Semi-supervised Stance Detection of Tweets
Via Distant Network Supervision

[1]Subhabrata Dutta, [2]Samiya Caur, [3]Soumen Chakrabarti, [2]Tanmoy Chakraborty

[1]Jadavpur University, India; [2] IIIT-Delhi, India; [3] IIT Bombay, India

ABSTRACT

Detecting and labeling stance in social media text is strongly motivated by hate speech detection, poll prediction, engagement forecasting, and concerted propaganda detection. Today's best neural stance detectors need large volumes of training data, which is difficult to curate given the fast-changing landscape of social media text and issues on which users opine. Homophily properties over the social network provide strong signal of coarse-grained user-level stance. But semi-supervised approaches for tweet-level stance detection fail to properly leverage homophily. In light of this, We present SANDS, a new semi-supervised stance detector. SANDS starts from very few labeled tweets. It builds *multiple deep feature views* of tweets. It also uses a *distant supervision signal* from the social network to provide a surrogate loss signal to the component learners. We prepare two new tweet datasets comprising over 236,000 politically tinted tweets from two demographics (US and India) posted by over 87,000 users, their follower-followee graph, and over 8,000 tweets annotated by linguists. SANDS achieves a macro-F1 score of 0.55 (0.49) on US (India)-based datasets, outperforming 17 baselines (including variants of SANDS) substantially, particularly for minority stance labels and noisy text. Numerous ablation experiments on SANDS disentangle the dynamics of textual and network-propagated stance signals.

1 INTRODUCTION

Social media is regarded as a barometer of modern society's emotional state. Billions of social media users express their stances toward events, social issues, and political parties in their tweets, Facebook articles, or blogs. The 'target entity' may not be explicitly mentioned in the text expressing the stance. Automatic stance detection is a strongly motivated mining operation on social media and networks [24]. Some applications include hate speech detection [14], poll prediction [13], and rumor veracity detection [10]. The importance of political stance analysis over Twitter-like platforms has increased dramatically in recent times owing to several phenomena — sharp increase in partisanship among users, malicious efforts of organized groups to distort popular opinion at a large-scale, etc.

Leveraging homophily in stance detection. A large number of approaches have been proposed for text-based stance detection [1, 24, 32]. Relatively few approaches recognize and exploit the fact that text in social networks is accompanied by rich graph-structured metadata, e.g., friends, followers/followees, retweets and replies, hashtags, text-author associations, etc. [7, 23]. It is well-known that *homophily* (friends have similar taste) and *social balance* (enemy of an enemy is a friend, etc.) are pervasive in social networks. Therefore, these signals have the potential to improve the accuracy of stance prediction. Consider the tweets shown in Figure 1. The stance labels in this example include pro-Republican, anti-Republican, pro-Democrat, anti-Democrat, and neutral. Tweets from users 2–5 clearly express an anti-Democrat stance, while user 6's tweet is pro-Republican. It may be non-trivial to correctly label the tweet of user 1 (because 'she' is not identified, and the only handle @realDonaldTrump is uninformative) unless we see whom user 1 follows and what tweets they write. However, previous works connecting network dynamics for stance classification mostly deal with user-level stance analysis. An overall stance of the user often does not reflect in the tweet. For example, a Democrat supporter can tweet something pro-democrat or anti-republican. Volatile users may often switch their stances depending on the issue. For example, among the tweets we collected and annotated for the analysis presented in this work, about 8% show stance switch. On the other hand, supplementing local, tweet-level features with homophily-driven features might add bias towards the majority stance of the neighbor nodes. Instead, we seek to use homophily as a navigator of distant supervision. The end results are simple, tweet-level stance classifiers that rely on network-level information only at training time.

Scarcity of labeled data for stance classification. A key hurdle in our setting is the paucity of human-labeled data. Neural text processors are among the best for stance detection; however, they need large volumes of training data. The rapid pace of information generation and consumption over social media leads to the emergence of completely new entities and concepts (persons, events, issues, etc.), too fast for the curation of human-labeled high-quality data for each scenario. Semi-supervised and active learning are common coping mechanisms. Starting from a small set of instances manually labeled with ground-truth ('gold instances'), they expose the learner to progressively larger sets of instances that are automatically and manually labeled, respectively. Label sampling decisions are not usually informed by an overlying network.

Proposed method: SANDS Our central research question is the manner in which network signals like homophily can improve semi-supervised stance detection. Our investigation results in a system, called SANDS (**S**tance **A**nalysis via **N**etwork **D**istant **S**upervision) that starts from a few high-quality seed instances, obtains noisy labeling guidance from homophily (between a user and her followers), uses multiple views of tweet content with customized feature extraction models, and iteratively train the component learners.

