusow. Political Moderation and Polarization in the Heartland: Economics, Rurality, and Social Identity in the 2016 U.S. Presidential Election. *The Sociological Quarterly*, 60(2):224–244, 2019.

[111] Maya Okawa and Tomoharu Iwata. Predicting Opinion Dynamics via Sociologically-Informed Neural Networks. In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, pages 1306–1316, 2022.

[112] Cathy O'Neil. *Weapons of Math Destruction: How Big Data Increases Inequality and Threatens Democracy*. Broadway Books, New York, NY, USA, 2016.

[113] Artidoro Pagnoni, Kevin Liu, and Shangyan Li. Conditional Variational Autoencoder for Neural Machine Translation. *arXiv preprint arXiv:1812.04405*, 2018.

[114] Chang Sup Park. Does Twitter Motivate Involvement in Politics? Tweeting, Opinion Leadership, and Political Engagement. *Computers in Human Behavior*, 29(4):1641–1648, 2013.

[115] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. GloVe: Global Vectors for Word Representation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

[116] Richard J. Pierce Jr. Two Problems in Administrative Law: Political Polarity on the District of Columbia Circuit and Judicial Deterrence of Agency Rulemaking. *Duke Law Journal (DLJ)*, 37:300–308, 1988. (Available at Availableat:https://scholarship.law.duke.edu/dlj/vol37/iss2/9.).

[117] Ferran Pla and Lluís-F Hurtado. Political Tendency Identification in Twitter using Sentiment Analysis Techniques. In *Proceedings of the International Conference on Computational Linguistics: Technical Papers (COLING)*, pages 183–192, 2014.

[118] Gary Pollock, Tom Brock, and Mark Ellison. Populism, Ideology and Contradiction: Mapping Young People's Political Views. *The Sociological Review*, 63:141–166, 2015.

[119] Keith T. Poole and Howard Rosenthal. A Spatial Model for Legislative Roll Call analysis. *American Journal of Political Science*, 29(2):357–384, May 1985.

[120] Mason A. Porter and James P. Gleeson. Dynamical Systems on Networks: A Tutorial. *Frontiers in Applied Dynamical Systems: Reviews and Tutorials*, 4, 2016.

[121] Maia Powell, Arnold D. Kim, and Paul E. Smaldino. Hashtags as signals of political identity: #BlackLivesMatter and #AllLivesMatter. *PLOS One*, 18(6):e0286524, 2023.

[122] Daniel Preoţiuc-Pietro, Ye Liu, Daniel Hopkins, and Lyle Ungar. Beyond Binary Labels: Political Ideology Prediction of Twitter Users. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, volume 1, pages 729–740, 2017. (Volume 1: Long Papers).

[123] Saeed Rahmani, Asiye Baghbani, Nizar Bouguila, and Zachary Patterson. Graph Neural Networks for Intelligent Transportation Systems: A Survey. *IEEE Transactions on Intelligent Transportation Systems (T-ITS)*, 24, 2023.

[124] Hegselmann Rainer and Ulrich Krause. Opinion Dynamics and Bounded Confidence: Models, Analysis and Simulation. *Journal of Artificial Societies and Social Simulation (JASSS)*, 5(3), 2002.

[125] Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP) and the International Joint Conference on Natural Language Processing (IJCNLP)*, pages 3982–3992, 2019.

[126] Yulia Rubanova, Ricky T. Q. Chen, and David K. Duvenaud. Latent Ordinary Differential Equations for Irregularly-Sampled Time Series. *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, 32, 2019.

[127] Sebastian Ruder. An Overview of Multi-Task Learning in Deep Neural Networks. *arXiv preprint arXiv:1706.05098*, 2017.

[128] Luana Ruiz, Luiz F. O. Chamon, and Alejandro Ribeiro. Transferability Properties of Graph Neural Networks. *IEEE Transactions on Signal Processing*, 71:3474–3489, 2023.

[129] B. J. Rye and Angela Underhill. Pro-choice and Pro-life Are Not Enough: An Investigation of Abortion Attitudes as a Function of Abortion Prototypes. *Sexuality & Culture*, 24:1829–1851, 2020.

[130] Rohit Sahasrabuddhe, Leonie Neuhäuser, and Renaud Lambiotte. Modelling Non-linear Consensus Dynamics on Hypergraphs. *Journal of Physics: Complexity*, 2(2):025006, 2021.

[131] Mohsen Sayyadiharikandeh, Onur Varol, Kai-Cheng Yang, Alessandro Flammini, and Filippo Menczer. Detection of Novel Social Bots by Ensembles of Specialized Classifiers. In *Proceedings of the ACM International Conference on Information and Knowledge Management (CIKM)*, pages 2725–2732, 2020.

[132] Michael Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. Modeling Relational Data with Graph Convolutional Networks. In *Proceedings of the European Semantic Web Conference (ESWC)*, pages 593–607, 2018.

[133] Ana Lucía Schmidt, Fabiana Zollo, Michela Del Vicario, Alessandro Bessi, Antonio Scala, Guido Caldarelli, H. Eugene Stanley, and Walter Quattrociocchi. Anatomy of News Consumption on Facebook. In *Proceedings of the National Academy of Sciences (PNAS)*, volume 114, pages 3035–3039. National Acad Sciences, 2017.

