Joint Learning for Tweet Sentiment Analysis using Textual and Link Data

Research Type Project

Group No. 9

20474922

ANG, Clyde Wesley Si

FYP/Research Supervisor: Raymond Wong
FYP/Research Topic: Session-based recommendation systems.

I hereby declare that this project is done solely within the course but not other scopes.

20466559

UY, Mark Christopher

FYP/Research Supervisor: Sung Kim Hyun
FYP/Research Topic: Source code generation.

I hereby declare that this project is done solely within the course but not other scopes.

20477053

WANG, Aaron Si-yuan

FYP/Research Supervisor: N/A
FYP/Research Topic: N/A

I hereby declare that this project is done solely within the course but not other scopes.

20466705

KUMYOL, Serkan

FYP/Research supervisor: Dekai Wu
FYP/Research Topic: Semantic role labelling and its applications to downstream NLP tasks.

I hereby declare that this project is done solely within the course but not other scopes.

Twitter is a social network with heterogeneous links between users, with each user having different opinions over various issues that create a complex distribution of political stances. Such a problem is usually studied either by using network analysis or textual sentiment analysis. However, we claim that the two domains can be incorporated to detect the sentiment of a user towards a political issue. Specifically, the links of a node provide a certain degree of information about the political alignment of a given user, while the tweets of the user provide context about the user's opinion. These two pieces of information are valuable to help classify a user's stance towards a political issue as either neutral or partisan.

Keywords: Heterogeneous graphs; opinion mining; sentiment analysis.

1. Introduction

As social networking sites like Twitter become a main channel to express political opinions, artificial intelligence solves the problem of interpreting this information and developing low-bias classification systems. Opinion mining, also known as sentiment analysis, plays a key role in measuring the public's stance towards key issues. One method used in gathering information from social networks is text-based sentiment analysis, which can take text as its input and output labels from a predefined set. Another method is identifying clusters of users in the social network topology using the network links. In this paper, we focus on integrating both approaches to determine the sentiment of tweets that are gathered from political accounts.

The main problem with topology-based approaches is the difficulty in determining whether links between nodes arise from a shared political view or from contrary stances. Additionally, a given user might have varying opinions on a topic, so discarding textual information will significantly hamper results. Thus, tweet sentimental analysis cannot be determined solely from the network topology. On the other hand, considering only textual information limits information that can be useful in determining political stances and detecting bot accounts. Although this paper does not aim to identify such phenomena, the belief is that combining both pieces of information is a necessary approach to obtain a better representation of users and their tweet sentiments in terms of political ideologies.

In this paper, we aim to explore existing methods for political sentiment classification, propose opinion mining methods that use both pieces of information, and conduct experiments to validate the proposed methods. Specifically, we attempt to hypothesize different models to learn tweet sentiment in a combined fashion, and aim to present the model using the latest tools for both social network analysis and sentiment analysis. Although both joint learning models and pipeline models will be discussed, this paper will only present the results for the pipeline model due to limitations over the required time to implement the models.

2. Background Knowledge and Related Work

2.1. *Related Literature*

Social networks are networks of accounts where users are represented as nodes and connections between users are represented as edges or links. According to [1], a social network has the following properties: "(1) construct a public or semi-public profile within a bounded system, (2) articulate a list of other users with whom they share a connection, and (3) view and traverse their list of connections and those made by others within the system." Additionally, the nature of the connections between users in Twitter is decentralized and highly interactive due to heterogeneous interactions, such as follows, retweets, and mentions, between accounts. Social network analysis aims to analyze complex interactions between social network users using mathematical models such as graph theory and linear algebra [2].

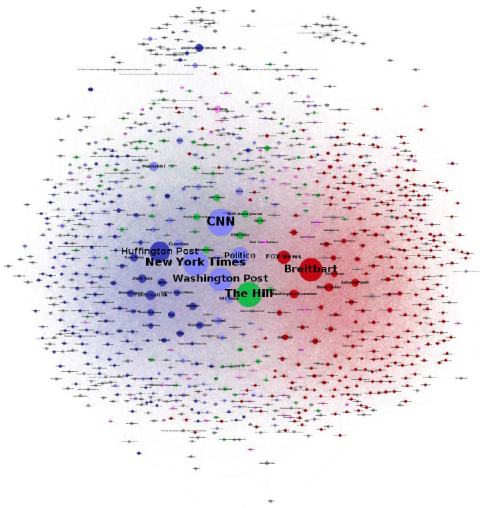


Fig. 1: Network map based on Twitter media sharing from May 1, 2015 to November 7, 2016 with nodes sized by number of Twitter shares [3]

Twitter nodes can pose complex sub-network topology that imply clusters over different interests. However, it is difficult and often inaccurate to cluster Twitter nodes using political alignments. This is because users that are stationed on the same cluster of political alignments might have contrasting opinions on different topics. Thus, we need to take a less polarized approach to understand the latent political features of the network. As seen in Figure 1, Twitter users from the United States can be divided into two groups, Republicans and Democrats. However, it is still deceptive to classify nodes into Republicans and Democrats as it can introduce a wrong bias. In a recent fact-checking system [4], it has been shown that the opinions of Republicans and Democrats can greatly vary across different topics. Hence, rather than classifying users into these two groups, classifying their tweets as either neutral or partisan is a more appropriate approach to reveal the latent political features of the network.

There have been various attempts to predict political alignments of tweets using sentiment analysis methods [5]. Early methods mostly used Support Vector Machines [6] in opinion mining, while lexicon-based approaches also try to infer information from the texts using ontological networks [7]. However, recent methods in sentiment analysis are dominated by neural networks [8] [9], with attention-based mechanisms leading to better performance on natural language processing tasks [10]. Given the recent breakthroughs in natural language processing, sentiment analysis results are significantly improved upon by using bidirectional **LSTM**, autoencoders and attention mechanisms. In particular, models such as **BERT** [11] have resulted in state-of-the-art results for multiple natural language processing tasks.