In more detail, SANDS uses three kinds of textual feature extractors, designed to fit their semantics in the context of Twitter —

[1]Subhabrata Dutta, [2]Samiya Caur, [3]Soumen Chakrabarti, [2]Tanmoy Chakraborty
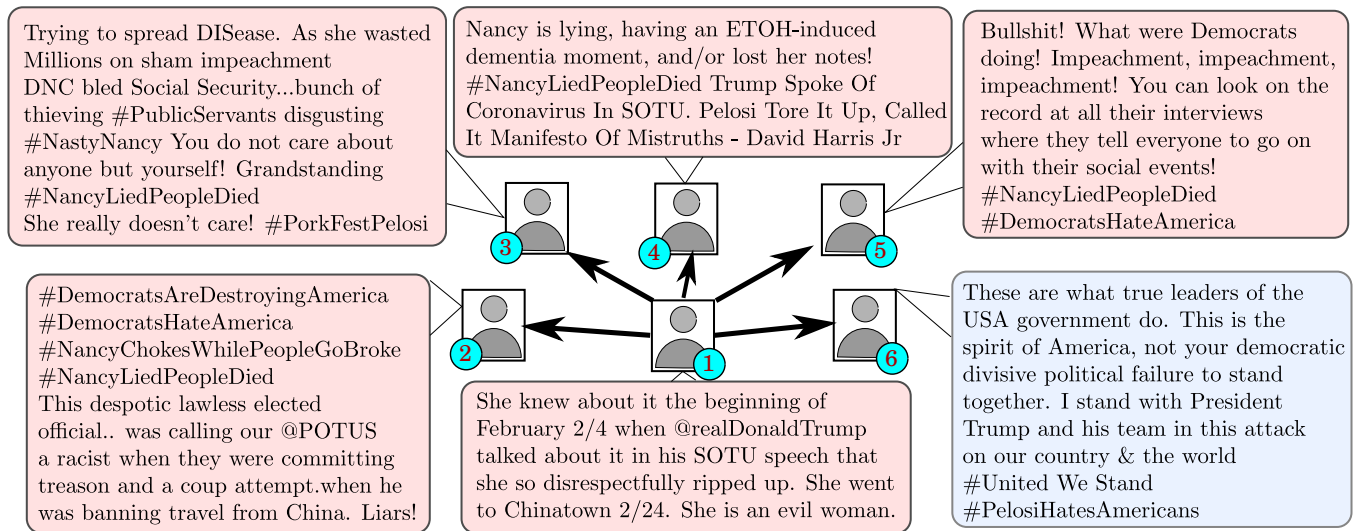
Figure 1: Stance homophily in Twitter. User 1 follows users 2, 3, 4, 5, and 6. All these users carry similar opinion, with user 6's tweet showing support to the Republicans while the rest are anti-Democrat. While the tweet posted by user 1 does not link any entity related to Republican or Democrats directly to some polarity words (thereby making the stance classification difficult), a classification framework with the knowledge of the rest of the tweets can break the ambiguity.

(i) A simple symmetric set aggregated encoding is used for hashtags. (ii) Short-range contextual text representation is captured through a convolutional network. (iii) Longer range textual context is captured through a bi-LSTM. These are combined into two main learners/predictors with somewhat different capabilities and strengths. Rather than throwing these into a standard co-training or disagreement-based learning scheme, we also make use of network information along with a homophily assumption that *followees of a user tend to have the same stance as the user*. Therefore, the predictors are also applied to recent tweets by followees of the user who posted the given tweet, and their majority vote turned into a suitable loss for the other learner.

This novel combination of *network-driven distant supervision* and *synthetic feature view separation* makes SANDS's predictive accuracy considerably superior to many recent and competitive baselines as well as ablations. SANDS is particularly effective at improving the accuracy of minority class labels.

***Two new datasets and the superiority of* SANDS.** As part of this work, we offer two large collections of politically tinted tweets from the two most vociferous social media with political content: $59,684$ and $176,619$ tweets from US and India-based users, respectively, along with the corresponding inter-user follow relations; $3,822$ and $4,185$ tweets respectively among these collections are manually annotated with corresponding stance labels. While using only $1,500$ samples as labeled data, SANDS achieves **0.55** and **0.47** macro-F1 scores on US and India-based datasets, surpassing the best baseline by (absolute) 4% and 5%, respectively.

Summary of our major contributions:
- We propose a novel stance classification framework for tweets, SANDS, which employs distant supervision using follow network information to learn tweet stances with frugal labeled data.

- Using separate neural feature extractors to focus on local and global contextual features from tweet texts separately, we formulate a learning strategy facilitated by distant network supervision that iteratively trains multiple models with two-phased view separations: local vs. global textual features within the tweet and candidate tweet vs. followee tweets over the network.
- We present two large collections of tweets from US and India-based users, along with manually annotated labels for political stance on subsets of each.
- We perform extensive experiments with multiple supervised and semi-supervised methods along with ablation variants to analyze signal importance.

Reproducibility: We detail the pre-processing and parameters of SANDS necessary to reproduce the results in the supplementary material. Also, the source codes are available on this anonymous repository: https://github.com/Annon-arch/SANDS.

2 RELATED WORK

Stance and target-dependent sentiment classification. Overlapping communities work on stance detection and target-dependent sentiment classification. Given a target entity, the task of predicting the sentiment (positive, negative, or neutral) expressed over a tweet or any other form of text is well explored, since the pioneering work by Jiang et al. [17]. Wang et al. [31] approached the problem of identifying sentiment toward multiple entities present in a tweet. Recent works have explored a variety of machine learning frameworks for the task: LSTM with attention mechanism [34], multi-task learning [15], dual transformer with graph learning [29], *inter alia*.