[134] Michael Schober, Simo Särkkä, and Philipp Hennig. A Probabilistic Model for the Numerical Solution of Initial Value Problems. *Statistics and Computing*, 29(1):99–122, 2019.

[135] Michael F. Schober, Josh Pasek, Lauren Guggenheim, Cliff Lampe, and Frederick G. Conrad. Social Media Analyses for Social Measurement. *Public Opinion Quarterly*, 80(1):180–211, 2016.

[136] Shahnawaz and Parmanand Astya. Sentiment Analysis: Approaches and Open Issues. In *Proceedings of International Conference on Computing, Communication and Automation (ICCCA)*, pages 154–158, 2017.

[137] Jingbo Shang, Jialu Liu, Meng Jiang, Xiang Ren, Clare R. Voss, and Jiawei Han. Automated Phrase Mining from Massive Text Corpora. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 30(10):1825–1837, 2018.

[138] Joakim Skarding, Bogdan Gabrys, and Katarzyna Musial. Foundations and Modeling of Dynamic Networks Using Dynamic Graph Neural Networks: A Survey. *IEEE Access*, 9:79143–79168, 2021.

[139] Michael E. Sobel. Causal Inference in the Social Sciences. *Journal of the American Statistical Association (JASA)*, 95(450):647–651, 2000.

[140] Richard Socher, Danqi Chen, Christopher D. Manning, and Andrew Ng. Reasoning With Neural Tensor Networks for Knowledge Base Completion. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, pages 926–934, 2013.

[141] Yeongkil Song, Seokwon Jeong, and Harksoo Kim. Semi-automatic Construction of a Named Entity Dictionary for Entity-based Sentiment Analysis in Social Media. *Multimedia Tools & Applications*, 76(9):11319–11329, 2017.

[142] Chenkai Sun, Jinning Li, Yi Fung, Hou Pong Chan, Tarek Abdelzaher, ChengXiang Zhai, and Heng Ji. Decoding the Silent Majority: Inducing Belief Augmented Social Graph with Large Language Model for Response Forecasting. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 43–57, 2023.

[143] Yizhou Sun and Jiawei Han. Mining Heterogeneous Information Networks: Principles and Methodologies. *Synthesis Lectures on Data Mining and Knowledge Discovery*, 3(2):1–159, 2012.

[144] Duyu Tang, Furu Wei, Nan Yang, Ming Zhou, Ting Liu, and Bing Qin. Learning Sentiment-Specific Word Embedding for Twitter Sentiment Classification. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, volume 1, pages 1555–1565, 2014. (Volume 1: Long Papers).

[145] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. LINE: Large-scale Information Network Embedding. In *Proceedings of the 24th International Conference on World Wide Web (TheWebConf/WWW)*, pages 1067–1077, 2015.

[146] Devendra Kr. Tayal, Sumit Yadav, Komal Gupta, Bhawna Rajput, and Kiran Kumari. Polarity Detection of Sarcastic Political Tweets. In *Proceedings of the International Conference on Computing for Sustainable Global Development (INDIACom)*, pages 625–628, 2014.

[147] Joseph H. Tien, Marisa C. Eisenberg, Sarah T. Cherng, and Mason A. Porter. Online Reactions to the 2017 'Unite the Right' Rally in Charlottesville: Measuring Polarization in Twitter Networks Using Media Followership. *Applied Network Science*, 5(1):1–27, 2020.

[148] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is All You Need. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, pages 5998–6008, 2017.

[149] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph Attention Networks. 2018. (poster).

[150] Maurice Vergeer. Twitter and Political Campaigning. *Sociology Compass*, 9(9):745–760, 2015.

[151] Prashanth Vijayaraghavan, Soroush Vosoughi, and Deb Roy. Twitter Demographic Classification Using Deep Multi-modal Multi-task Learning. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, volume 2, pages 478–483, 2017. (Volume 2: Short Papers).

[152] Hongwei Wang, Fuzheng Zhang, Min Hou, Xing Xie, Minyi Guo, and Qi Liu. SHINE: Signed Heterogeneous Information Network Embedding for Sentiment Link Prediction.

In *Proceedings of the ACM International Conference on Web Search and Data Mining (WSDM)*, pages 592–600, 2018.

[153] Lingfei Wang, Carmela Bernardo, Yiguang Hong, Francesco Vasca, Guodong Shi, and Claudio Altafini. Achieving Consensus in spite of Stubbornness: Time-varying Concatenated Friedkin-Johnsen Models. In *Proceedings of the IEEE Conference on Decision and Control (CDC)*, pages 4964–4969, 2021.

[154] Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S. Yu. Heterogeneous Graph Attention Network. In *The World Wide Web Conference (TheWebConf/WWW)*, pages 2022–2032, 2019.

[155] Michael L. Waskom. Seaborn: Statistical Data Visualization. *The Journal of Open Source Software*, 6(60):3021, 2021.