Aside from sentiment analysis approaches, various social network analysis approaches use link information to infer the political alignment of the network nodes. Some studies aim to estimate the political preference of the audience from different media outlets and organizations by using Twitter data [12]. The system is based on the statistics of the follow relationships between Twitter users that are related to politician accounts from both Republican and Democratic parties. A more comprehensive approach [13] makes use of the information from retweets as well. Retweets contain information involving the positive or negative sentiment of the user, thereby accounting for the heterogeneity of the social network. A further work [14] was brought up by questions of quantification and scalability. The main concern of the study was quantifying the political ideas in the social network, and whether experimental methods can be implemented over the entire Twitter network instead of a subsample. In order to solve such problems, they reduced the data by grouping tweets into their relevancy to a topic, and then conducted sentiment analysis over the relevant tweets to obtain the sentiment, i.e. positive, negative, or neutral, of the user towards the topic. Another approach [15] uses hierarchical classification (elite/normal accounts) using only follow links and famous accounts on Twitter as its seed. Finally, [16] estimates the ideal point (ideology) of a node in the graph by treating it as a latent variable in Bayesian Network, using only follow links between politicians and normal users.

2.2. *Definitions*

In this paper, we aim to combine both the sentiment analysis approach and the social network analysis approach to find the sentiment of a given tweet. Specifically, our problem statement is defined below.

> **Problem definition**: Given a network N and a user u, together with the user's twitter feed T, we want to learn the partisanship of each tweet t in T by using the distance of the user from other accounts $d(u)$, where $d(u)$ is based on follows u_f and retweets u_{rt}, together with the contents of each tweet t to produce a binary classification of **neutral** or **partisan** for each t.

In network analysis, we adopt the **ML-IPM** (Multi-Link Ideal Point Estimation Model) [17], which is a probabilistic model that detects the numerical ideology of Twitter users using heterogeneous link information. Meanwhile, we use the **BERT** model [11] to help with the natural language processing task. Our goal is to incorporate the political distances obtained from **ML-IPM** and the outputs of **BERT** for the classification model. Both models will be discussed in further details.

2.3. *ML-IPM*

ML-IPM is a unified probabilistic model which combines information from R heterogeneous types of links in a network to determine a user's ideology. This is better

than standard models that only account for one type of link in the network. This is done by representing each user u_i by a vector in a K-dimensional ideology space $\boldsymbol{p}_i \in \mathbb{R}^K$. Each user is also associated with R vectors $\boldsymbol{q}_i^{(r)} \in \mathbb{R}^K$ for $r \in \{1, \cdots, R\}$, which can be interpreted as the image of user u_i when viewed by others, as well as R bias terms $b_i^{(r)}$. There are R different vectors since the likelihood of each link occurring is not the same and depends on the nature of the link. With this, the probability that a user u_i follows target user v_j with link type r can be modelled by Equation 1.

$$p(u_i \xrightarrow{\text{link type } r} v_j) = \sigma(\boldsymbol{p}_i \cdot \boldsymbol{q}_j^{(r)} + b_j^{(r)}) = \sigma_{r,ij} \tag{1}$$

where $\sigma(x) = \frac{1}{1+e^{-x}}$ is the sigmoid function.

Using this equation, the **ML-IPM** model tries to maximize the average log-likelihood of the existing link probabilities using gradient descent for each of the different link types. The model then combines the log-likelihood values as a weighted sum that can be used to indicate the relevance of each link type in determining the user's ideal point. The structure of this loss function will be discussed in further detail in Section 5.1.

The benefit of **ML-IPM** is that it is able to capture the heterogeneous links between Twitter users in the user embeddings, as well as learn the relative weights between these different link types automatically. As such, it is well suited to our task since the user's embedding in the ideology space is likely to affect the tweeting style and content.

2.4. *BERT*

Transformers [21] and its variations made a breakthrough in natural language processing since these models are capable of learning complex hidden relationships within the data. However, such models can only converge after an expensive and data-intensive training process. Thus, using such models for machine learning tasks is a plausible option only if they are pre-trained. Some models that fit this criteria and are suitable for our task are **BERT**, **OpenAI GPT-2**, and **RoBERTa**. Although we are not going to fully analyze the distinctions between the models, we proceed with using the vanilla **BERT** as our learning framework, together with its pre-trained matrix weights.

The main advantage of **BERT** is its ability in learning a deeply bidirectional encoding for the words. This is possible since **BERT** introduces random masking, where 15% of the tokens are to be predicted during training time. In order to prevent the elimination of the tokens, they rolled a dice to choose the processing on the selected token, with 80% probability to replace the chosen word with [**MASK**], 10% probability to replace it with a random word and 10% probability to do nothing. The final layer of the model predicts the missing words, allowing the model to get an accurate representations for these words. Another distinction of **BERT** is the next

sentence prediction task, where it uses a binary classifier to label whether sentence pairs are sequential. The model is trained for the task such that for each sentence, there is a 50% chance that the following sentence is indeed the original subsequent sentence. This allows **BERT** to learn the dependencies between sentences, which will prove useful in tweets that contain multiple ideas and are essentially a summary of multiple sentences.

As shown in Figure 2, **BERT** is a model that brings deeply bidirectional encoding and discourse-level representation to the word embeddings. **BERT** is the recent state-of-the-art model for word embeddings in many downstream natural language processing tasks. One other advantage of using **BERT** is that its structure allows the user to easily fine-tune the model for the task by adding a single layer. This is seen in the fine-tuning section of the figure.