While the detection of stance (pro-, anti-, or none) towards a target expressed in a single tweet [24] is similar to target-dependent sentiment classification, the general literature of stance classification covers a much broader ground — stance towards a rumor

[20, 33, 35], user stance prediction [7, 28], etc. Mohammad et al. [24] presented a benchmark dataset for tweet stance classification task, focused on political tweets about the 2016 US Presidential Election. To explore stance detection in a multi-lingual scenario, Zotova et al. [37] annotated Catalan and Spanish tweets regarding the Catalan referendum. Conforti et al. [6] presented a large-scale dataset of tweets annotated with stance expressed towards several corporate events in the pharmaceutical industry. Mohammad et al. [24] showed that a linear SVM model with n-gram and several sentiment-based features achieve a consistent baseline for this task. Other approaches seek to develop sophisticated target-augmented learning of tweet representation using different neural network-based methods [1, 9, 32]. Most of these datasets (and consequently, the models) follow a similar stance labeling strategy — given a tweet and a specific target, the classes are *pro*-target, *anti*-target, or *none*. However, in the interest of higher-quality human annotation, we use a slightly different classification scheme in this work. For a given set of targets, we merge all the pro- and anti- labels in a single label set, and choose the most expressed opinion as the stance label. For example, given the target set as *Democrat* and *Republican*, our label set becomes pro-Democrat, anti-Democrat, pro-Republican, anti-Republican, and other/none; the tweet from user 6 shown in Figure 1 is associated with a pro-Republican label following its major bias, though an anti-Democrat tone is also present.

User stance and network dynamics Some studies explored the problem of identifying the stance held by a user in general, instead of specific tweets. While tweets posted by a user signal the overall user stance, there are other possible metadata as well — profile information, follow information, polls, etc. Darwish et al. [7] implemented a user-clustering approach to label them according to their stances. Stefanov et al. [28] explored a semi-supervised approach based on label propagation. Magdy et al. [23] tackled the problem of predicting future user stance based on user dynamics and network activity, in the aftermath of a major event. Their findings suggest that network-based features provide strong signals for predicting user stance.

While the aforementioned studies mostly focused on stance at a micro level, the large-scale dynamics of user stance at the macro level is another focus of interest. Himelboim et al. [16] explored the homophily properties of the social network of Twitter based on opinion bias. For half a decade now, the increasing trend of "echo chamber" formation among Twitter and other social media users have led to users increasingly getting "locked up" within communities of similar opinion [5, 12]. Lai et al. [22] observed inverse homophily, where some social ties are defined as "reply-to-messages" relationships. Conflicts of opinion among clustered user groups have been observed in platforms other than Twitter as well [11, 21].

Co-training for text classification Co-training is a semi-supervised learning approach where multiple classifiers 'teach' each other based on learning from independent views of the data [2, 19]. In the absence of naturally multi-view data, Chen et al. [4] proposed feature space partitioning to enforce different classifiers receiving a synthetic view-difference. Wan [30] exploited languages as views to apply co-training for sentiment classification of bilingual texts. Chen et al. [3] applied co-training to classify sentiments over MOOC forum posts using a synthetic difference of views generated by classifier design. They used character-level and word-level processing of text to enforce view difference.

Our proposed method, SANDS, exploits view difference at both micro (tweet instance) as well as macro (network dynamics) levels. We extract feature signals within local and global contexts of the tweet separately using two independent classifiers, imposing a feature-level synthetic view-separation. On top of this, we use the homophily property in the user-user follow network as a distant supervision strategy, to present to the learners a parallel view of the tweet stance that is distinct from the text contents. Altogether, the classifiers leverage massive unlabelled data to minimize disagreement from these two levels of view separations, along with a small set of labeled data to minimize the task specific loss.

3 PROPOSED METHOD

3.1 Preliminaries and overview

Let $\mathcal{U}$ be a set of users in a (partial) social network of Twitter, where $u_i \rightarrow u_j$ signifies user u_i *follows* user u_j, $u_i, u_j \in \mathcal{U}$. The followee set of u_i is defined as $f(u_i) := \{u_j | u_i \rightarrow u_j\}$. Let $\mathcal{D} := \{\tau_t^i | 0 \leq t \leq T\}$ denote the temporally ordered set of tweets over a period of time $[0, T]$, where τ_t^i is the tweet posted by user u_i at time t. Given a tweet τ_t^i, we define its stance to be a categorical distribution over a specified label set $\mathcal{L}$. A small subset of $\mathcal{D}$ is manually annotated, denoted by $\mathcal{D}_s = \{(\tau_t^i, l_t^i) | \tau_t^i \in \mathcal{D}, l_t^i \in \{0, 1\}^{|\mathcal{L}|}\}$, where l_t^i represents the one-hot class label for tweet τ_t^i. For example, the tweet from user 1 in Figure 1 has the corresponding stance label **Anti-Democrat**, where the label set $\mathcal{L}$ is [**Pro-Democrat**, **Anti-Democrat**, **Pro-Republican**, **Anti-Republican**, **Other**]. A stance classifier in our setting can be defined as a mapping $C : y^i = C(\tau_t^i | \theta)$, where $y \in (0, 1)^{|\mathcal{L}|}$ is the predicted distribution of label probability, and θ is the set of parameters of the model.