[156] Steven W. Webster and Alan I. Abramowitz. The Ideological Foundations of Affective Polarization in the US Electorate. *American Politics Research*, 45(4):621–647, 2017.

[157] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S. Yu Philip. A Comprehensive Survey on Graph Neural Networks. *IEEE Transactions on Neural Networks and Learning Systems (TNNLS)*, 32(1):4–24, 2020.

[158] Zhiping Xiao, Weiping Song, Haoyan Xu, Zhicheng Ren, and Yizhou Sun. TIMME: Twitter Ideology-Detection via Multi-Task Multi-Relational Embedding. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, page 2258–2268, 2020.

[159] Zhiping Xiao, Jeffrey Zhu, Yining Wang, Pei Zhou, Wen Hong Lam, Mason A. Porter, and Yizhou Sun. Detecting political biases of named entities and hashtags on Twitter. *European Physical Journal (EPJ) — Data Science*, 12(1):20, 2023.

[160] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How Powerful are Graph Neural Networks? In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019.

[161] Bishan Yang, Scott Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. Embedding Entities and Relations for Learning and Inference in Knowledge Bases. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2015.

[162] Liang-Chih Yu, Jin Wang, K. Robert Lai, and Xuejie Zhang. Refining Word Embeddings for Sentiment Analysis. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 534–539, 2017.

[163] Hao Yuan, Jiliang Tang, Xia Hu, and Shuiwang Ji. XGNN: Towards Model-Level Explanations of Graph Neural Networks. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 430–438, 2020.

[164] Seongjun Yun, Minbyul Jeong, Raehyun Kim, Jaewoo Kang, and Hyunwoo J. Kim. Graph Transformer Networks. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, pages 11960–11970, 2019.

[165] Junhai Zhai, Sufang Zhang, Junfen Chen, and Qiang He. Autoencoder and Its Various Variants. In *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 415–419, 2018.

[166] Chuxu Zhang, Dongjin Song, Chao Huang, Ananthram Swami, and Nitesh V. Chawla. Heterogeneous Graph Neural Network. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 793–803, 2019.

[167] Hong Zhang and Wenjun Zhao. A Memory-Efficient Neural Ordinary Differential Equation Framework Based on High-Level Adjoint Differentiation. *IEEE Transactions on Artificial Intelligence*, 5(3):1110–1120, 2022.

[168] Yifei Zhang, Hao Zhu, Ziqiao Meng, Piotr Koniusz, and Irwin King. Graph-adaptive

Rectified Linear Unit for Graph Neural Networks. In *Proceedings of the ACM Web Conference (TheWebConf/WWW)*, pages 1331–1339, 2022.

[169] Yufeng Zhang, Jialu Pan, Li Ken Li, Wanwei Liu, Zhenbang Chen, Xinwang Liu, and Ji Wang. On the Properties of Kullback-Leibler Divergence Between Multivariate Gaussian Distributions. *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, 36, 2024.

[170] Ziwei Zhang, Peng Cui, and Wenwu Zhu. Deep Learning on Graphs: A Survey. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 34(1):249–270, 2020.

[171] Jieyu Zhao, Tianlu Wang, Mark Yatskar, Vicente Ordonez, and Kai-Wei Chang. Gender Bias in Coreference Resolution: Evaluation and Debiasing Methods. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, volume 2, pages 15–20, 2018. (Volume 2: Short Papers).

[172] Jieyu Zhao, Yichao Zhou, Zeyu Li, Wei Wang, and Kai-Wei Chang. Learning Gender-Neutral Word Embeddings. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4847–4853, 2018.

[173] Ling Zhao, Yujiao Song, Chao Zhang, Yu Liu, Pu Wang, Tao Lin, Min Deng, and Haifeng Li. T-GCN: A Temporal Graph Convolutional Network for Traffic Prediction. *IEEE Transactions on Intelligent Transportation Systems (T-ITS)*, 21(9):3848–3858, 2020.

[174] Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph Neural Networks: A Review of Methods and Applications. *AI Open*, 1:57–81, 2020.

[175] Di Zhu and Yu Liu. Modelling Spatial Patterns Using Graph Convolutional Networks. In *Proceedings of the International Conference on Geographic Information Science (GIScience)*, 2018. (Short Paper).

[176] Hao Zhu and Piotr Koniusz. Generalized Laplacian Eigenmaps. *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, 35:30783–30797, 2022.

[177] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired Image-To-Image Translation Using Cycle-Consistent Adversarial Networks. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 2223–2232, 2017.

[178] Juntang Zhuang, Nicha Dvornek, Xiaoxiao Li, Sekhar Tatikonda, Xenophon Papademetris, and James Duncan. Adaptive Checkpoint Adjoint Method for Gradient Estimation in Neural ODE. In *Proceedings of the International Conference on Machine Learning (ICML)*, volume 119 of *Proceedings of Machine Learning Research (PMLR)*, pages 11639–11649, 13–18 Jul 2020.

[179] Juntang Zhuang, Nicha C. Dvornek, Sekhar Tatikonda, and James S. Duncan. MALI: A Memory Efficient and Reverse Accurate Integrator for Neural ODEs. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2020.