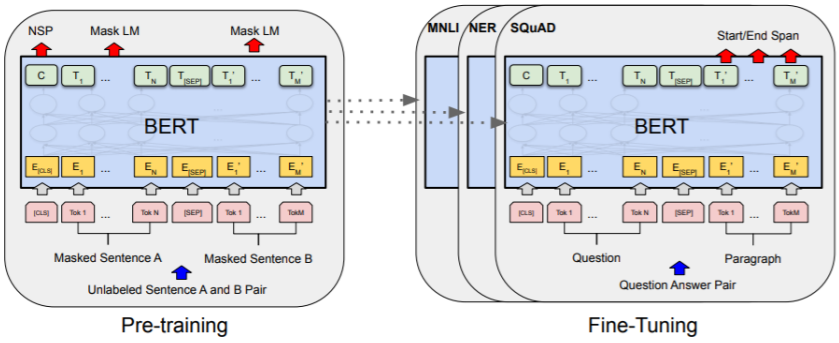


Fig. 2: Pre-training and fine-tuning phases of the **BERT** model [11]

3. Proposed Models

In this section, we introduce our proposed models for the given task. We have two main ideas for combining textual data and network data, namely the joint learning model and the pipeline model. We will discuss both models in detail and outline the advantages and disadvantages when compared to standard models.

3.1. *Joint Learning Model*

The joint learning model is a simplistic deep learning model that uses jointly learned embeddings to achieve a multimodal representation of the textual and link data. Joint embeddings have been used for the joint learning of images and texts in literature [20], but it has not yet been used for the joint learning of networks and texts. As such, the proposed model still has room for theoretical improvements.

The joint learning model is based on learning the correlation between two datasets. Given two heterogeneous datasets X and Y, we aim to learn a function f

where the function can learn $f : X \rightarrow Y$. The key challenge here is that X and Y are from different domains, namely the network domain and the language domain. Therefore, we need to create a model that simultaneously learns both domains, as shown in Figure 3. This is done by first training the model to learn the relationship between the network and the tweets, then retraining the model to transfer the knowledge from joint learning to the binary classification task.

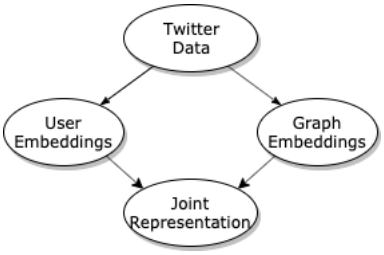


Fig. 3: Projection of Twitter data onto two different domains

In order to model the multimodal representation, we can use the multimodal embedding method [18], which enables us to jointly correlate data from different domains. Such a method results in a joint representational space where similar modalities have a closer space. We will borrow the formulation proposed in [19] as shown in Equation 2.

$$X \xrightarrow{\phi} V_x \rightarrow J(V_x, V_y) \leftarrow V_y \xleftarrow{\psi} Y \tag{2}$$

where $\phi : X \rightarrow R^d$ is an embedding function that maps X into a d-dimensional vector space V_X, $\psi : Y \rightarrow R^d$ is an embedding function that maps Y into a similar vector space V_Y, and $J(,)$ is the cosine similarity function used to score the matching degrees of V_X and V_Y in order to learn the embedding functions.

The goal of the model during the first phase of training is to maximize the similarity between the tweet vector and the user vector for each tweet. The first phase of training to jointly learn the embeddings is seen in Figure 4. In the left portion of the figure, we add a language model such as **LSTM** to transform the tweet into a fixed vector v_t in R^d, and in the right portion of the figure, we add a network model such as Graph Convolutional Networks (**GCN**) to transform the graph into a user vector v_u also in R^d. The similarity measure is then calculated using the formula in Equation 3.

$$\cos(v_t, v_u) = \frac{{v_t}^\top v_u}{\|v_t\| \|v_u\|} \tag{3}$$

After the first phase of training, we have learned the functions $\phi : X \rightarrow R^d$ and $\psi : Y \rightarrow R^d$. Then during the second phase of training, the goal of the model is to

produce a classifier that can accurately predict the labels. So in the second phase of training, we retrain our classifier by adding a Multi-layer Perceptron (**MLP**) and sigmoid layer that takes in the tweet vector v_t and the user vector v_u as inputs and outputs the probability for each label. The details of the second phase of training for producing the labels are presented in Figure 5.

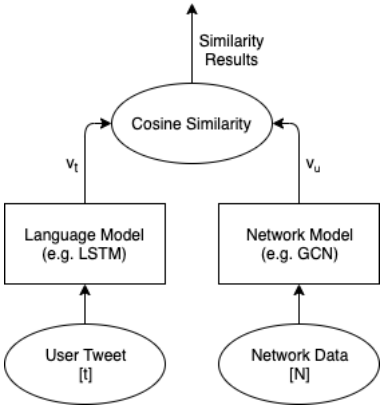


Fig. 4: First Phase - Joint Learning of Embeddings

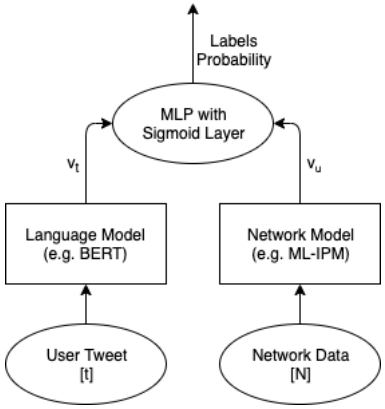


Fig. 5: Second Phase - Transfer Learning of Embeddings

Because of the first phase of training, this model can successfully learn embeddings for the tweets and the network in a joint manner. This means that during the second phase of training, we are able to transfer this jointly learned embedding into the classification task, which should result in a better representation and more accurate classification. However, one possible drawback to this model is that the model might take longer to train or produce inaccurate results if the two domains have low correlation.