The classical co-training paradigm requires the existence of two mutually independent views of the data [2]. In our case, the data source (a single tweet) does not provide any such natural partitioning. We seek to mitigate this by introducing two major design decisions. First, we design two separate classifiers to focus on local and global textual contexts of the tweet, thereby aggregating separate feature signals for stance polarity from a single tweet independently. Second, we exploit stance homophily over Twitter [16] to provide another avenue of incorporating multiple views; we assume that the stance polarity expressed by a user u_i in tweet τ_t^i can be independently inferred from the polarities of the previous tweets from its followees, $\{\tau_{t-}^{f(u_i)_j}\}$.

3.2 Classification models

Here, we describe the architectures of two classifiers we use for our task. Typically, a tweet consists of a piece of text body and one or more hashtags. For both our classifiers, these two input components are processed independently at the initial stages and fused before the final prediction. We represent the textual body of the tweet as a sequence of words $W_1, W_2, \cdots, W_n$ and the hashtags as a list $\#H_1, \#H_2, \cdots, \#H_m$. To learn the feature representations from the hashtags, both the classifiers use a similar strategy based on self-attention. However, to enforce learning pseudo-separate views from the text body, the classifiers use different methods of learning text

[1]Subhabrata Dutta, [2]Samiya Caur, [3]Soumen Chakrabarti, [2]Tanmoy Chakraborty

representations: C_1 uses a *convolutional* architecture focusing on short, contiguous segments of the text (i.e., local context) while C_2 uses a *bidirectional LSTM-based* one which encodes distantly separated textual signals into one single representation (i.e., global context)[1].

3.2.1 Learning hashtag representations.
We use an embedding layer to transform the list of one-hot representations of hashtags #H_i to a list of d_H dimensional dense vectors $\mathbf{X^H} = \{x_i^H\}$.

Next, to control the contribution of each of these vectors for the prediction task, we compute *scaled dot product attention* between them. For this, we compute the *query*, *key*, and *value* vectors from each of x_i^H as, $q_i = \text{ReLU}(\mathbf{W}_q x_i^H + \mathbf{B}_q)$; $k_i = \text{ReLU}(\mathbf{W}_k x_i^H + \mathbf{B}_k)$; $v_i = \text{ReLU}(\mathbf{W}_v x_i^H + \mathbf{B}_v)$, where $\mathbf{W}_q, \mathbf{W}_k, \mathbf{W}_v, \mathbf{B}_q, \mathbf{B}_k, \mathbf{B}_v$ are learnable weight and bias matrices. We compute the attention weights and scale the representations for each hashtag as follows:

$$\alpha_{ij} = \frac{\exp(q_i^\top k_j)}{\sum_j \exp(q_i^\top k_j)}; \qquad \hat{x}_i^H = \sum_j \alpha_{ij} v_j$$

Finally, we apply max-pooling over $\{\hat{x}_i^H\}$ to compute the combined representation of the hashtags, $\mathbf{Z}^H$.

3.2.2 Convolutional feature extraction.
Given a sequence of one-hot word representations $\{W_i\}$, an embedding layer maps them to d_W-dimensional vectors $\mathbf{X}^W = \{x_i^W\}$. Then, we apply three branches of consecutive 1d-convolution and max-pooling operations with window sizes 1, 3, and 5 on $\mathbf{X}^W$. A single pair of convolution-pooling operation can be represented as $\mathbf{h}_{i+1} = \text{Maxpool}(\text{ReLU}(\text{Conv}(\mathbf{h}_i)))$. For three successive such operations within a branch of window size k, $\mathbf{h}_0 = \mathbf{X}^W$ and $\mathbf{h}_3 = \hat{\mathbf{Z}}_{\text{conv}-k}^W$. Finally, outputs of each of these branches along with the hashtag representation $\mathbf{Z}^H$ are concatenated, normalized, and forwarded to the prediction feed-forward layer with softmax activation:

$$\mathbf{Z}_1 = \text{LayerNorm}([\mathbf{Z}^H : \mathbf{Z}_{\text{conv}-1}^W : \mathbf{Z}_{\text{conv}-3}^W : \mathbf{Z}_{\text{conv}-5}^W])$$
$$y_1 = \text{Softmax}(\mathbf{W}_{p1}\mathbf{Z}_1 + \mathbf{B}_{p1})$$

where $\mathbf{W}_{p1}$ and $\mathbf{B}_{p1}$ are learnable kernel and bias matrices of the feed-forward layer, $(\cdot : \cdot)$ denote the concatenation operation, and $y_1 \in (0,1)^{|\mathcal{L}|}$ is the predicted class probability. Figure 7 in the supplementary material contains the detailed information flow.

3.2.3 Bi-LSTM-based feature extraction.
Similar to its convolutional counterpart, the bidirectional LSTM-based module also uses an embedding layer to map one-hot words into fixed dimensional vectors of size d_W. A bidirectional LSTM layer then maps these vectors into an intermediate sequential representation $\mathbf{h}_{\text{blstm}} = \{h_i\}$, which, upon max-pooling along the sequence axis, produces the representation of the tweet body $\mathbf{Z}_{\text{blstm}}$. This is then concatenated with the hashtag representation, normalized, and passed to the prediction layer, similar to Section 3.2.2:

$$\mathbf{Z}_2 = \text{LayerNorm}([\mathbf{Z}^H : \mathbf{Z}_{\text{blstm}}])$$
$$y_2 = \text{Softmax}(\mathbf{W}_{p2}\mathbf{Z}_2 + \mathbf{B}_{p2})$$

Henceforth, we denote the convolutional and bi-LSTM based models as C_1 and C_2, respectively. Additionally, each sub-block of operations in each model except the final feed-forward layer is

---

[1]Experiments with pretrained BERT did not yield satisfactory results; see Section 5.5
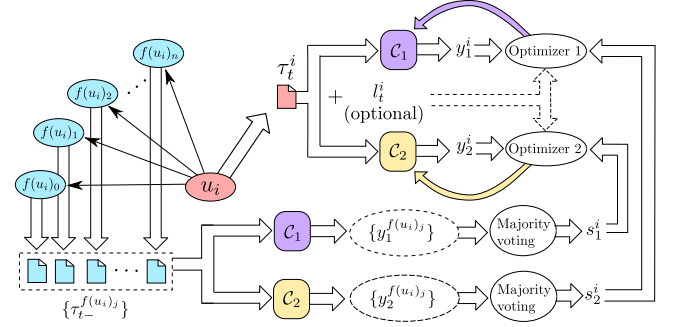
Figure 2: Training process of SANDS. Classifiers C_1 and C_2 independently predict the stance for a tweet from user u_i. In parallel, the classifiers generate pseudo-label sets $\{y_1^{f(u_i)_j}\}$ and $\{y_1^{f(u_i)_j}\}$, respectively from the recent tweets by followees of u_i. After majority voting, lables s_1^i and s_2^i are selected from pseudo-labels. C_1 is optimized using the label generated by C_2, and vice versa. Additionally, annotated labels l_t^i (if present) of the tweet are also used to compute loss and optimize the classifiers.

followed by a dropout layer with probability $p_{\text{dropout}} \in (0,1)$ to handle overfitting and noise.

3.3 Training method
Our proposed training methodology in SANDS consists of two repeating phases. First, each of the classifiers C_1 and C_2 defined in Section 3.2 is optimized separately on the labeled dataset $\mathcal{D}_s$ in a purely supervised manner. Then, we incorporate the full dataset (labeled as well as unlabeled) to jointly train the classifiers. In this phase, relying on the homophily of user opinions, the classifiers generate labeling of tweets depending on the network level view (i.e., recent tweets from followees) for each other. This way, the local context aggregating classifier C_1 receives the alternate, global context-based, network-level view from C_2, and vice versa.

3.3.1 Supervised phase.
Given the annotated dataset $\mathcal{D}_s$, in the supervised phase, each of the classifiers C_1 and C_2 is trained independently. Formally, given a tweet-label pair $(\tau_t^i, l_t^i) \in \mathcal{D}_s$, a forward pass on the models computes the predicted class probabilities $y_1^i = C_1(\tau_t^i|\theta_1)$ and $y_2^i = C_2(\tau_t^i|\theta_2)$. Then two independent gradient-descent based optimizers minimize the following loss functions:

$$\mathcal{J}_1^s = -\omega_s(l_t^i) \sum_{j=1}^{|\mathcal{L}|} \mu_j \log \nu_{1,j}; \;\; \mathcal{J}_2^s = -\omega_s(l_t^i) \sum_{j=1}^{|\mathcal{L}|} \mu_j \log \nu_{2,j}$$

where $\mu_j \in l_t^i$, $\nu_{1,j} \in y_1^i$, $\nu_j \in y_2^i$ are probability of the j-th class in the ground-truth, prediction of C_1 and prediction of C_2, respectively. ω_s is a weighting function to handle class imbalance, defined as:

$$\omega_s(l_t^i) = \log \frac{|\mathcal{D}_s|}{|\{l_{t'}^j|l_{t'}^j = l_t^i, \tau_{t'}^j \neq \tau_t^i \forall \{\tau_{t'}^j, l_{t'}^j\} \in \mathcal{D}_s\}| + \epsilon}$$

where ϵ is a very small real value to avoid division by zero.

3.3.2 Semi-supervised phase.
In Figure 2, we outline the steps of our proposed semi-supervised phase. After the purely supervised

pass discussed earlier, for any given tweet $\tau_t^i \in \mathcal{D}$, we use the two models to generate pseudo-labels for the last tweets from $f(u_i)$, i.e, the users followed by u_i:

$$y_1^{f(u_i)_j} = C_1(\tau_{t-}^{f(u_i)_j}|\theta_1); \; y_2^{f(u_i)_j} = C_2(\tau_{t-}^{f(u_i)_j}|\theta_2)$$

where $f(u_i)_j$ denotes the j-th followee of u_i, and $\tau_{t-}^{f(u_i)_j}$ denotes the most recent tweet posted by $f(u_i)_j$ before t.