3.2. *Pipeline Model*

The pipeline model borrows from an existing idea based on autoencoders suggesting that a mere concatenation of the user's graph embeddings and tweet embeddings can improve opinion classification. This model combines the network model and the language model in a more shallow manner. Essentially, the model treats the network model and the language model as separate pipelines that will each output an unnormalized prediction score. The model then combines these scores using some operation, such as an add operation, and passes the result to the sigmoid layer to get the final output probability. One advantage of this approach is that we are able to work on different pipelines at the same time, which enables for faster development. Additionally, we can also replace specific components or pipelines of the model easily as the two pipelines are loosely tied together.

For the network model pipeline, the result of the network model will associate each user with a fixed length vector v_u, which can either be made trainable or not. This vector is then passed into a trainable **MLP** to obtain an unnormalized prediction score, s_u. Meanwhile, the language model pipeline should also output some unnormalized prediction score, s_t. The language model should output a vector v_t that encapsulates the whole tweet. The way we obtain the tweet vector in this paper will be discussed in detail in Section 5. Afterwards, we pass the tweet vector into another trainable **MLP** in order to get s_t. Finally, we combine s_u and s_t with some operation, such as a simple add operation, to get s, which we will pass that to the sigmoid function to obtain the final result. The full details of this model is demonstrated in Figure 6.

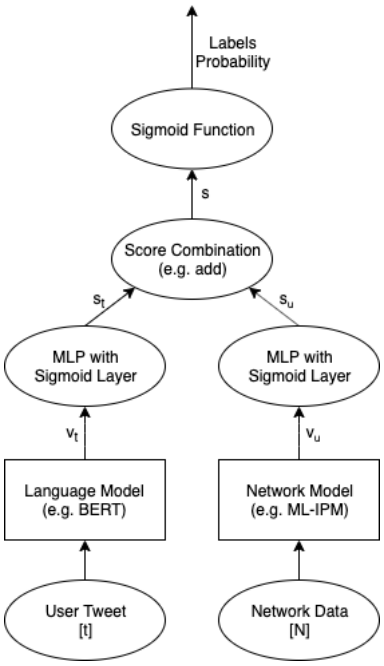


Fig. 6: Pipeline Model

Ultimately, this model will take as input the network graph as spatial information and the tweet as textual information in order to provide a more holistic evaluation of whether the tweet's sentiment is neutral or partisan. Although this model has the advantage of easily incorporating existing state-of-the-art models, its drawback is that it does not allow for a truly joint learning mechanism, as the two domains are just learned as features.

3.3. *Model Evaluation*

We acknowledge that there are possible advantages and disadvantages of using either the joint learning model or the pipeline model when compared to standard models:

- **Advantages**
  - The model can incorporate two state-of-the-art models, thereby inheriting the advantages of both models.
  - The model is more robust to noise since there are two sources of data.
  - Using a network model such as **ML-IPM** can take advantage of the network's heterogeneity.
- **Disadvantages**
  - The model incorporates two models, thereby inheriting the disadvantages of both models.
  - The model can only predict tweet sentiment for users in the graph. For users not in the graph, the results will be subpar to textual models.
  - The model needs to periodically update the user embeddings from the graph, thereby needing to call network models such as **ML-IPM** repeatedly.

4. Dataset

In this section, we describe the original source for the dataset used in the paper, together with its description and further processing done to obtain data needed for the user embeddings.

4.1. *Original Dataset*

The objective of the paper was to produce a classifier that can accurately identify the political inclination of tweets. As such, we needed a dataset containing labelled political tweets. We decided to use a freely available dataset on Kaggle called "**Political Social Media Posts**"[a] to prevent researcher bias into the dataset and enable easier reproduction of results.

The original dataset contained 2500 tweets from various political figures such as senators, with each tweet having a manually labelled column called *bias* to describe whether the tweet is **neutral** or **partisan**. This is what the model tries to predict and is considered the ground truth data for the paper. Additionally, there were other columns such as *audience*, to describe the intended reader of the tweet as either the politician's **constituents** or the whole **nation**, and *message*, to categorize the tweet into its intended purpose, including **attack**, **support**, or **personal**. However, these were not related to the intended purpose of the paper and were discarded to avoid influencing the model's prediction.

[a]https://www.kaggle.com/crowdflower/political-social-media-posts

4.2. *Data Pre-processing*

Since we already have the textual data (i.e. tweets) needed for the paper, we focus on processing the data further to obtain our network data. We first gather the Twitter usernames of all politicians included in the dataset. To ensure the validity of the dataset, we remove all politicians whose usernames are not verified, since this can indicate that the politician changed username or removed their Twitter account.

We then gather the first type of link, follow links, between users. As the user embeddings in **ML-IPM** require more nodes in the network, we gather all of the followers and followees of the verified politician accounts as potential nodes for the network. To ensure that majority of the users in the dataset are politics-related, we imitate [17] and keep only users that have at least 20 links to politicians, together with a random 1% of the remaining users as peripheral users. The follow links used in **ML-IPM** always include politician accounts so that noise found in links between non-politician users will not be introduced to the model.

Afterwards, we gather a second type of link, retweet links, between users. In order to do this, we get the 20 most recent tweets for each politician, and check whether any user from our network retweeted it. If there is a retweet, we will add this to our network as a retweet link. Although the amount of tweets per politician is small, due to Twitter's limit in allowing API calls, adding a second type of link to the network makes it heterogeneous and enhances the quality of the user embeddings. A summary of the network can be found in Table 1.

Data Type	**Count**
Politicians	379
Normal Users	19443
Follow Links	588892
Retweet Links	4478

Table 1: Network Summary

5. System Architecture

The technical details for the implementation of **ML-IPM** and **BERT** for the pipeline model in Section 3.2 are detailed in this section.

5.1. *ML-IPM Implementation*

5.1.1. *Objective Function*

In order to express the optimization process more succinctly, we will state the objective function in a different but equivalent form to that presented in [17]. The

objective function of the **ML-IPM** model is to maximize the weighted sum of the average log-likelihood of the link probabilities, $\sigma_{r,ij}$, as described in Equation 1. To calculate this, we denote $S_+^{(r)}$ to be the set of existing links of type r, with the number of such links being $N_r = |S_+^{(r)}|$. In addition, we sample N_r non-existing links to create set $S_-^{(r)}$. Denoting $e_{r,ij}^+$ and $e_{r,ij}^-$ to be the number of links from user u_i to user v_j in $S_+^{(r)}$ and $S_-^{(r)}$ respectively, the negative average log-likelihood of the link probabilities of type r can be expressed as:

$$L_r = -\frac{\sum\limits_{(i,j)\in S_+^{(r)}} e_{r,ij}^+ \log \sigma_{r,ij} + \sum\limits_{(i,j)\in S_-^{(r)}} e_{r,ij}^- \log(1-\sigma_{r,ij})}{N_r} \tag{4}$$

Thus, the loss function of **ML-IPM** can be described as in Equation 5.

$$L(\boldsymbol{G}|\boldsymbol{P},\boldsymbol{Q},\boldsymbol{b}) = \sum_{r=1}^{R} w_r L_r + \frac{\mu}{2}\left(\|\boldsymbol{P}\|_F^2 + \sum_{r=1}^{R}\|\boldsymbol{Q}^{(r)}\|_F^2 + \sum_{r=1}^{R}\|\boldsymbol{b}^{(r)}\|_2^2\right) \tag{5}$$

where $\boldsymbol{P} = \{\boldsymbol{p}_i\}$, $\boldsymbol{Q}^{(r)} = \{\boldsymbol{q}_j^{(r)}\}$, $\boldsymbol{b}^{(r)} = \{b_j^{(r)}\}$, link type weights $\boldsymbol{w} = \{w_r\}$ are to be learned, $\|\cdot\|_F$ is the Frobenius norm, $\|\cdot\|_2$ is the L2 norm, and $\mu > 0$ is a regularization hyperparameter.

5.1.2. *Optimization*

The learning process of **ML-IPM** has two alternating steps to optimize the model parameters:

- Update $\boldsymbol{P}, \boldsymbol{Q}, \boldsymbol{B}$ given $\boldsymbol{w}$ using gradient descent on Equation 5.
- Update $\boldsymbol{w}$ given $\boldsymbol{P}, \boldsymbol{Q}, \boldsymbol{B}$ using Equation 6.

$$w_r = \frac{\left(\prod\limits_{r=1}^{R} L_r\right)^{\frac{1}{R}}}{L_r} \tag{6}$$

This training process has time complexity linear to the number of edges in the network, making the algorithm efficient and scalable to large networks. After doing this, we will be able to get user embeddings based on their position $\boldsymbol{p}_i$ in the ideology space $\mathbb{R}^K$. We implemented the **ML-IPM** model as described above using the **NetworkX** and **Tensorflow** Python libraries to obtain the network pipeline of the model.

5.2. *BERT Implementation*

5.2.1. *Data Preparation*

Due to the noisy nature of tweet data. The data was cleaned to remove usernames, hashtags, hyperlinks, punctuation marks, etc. Furthermore, since we plan to use

MLP and **CNN** layers, the input sequences have to be of the same length. Hence, a fixed sequence length of 32 was used since tweets are short by nature, with a maximum of 140 characters[b]. It is important to note that when we mean a sequence length of 32, this is after applying the pre-trained **BERT** tokenizer to the tweets. The tokenizer was created using word-piece tokens and essentially splits unknown or unseen words into known word-pieces or word-parts. As a result, out of vocabulary words is not really an issue. If the number of tokens is less than 32, the sequence is padded with 0 tokens. **BERT** is able to distinguish real tokens and these padded tokens by providing a mask input, which is 1 if the token is real and 0 if it is a padded input. For our purposes, we use the **bert-based-uncased** tokenizer as the tweets no longer have any capital letters after cleaning them.

5.2.2. *Model Architecture*

The original **BERT** paper [11] achieved state-of-the-art results in different natural language processing tasks by simply using a dense layer on top of the main **BERT** as its classification layer. However, this paper explores using three different classification layers or architectures to get better results. The resulting models are called **BERT-MLP**, **BERT-LSTM**, and **BERT-CNN**. As the names suggest, **BERT-MLP** uses a a single **MLP** layer as its classification layer, which is the same architecture used in the fine-tuning of the original paper, while **BERT-LSTM** uses an additional **BiLSTM** layer, and **BERT-CNN** uses an additional **CNN** layer. The architectures of the models can be found in Figure 7.