Next, we move on to select labels for the unsupervised part from these pseudo-label sets. One approach can be to compute *soft* labels from the sets $\{y_1^{f(u_i)_j}\}_{j=1}^{|f(u_i)|}$ and $\{y_2^{f(u_i)_j}\}_{j=1}^{|f(u_i)|}$ using simple methods like finding the mean. However, similar to [27], we empirically find that such soft labels do not help much. Instead, we compute *hard* or one-hot encoded labels s_1^i (*resp.* s_2^i) using majority voting from $\{y_1^{f(u_i)_j}\}_{j=1}^{|f(u_i)|}$ $\left(resp. \{y_2^{f(u_i)_j}\}_{j=1}^{|f(u_i)|}\right)$ as follows:

$$I_1^{f(u_i)} = \{\mathrm{argmax}(y_1^{f(u_i)_j})\}_{j=1}^{|f(u_i)|}$$
$$C_1^{f(u_i)} = \{\mathrm{freq}(k, I_1^{f(u_i)})\}_{k=0}^{|\mathcal{L}|-1}$$
$$s_1^i = \mathrm{one\text{-}hot}(\mathrm{argmax}(C_1^{f(u_i)}))$$

where $\mathrm{freq}(x, X)$ returns the frequency of element x in an array X, and $\mathrm{one\text{-}hot}(k)$ returns one-hot encoded vector of size $|\mathcal{L}|$ with k-th element being 1.

Additionally, we devise a sample weighing function ω_u to handle class imbalance in the unsupervised labels. Given a mini-batch of inputs $\mathbf{B} \subset \mathcal{D}$ (see Section 3.4 for the mini-batch creation policy), let $\mathcal{S}_1 = \{s_1^i \forall t_1^i \in \mathbf{B}\}$. Then

$$\omega_{u,1}^{\mathbf{B}}(s_1^i) = \log \frac{|\mathbf{B}|}{\mathrm{freq}(s_1^i, \mathcal{S}_1) + \epsilon}$$

Finally, the pseudo-labels generated by C_1 are used to optimize C_2 and vice-versa. The classifiers predict the class probability distributions y_1^i and y_2^i from the input tweet τ_t^i in the forward passes of C_1 and C_2, respectively. The objective functions to minimize are:

$$\mathcal{J}_1^u = -\omega_u(s_2^i)\sum_{j=1}^{|\mathcal{L}|}\sigma_{2,j}\log v_{1,j} - \omega_s(l_t^i)\sum_{j=1}^{|\mathcal{L}|}\mu_j\log v_{1,j}$$

$$\mathcal{J}_2^u = -\omega_u(s_1^i)\sum_{j=1}^{|\mathcal{L}|}\sigma_{1,j}\log v_{2,j} - \omega_s(l_t^i)\sum_{j=1}^{|\mathcal{L}|}\mu_j\log v_{2,j}$$

where $\sigma_{1,j} \in s_1^i$, $\sigma_{2,j} \in s_2^i$, $v_{1,j} \in y_1^i$, $v_{2,j} \in y_2^i$ and $\mu_j \in l_t^i$. The right-hand components of the losses correspond to the supervised loss calculated from the annotated label l_t^i, if available, and 0 otherwise.

3.4 Further implementation details of SANDS

As discussed in Section 3.3.2, to compute the pseudo-labels for τ_t^i, we need a forward pass of both the models on the last tweets till t posted by $f(u_i)$. However, if there are multiple consecutive tweets from u_i where most of its followees have not posted a new tweet, we tend to perform redundant, costly forward pass computations. To deal with this, we initialize two matrices $\mathbf{Y}_1$ and $\mathbf{Y}_2$ with the predictions of C_1 and C_2 (right after their supervised training phases), respectively, for the first tweets from each user in $\mathcal{U}$. Then, at each semi-supervised pass, both these matrices are updated with the predictions from C_1 and C_2. We maintain a sparse adjacency matrix of the follow-network for $O(1)$ time access of $\{y_1^{f(u_i)_j}\}_{j=1}^{|f(u_i)|}$ from

$\mathbf{Y}_1$ and $\mathbf{Y}_2$. This way, a single epoch of the semi-supervised training phase is done with $O(|\mathcal{D}|)$ forward passes for each classifier.

For further speed-up of the training procedure, we opt for mini-batch learning. However, to ensure that $\mathbf{Y}_1$ and $\mathbf{Y}_2$ are updated in a correct temporal order, we maintain the following two conditions:

(1) Given consecutive batches of input tweets $\mathbf{B}_k$ and $\mathbf{B}_{k+1}$, for any pair of tweets $\tau_{t1}^i \in \mathbf{B}_k$ and $\tau_{t2}^i \in \mathbf{B}_{k+1}$, $t1 < t2$.
(2) For any pair of tweets τ_{t1}^i and τ_{t2}^j in the same batch, $u_i \notin f(u_j)$ and vice versa.

Hyperparameter details for implementation of SANDS and the classifiers (learning rate, batch size, minimum followee tweets to threshold, dropout probability, etc.) are detailed in Appendix C in the supplementary material.

4 EXPERIMENTS

4.1 Dataset

None of the existing on tweet stance classification datasets include explicit user-user follower information necessary for our experiments. Moreover, due to the ever-changing dynamics of Twitter itself, crawling such additional information related to the old collection of tweets often results in missing data due to deleted tweets, suspended accounts, etc. For this reason, we proceed to collect and annotate our own tweet dataset from scratch. We specifically focused on political tweets from two different demographics: US and India, targeting the opinion dynamics of users around major political parties in these two countries. Details of dataset collection and annotation are described in Appendix A in the supplementary material.

From US-based users, we collected with a total of $59,684$ tweets from $24,490$ users, with an average of 2.44 tweets and 41.79 followees per user, within the period from 1st January, 2020 to 3rd April, 2020. We refer to this dataset as **StanceUS**. For the Indian counterpart, our crawling period ranges from 26th October, 2019 to 2nd March, 2020, resulting in a total of $176,619$ tweets from $63,230$ users, with an average of 2.79 tweets and 22.93 followees per user. We refer to this dataset as **StanceIN**.