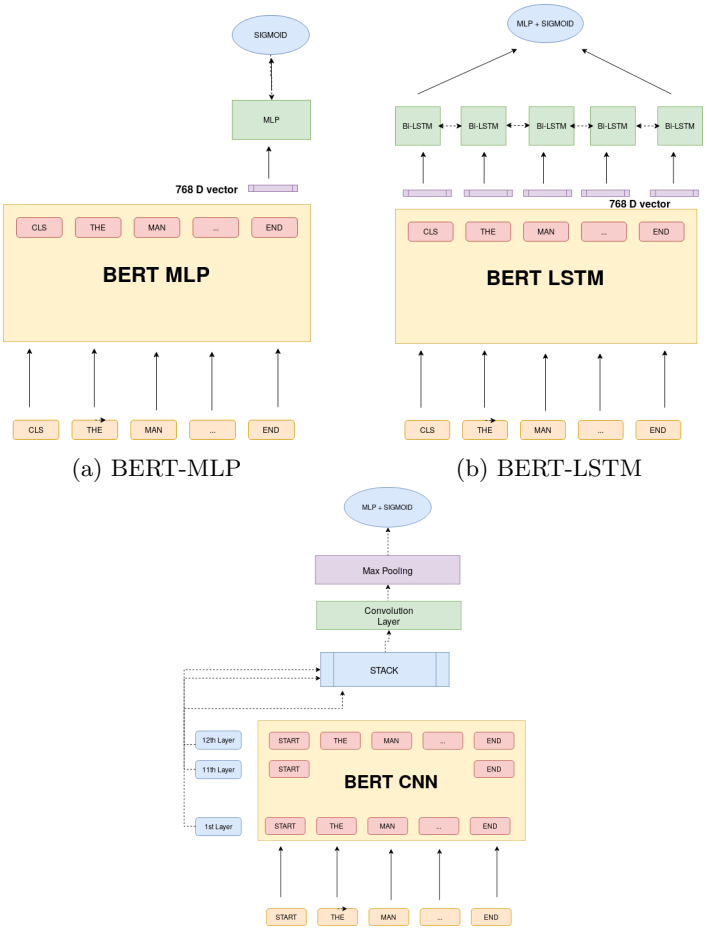
5.2.3. *Incorporating Graph Embeddings*

As each tweet was sent by a specific user, we match the tweet with the corresponding user embedding created from the **ML-IPM** model. As detailed in Section 3.2, we use a pipeline approach that treats the user embeddings from **ML-IPM** and the contextualized tweet embeddings from **BERT** as separate pipelines. This means that the user embeddings from the graph serve as additional features for the model, and can be easily substituted or replaced without affecting the **BERT** pipeline.

5.2.4. *Fine-Tuning and Implementation*

The power of **BERT** is its ability to transfer knowledge from pre-trained weights to more specific downstream natural language processing tasks in a process called fine-tuning. Unlike normal training procedures, which have a global learning rate for all the parameters of the model, fine-tuning with **BERT** uses different learning rates for different parts of the model. The exact learning rates can be found in Section 6.2. However, the parameters of the pre-trained **BERT** model are generally fine-tuned using a much smaller learning rate than the classification layer.

[b]The tweets were gathered before Twitter's change from 140 to 280 maximum characters



(a) BERT-MLP (b) BERT-LSTM

(c) BERT-CNN

Fig. 7: Different BERT Model Architectures

The full implementation of the **BERT** pipeline is completed by using a popular library called **pytorch-transformers**, which allows users to easily incorporate different transformer models such as **BERT**, **OpenAI GPT-2**, and **RoBERTa** into a custom architecture in **PyTorch**.

6. Experiments and Results

In this section, we detail the specific parameters used for the pipeline model, as well as the different results achieved by the model.

6.1. *ML-IPM Hyperparameter Summary*

For the **ML-IPM** model, we use 2 link types due to the nature of the dataset. We then have 5 dimensions for the user and a regularization parameter μ of 0.01 as recommended in [17]. We then add a step decay schedule to decrease the learning rate over time. A summary of the hyperparameters are listed in Table 2.

Hyperparameter	**Value**
Ideology Space Dimensions (K)	5
Link Types (R)	2
Regularization Parameter (μ)	0.01
Epochs	100
Initial Learning Rate	2.0
Learning Rate Decay Steps	10
Learning Rate Decay	0.75

Table 2: ML-IPM Hyperparameter Summary

6.2. *BERT Hyperparameter Summary*

The hyperparameters used to train the **BERT** model are also standardized, as listed in Table 3. Note that we use a learning rate decay scheduler to adjust the learning rate after a fixed number of epochs. Additionally, we have three variants of **BERT** that each add a different classification layer. The additional hyperparameters needed for the three variants of **BERT** are also included in the table.

Hyperparameter	**Value**
Batch Size	32
Learning Rate Scheduler Step Size	5
Gamma	0.1
Epochs	24
Optimizer	Adam
Weight Initialization	Xavier Normal
Dropout	0.1
Dense Hidden Size (BERT-MLP)	768
Convolution Out Channels (BERT-CNN)	16
LSTM Hidden Size (BERT-LSTM)	256

Table 3: BERT Hyperparameter Summary

Meanwhile, as stated in Section 5.2.4, we used different learning rates for different sections and variants of the model. The learning rate of the **BERT** section will have a lower learning rate, as this section of the model has already been pre-trained. These learning rates can be seen in Table 4.

Layer Type	**Learning Rate**
BERT-MLP Dense Layer	0.001
BERT-CNN Dense Layer	0.001
BERT-LSTM Dense Layer	0.002
Convolution Layer	0.0001
BiLSTM Layer	0.0001
BERT	0.00001

Table 4: BERT Parameter Learning Rate

6.3. *Experimental Setup and Results*

We designed 6 different experimental setups to measure the performance of our model. We first check the performance of the model by just using the three variants of the **BERT** pipeline without adding the network model pipeline. After that, we check the performance of the model by including the network model pipeline to the three variants of the **BERT** pipeline. The results are found in Table 5. Note that we report the results in terms of its prediction accuracy as well as the F_1 Score. The formulations are described in Equation 7.

$$\begin{aligned} Accuracy &= \frac{TP + TN}{TP + FP + TN + FN} \\ Precision &= \frac{TP}{TP + FP} \\ Recall &= \frac{TP}{TP + FN} \\ F_1 &= \frac{2}{Precision^{-1} + Recall^{-1}} \end{aligned} \tag{7}$$

where TP denotes the true positives, FP denotes the false positives, TN denotes the true negatives, and FN denotes the false negatives.

Note that in both forms of evaluation metrics, and in all three variants of **BERT** used, the model is able to achieve a higher score when we include the graph embeddings obtained from the network model. This seems to indicate that the study has promising results.

We would like to elaborate the **BERT-LSTM** results in more detail. Since **LSTMs** are known as the memory networks, they are suitable in modeling sequen-

Model Type	**Accuracy(%)**	F_1 **Score**
BERT-MLP	73.11	0.373
BERT-CNN	71.23	0.0
BERT-LSTM	76.89	0.473
BERT-MLP with Graph Embeddings	75.42	0.499
BERT-CNN with Graph Embeddings	77.83	0.617
BERT-LSTM with Graph Embeddings	77.83	0.552

Table 5: BERT Results Summary

tial data with interdependent sequences. However, the **LSTM** model with graph embeddings did not improve as well as the **BERT-CNN** model with graph embeddings in our experiments. This is most likely because the **BERT-CNN** with graph embeddings placed more weights on the graph network due to the low performance rate of the **BERT-CNN** model.

7. Conclusion

Multimodal classification is an area of study that involves different variations of incorporating multiple modalities. However, many existing works are focused on implementing image-text multimodality. In this work, we claim that spatial information and textual information can also be handled in multimodal representations. This improves upon approaches based on a single domain, as these models discard useful information from other domains.

We focused on incorporating both network data and textual data that are obtained from Twitter to build a tweet sentiment analysis system. The proposed way to approach this task is by either having a joint learning model or a pipeline model. The joint learning model is able to learn embeddings from two domains that are jointly trained. This means that the two embeddings are already pre-trained, which can greatly enhance the results of the classification system. Meanwhile, the pipeline model allows users to build different pipelines for the task that will later be combined. This allows for parallel work and faster results, while still achieving better results than the baseline model. As we are able to show promising results that provide proof of concept, the idea of combined representation network and textual data for tweet sentiment analysis is promising.

8. Future Work

Although the published results don't result in a state-of-the-art model, we believe that both proposed systems should be extended. In particular, the modules for the joint learning model should be explored using different deep learning methods such as **BERT** and **GCNNs**. Another possible form of exploration is in the pipeline

model, so that the language model, such as **BERT** in this paper, will be able to train the embeddings from the network model pipeline.

In addition, one other area for future work would be to employ a network model that can account for unknown nodes in the network, as **ML-IPM** can only train on an already discovered graph. Besides that, we should also investigate how the combined error is propagating across the two domains. A final issue to address in future works is to address the challenge of obtaining the ground truth data in the absence of such datasets. This is because a well-formed representation of the ground truth data can lead to a more precise classification system where the **neutral** and **bias** labels can be further discussed.

9. Contributions

9.1. *ANG, Clyde Wesley Si*

For the initial part of the project, our task was to first choose a topic that we will focus on. I initiated this part with the group and read papers from different topics to get a better idea. After we finalized our topic as social networking, we divided the papers listed in the course website. Although we were each assigned to read a specific paper and make a summary, I also read some other papers that potentially seemed interesting in case the group wanted to proceed in that direction. After we finalized on the topic of political ideologies in social networks, the next step was to write the project proposal. I contributed some of the content in the task description and proposed models, as well as did editing and proofreading for the other sections.

The next phase was the actual project. We had multiple meetings to further finalize the model. However, since the initial ideas were too idealized, I proposed to change the model slightly to have easier implementation and also found a dataset that we will be able to work on. Although I tried to contact some other paper's authors to potentially get their dataset or possibly the source code, this was not successful. I was then in charge of the data for the project. What I did was first check the validity of the data, since the dataset might be outdated or have wrong data points. After that, I used the Tweepy library to help gather the network data. Tweepy allows for Python implementation of Twitter API calls. I then gathered the follow links first, which took a long time because of Twitter rate limits, so that I can get the network nodes. Then I gathered the retweet links, and set a limit of 20 tweets per politician since it would have taken weeks to scrape all tweets.

Finally, the last phase was to write the project report. My contributions include writing most of the content in Sections 3.2 (which I rewrote), 4, 4.1, 4.2 (since I am in charge of the dataset), and parts of the conclusion. I also edited most of the other parts to make the paper cohesive and did proofreading for the final paper.

9.2. *UY, Mark Christopher*

Initially, I would say I contributed greatly to the team's search for a topic. I read multiple papers in the field of social network mining. After reading them, I sum-

marized the papers as stated and shared my thoughts and insights with my group. After a few rounds of discussion, we finally decided to do a research based work that focused on trying to predict a users' political preference from their tweets using a joint network embedding approach, and an natural language processing approach. One of group's final architecture, the parallel pipeline approach was partly suggested by me.

The next phase was the actual project itself. I did the entire BERT phase of the project. I was tasked with understanding the inner workings of BERT and setting up the entire training phase, which includes cleaning up the tweet data, defining the model architectures, and then setting up the training and validation process. I was also tasked with combining the BERT part of the project with the ML-IPM part of the project. I also then output then calculated the accuracy of the different models, and storing its results in order to allow for calculation of F1-score, which was done by one of my colleagues.

Finally, my contributions to the final paper include writing most of 5.2, 6.2, and majority of 6.3.

9.3. *WANG, Aaron Si-yuan*

During the early phases of the project, I contributed by reading papers in order to propose potential topic ideas, summarizing the contents of the papers I read for the others to better understand them and so that we can narrow down and decide on a topic for our project. During these meetings, I contributed to the discussion by thinking of and pointing out the potential challenges of the different projects we shortlisted. After we decided to go with the Social Network topic and to analyze Twitter data, I helped by researching on existing libraries for graph and network learning that we could make use of or take reference from.

For the project implementation, my main contribution has been with the graph network part of our model, namely I implemented the ML-IPM model from scratch in Python, making use of a few existing libraries. For the graph manipulation, I used NetworkX, in order to transform the edge list data into a graph, create the condensed list of the edges to calculate the loss function, as well as to generate the list of non-edges. In order to perform gradient descent on the loss function, I made use of Tensorflow, as they have a lot of support for these types of optimization tasks.