For initial supervised training and final evaluation of SANDS as well as the baselines, we proceed to manually annotate a randomly selected subset of the collected data according to the stance they express toward leading political parties. For US-based tweets, we annotate each tweet with one among the following labels: **Pro-Dem**, **Anti-Dem**, **Pro-Rep**, **Anti-Rep**, and, **Other**. In case of India-based tweets, the labels are **Pro-BJP**, **Anti-BJP**, **Pro-INC**, **Anti-INC**, **Pro-AAP**, **Anti-AAP**, and **Other**. We finally end up with $3,822$ and $4,185$ annotated tweets from US and India with inter-annotator agreements 0.84 and 0.78 Cohen's κ, respectively. The distribution of annotated samples among different classes is shown in Table 1. To investigate the effects of the size of the labeled training data, we train and test SANDS and all the baselines on three different training sets of size 500, 1000, and 1500.

4.2 Baselines and ablation variants

We employ multiple supervised as well as semi-supervised methods to compare with SANDS. We also remove the contextual view difference and distant network supervision in SANDS step-by-step to

[1]Subhabrata Dutta, [2]Samiya Caur, [3]Soumen Chakrabarti, [2]Tanmoy Chakraborty

Classes	Train-500	Train-1000	Train-1500	Test
StanceUS				
Pro-Dem	320	654	981	1543
Anti-Dem	9	22	32	46
Pro-Rep	133	253	381	576
Anti-Rep	13	17	29	46
Other	25	54	77	111
Total	500	1000	1500	2322
StanceIN				
Pro-BJP	67	149	208	360
Anti-BJP	136	275	425	680
Pro-INC	24	60	83	158
Anti-INC	2	3	6	15
Pro-AAP	35	61	99	142
Anti-AAP	52	101	163	210
Other	184	351	516	1120
Total	500	1000	1500	2685

Table 1: Class-wise sample distribution in different training and testing splits of the annotated datasets.

investigate their relative importance. Following are the supervised baselines we implement:

SVM. A linear kernel Support Vector Machine framework with different text-based features, proposed by Mohammad et al. [25].

TAN. A bidirectional-LSTM based framework with target-specific attention mechanism, proposed by Du et al. [9].

SiamNet. A siamese adaptation of LSTMs, following the works of Mueller and Thyagarajan [26] and Conforti et al. [6].

BICE. As proposed by Augenstein et al. [1], this model computes the tweet representation conditioned on targets using Bi-LSTMs.

BERT. Pretrained BERT[8] (base, uncased) followed by a feed-forward layer for stance prediction.

ConvNet. A convolutional model (same as C_1 in SANDS) trained in a *purely supervised* manner using the labeled data only.

BLSTM. A bidirectional LSTM model (same as C_2) trained in a *purely supervised* manner using the labeled data only.

In the following six semi-supervised baseline methods, we keep the base classifiers among the supervised ones:

LP-SVM. A semi supervised approach based on label spreading [36] using the SVM model as the base classifier.

ST-ConvNet. A vanilla self-learning based semi-supervised method using ConvNet as the base classifier. This framework selects k-best samples (based on predicted class probability) from the unlabeled dataset and augments the labeled data at each iteration.

ST-BLSTM. Similar to ST-ConvNet, it replaces the base classifier with the BLSTM model.

UST. A semi-supervised approach, as proposed by Mukherjee and Awadallah [27] which uses a self-training framework along with BERT for text classification, tailored to our defined categories of stance.

GCN-ConvNet A semi-supervised node-classification approach similar to [18] with ConvNet as node feature extractor followed by two graph convolution layers on the follow-network.

GCN-BLSTM Similar to GCN-ConvNet with a BiLSTM layer instead of ConvNet.

Ablation variants of SANDS. First, we remove the distant network supervision in SANDS (i.e, no follow network information is used) and reduce it to a vanilla co-training framework with no sophisticated label selection strategy; for each unlabeled instance,

Model	StanceUS			StanceIN		
	$\|\mathcal{D}_s\|$ =0.5K	$\|\mathcal{D}_s\|$ =1K	$\|\mathcal{D}_s\|$ =1.5K	$\|\mathcal{D}_s\|$ =0.5K	$\|\mathcal{D}_s\|$ =1K	$\|\mathcal{D}_s\|$ =1.5K
SiamNet	0.39	0.43	0.42	0.12	0.14	0.13
BICE	0.27	0.30	0.33	0.16	0.17	0.23
TAN	0.38	0.46	0.45	0.14	0.14	0.17
SVM	0.37	0.37	0.45	0.13	0.13	0.16
BERT	0.39	0.50	0.51	0.17	0.17	0.21
ConvNet	0.37	0.43	0.45	0.35	0.40	0.41
BLSTM	0.35	0.43	0.44	0.31	0.39	0.38
LS-SVM	0.39	0.42	0.44	0.18	0.19	0.18
ST-ConvNet	0.13	0.15	0.16	0.10	0.11	0.11
ST-BLSTM	0.13	0.16	0.19	0.09	0.12	0.11
UST	0.35	0.42	0.41	0.12	0.16	0.16
GCN-ConvNet	0.41	0.45	0.47	0.33	0.35	0.40
GCN-BLSTM	0.39	0.42	0.46	0.36	0.41	0.42
SANDS/Net.(C_1)	0.32	0.41	0.42	0.10	0.12	0.15
SANDS/Net.(C_2)	0.36	0.46	0.46	0.28	0.31	0.37
SANDS/Cont.(C_1)	0.41	0.47	0.49	0.36	0.41	0.43
SANDS/Cont.(C_2)	0.47	0.51	0.53	0.38	0.44	0.45
SANDS(C_1)	**0.46**	**0.47**	**0.49**	**0.37**	**0.42**	**0.45**
SANDS(C_2)	**0.49**	**0.53**	**0.55**	**0.42**	**0.45**	**0.47**