For the report, the main sections I wrote were 2.3 and 5.1, as these sections are concerning the ML-IPM model and its implementation. Besides this, I have also helped with proof-reading and editing the other sections of the report.

9.4. *KUMYOL, Serkan*

At the beginning, our group scattered to explore possible directions of research that would be common for us persuade. We finally voted for social network as first and graph learning as second directions. After we all did our diggings in social

network, I proposed that combining both decisions of the group by designing a unifying model. With regard to idea I studied and proposed the baseline paper "Ideology Detection for Twitter Users with Heterogeneous Types of Links". This in fact led me to the idea of combining NLP and graph learning to have a more accurate political knowledge representation using multimodal representation. Then I expanded my idea to two proposed systems in Section 3. The original idea was having a joint learning model for both spatial and textual modalities to detect the political ideology. Then with the aforementioned limitations and the Clyde's proposals regarding to challenges, I formulated the weighted sum model. To propose such ideas I did the literature review and tried the conceptualize our research at each step.

In actual paper here, I wrote the abstract, sections 1, 2 (except 2.3), 3, 3.1, 3.3. and a part of the conclusion and experiment reporting, finally, evaluated the experiments to conclude the results.

References

[1] Boyd, D.M. and Ellison, N.B., 2007. Social network sites: Definition, history, and scholarship. Journal of computer-mediated Communication, 13(1), pp.210-230.

[2] Franchi, E. and Tomaiuolo, M., 2013. Distributed social platforms for confidentiality and resilience. In Social Network Engineering for Secure Web Data and Services (pp. 114-136). IGI Global.

[3] Faris, R., Roberts, H., Etling, B., Bourassa, N., Zuckerman, E. and Benkler, Y., 2017. Partisanship, propaganda, and disinformation: Online media and the 2016 US presidential election. Berkman Klein Center Research Publication, 6.

[4] Barron-Cedeno, A., Da San Martino, G., Jaradat, I. and Nakov, P., 2019, July. Proppy: A system to unmask propaganda in online news. In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 33, pp. 9847-9848).

[5] Nakov, P., Ritter, A., Rosenthal, S., Sebastiani, F. and Stoyanov, V., 2016. SemEval-2016 task 4: Sentiment analysis in Twitter. In Proceedings of the 10th international workshop on semantic evaluation (semeval-2016) (pp. 1-18). Integrating Software Process Modelling and Software Measurement, Fraunhofer IESE-Report No. 047.99 (Ed.: Fraunhofer IESE, 1999), http://www.iese.fhg.de/Publications/Iese_reports/.

[6] Pang, B. and Lee, L., 2008. Opinion mining and sentiment analysis. Foundations and Trends® in Information Retrieval, 2(1–2), pp.1-135.

[7] Baldoni, M., Baroglio, C., Patti, V. and Rena, P., 2012. From tags to emotions: Ontology-driven sentiment analysis in the social semantic web. Intelligenza Artificiale, 6(1), pp.41-54.

[8] Liu, P., Joty, S. and Meng, H., 2015, September. Fine-grained opinion mining with recurrent neural networks and word embeddings. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (pp. 1433-1443).

[9] Liu, P., Joty, S. and Meng, H., 2015, September. Fine-grained opinion mining with recurrent neural networks and word embeddings. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (pp. 1433-1443).

[10] Wang, Y., Huang, M. and Zhao, L., 2016, November. Attention-based LSTM for aspect-level sentiment classification. In Proceedings of the 2016 conference on empirical methods in natural language processing (pp. 606-615).

[11] Devlin, J., Chang, M.W., Lee, K. and Toutanova, K., 2018. Bert: Pre-training

of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.

[12] Golbeck, J. and Hansen, D., 2011, May. Computing political preference among twitter followers. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (pp. 1105-1108). ACM.

[13] Wong, F.M.F., Tan, C.W., Sen, S. and Chiang, M., 2016. Quantifying political leaning from tweets, retweets, and retweeters. IEEE transactions on knowledge and data engineering, 28(8), pp.2158-2172.

[14] Wong, F., Tan, C.W., Sen, S. and Chiang, M., 2013. Media, pundits and the us presidential election: Quantifying political leanings from tweets. In Proceedings of the International Conference on Weblogs and Social Media (pp. 640-649).

[15] Wu, S., Hofman, J.M., Mason, W.A. and Watts, D.J., 2011, March. Who says what to whom on twitter. In Proceedings of the 20th international conference on World wide web (pp. 705-714). ACM.

[16] Barbera, P., 2015. Birds of the same feather tweet together: Bayesian ideal point estimation using Twitter data. Political Analysis, 23(1), pp.76-91.

[17] Gu, Y., Chen, T., Sun, Y. and Wang, B., 2016. Ideology detection for twitter users with heterogeneous types of links. arXiv preprint arXiv:1612.08207.

[18] Xu, R., Xiong, C., Chen, W. and Corso, J.J., 2015, February. Jointly modeling deep video and compositional text to bridge vision and language in a unified framework. In Twenty-Ninth AAAI Conference on Artificial Intelligence.

[19] Gu, X., Zhang, H. and Kim, S., 2018, May. Deep code search. In 2018 IEEE/ACM 40th International Conference on Software Engineering (ICSE) (pp. 933-944). IEEE.

[20] A. Karpathy and L. Fei-Fei. Deep visual-semantic alignments for generating image descriptions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 3128–3137, 2015.

[21] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł. and Polosukhin, I., 2017. Attention is all you need. In Advances in neural information processing systems (pp. 5998-6008).

[22] Le, H., Sahoo, D., Chen, N.F. and Hoi, S.C., 2019. Multimodal transformer networks for end-to-end video-grounded dialogue systems. arXiv preprint arXiv:1907.01166.