Table 2: F1 scores of all models with different sizes of labeled training data on StanceUS and StanceIN.

the label predicted by C_1 is used to train C_2 and vice versa. We denote this strategy as **SANDS/Net.** and the resulting classifiers as SANDS/Net.(C_1) and SANDS/Net.(C_2).

Next, to remove the context-based view difference, we use a single classifier in SANDS which computes the majority label from followee tweets for itself. We denote this method as **SANDS/Cont.** and the resulting classifiers as SANDS/Cont.(C_1) and SANDS/Cont.(C_1).

5 RESULTS AND DISCUSSION

To evaluate the performance of SANDS along with the baseline models, we use macro-averaged F1 scores. SANDS trains two separate classifiers C_1 and C_2 jointly; however, they are evaluated as independent models on the test sets. We denote them as SANDS(C_1) and SANDS(C_2), respectively.

5.1 Overall performance

Table 2 presents the performance of SANDS along with the baseline classifiers for three different training splits on the two datasets. Clearly, SANDS induces superior classification power in both the classifiers compared to their purely supervised counterparts. The convolutional classifier achieves 0.09, 0.04, and 0.04 points gain in macro-F1 over its supervised version with 500, 1000, and 1500 labelled data, respectively on StanceUS. On StanceIN, these gains are 0.02, 0.02, and 0.04, respectively. The performance gains for the bi-LSTM based classifier with SANDS are even more remarkable: 0.14, 0.10, and 0.11 on the StanceUS while 0.11, 0.06, and 0.09 on StanceIN on three different splits respectively. One may relate this disparity in performance gain with the intuitive fact that, bi-LSTMs can capture more complex, long-distance dependencies within the elements of the input text compared to convolutional operations; while small supervised training data hinders this superior representation power to be exploited, with the large volume of data utilized by SANDS empowers the bi-LSTM model to its full potential, thereby accounting for the sharper gain.
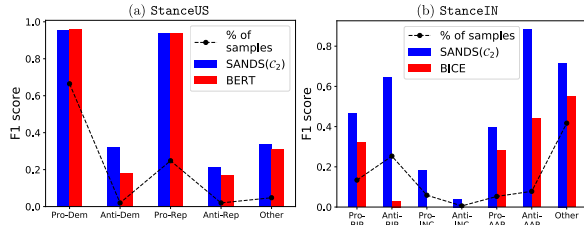
Figure 3: Comparison of class-wise F1 scores between SANDS(C_2) and best supervised baselines – (a) BERT in US data and (b) BICE in India data.

Among the external baselines, BERT and BICE emerge as the best-performing ones in US and India datasets, respectively. However, all of these models perform poorly for India-based tweets compared to the US counterpart. A prominent reason behind this difference is the variation in the linguistic quality of tweets from these two demographics. While US-based tweets are purely in English, Indian tweets often contain code-mixed tokens and noisy language. The larger size of the label set in the Indian case is also responsible for the confusion in classification.

Not surprisingly, vanilla self-training with each of the classifiers (ST-ConvNet and ST-BLSTM) affects the predictive power severely, due to the absence of both contextual as well as text-network view differences. Moreover, these two models do not use any sophisticated label selection methods. With UST, the later problem is resolved with an uncertainty-aware selection strategy. But again, in the absence of network-guided view exploited by SANDS, UST does not provide any comparable performance. As UST relies purely on textual signals for learning, the performance gap is more evident on StanceIN due to a higher degree of noise.

GCN-ConvNet and GCN-BLSTM are the two baselines that, apart from SANDS, relies on network information. Each of these models show comparative gains with respect to their base counterparts (ConvNet and BLSTM) in most of the cases. However, the models need the network input both at the time of training and testing. This incurs large computation cost at the time of prediction. The classifiers trained using SANDS requires the network-level information only at the time of training to acquire the knowledge about data-distribution from the unlabeled samples. So GCN-based approaches have a relatively higher chance to bias the prediction towards the majority stance of the neighboring nodes.

5.2 Handling class imbalance

Most real-world datasets suffer from imbalanced ratios of labels, and classifiers that are trained on such data tend to be biased towards dominant classes. This scenario remains the same in our dataset as well, evident from the sample distributions as presented in Table 1.

We start with investigating the effects of class imbalance on SANDS in comparison with the best performing external baselines on each dataset. In Figure 3(a), we plot the class-wise F1 scores for SANDS(C_2) vs. BERT, on StanceUS trained with 1500 labelled instances. While for the dominant classes like **Pro-Dem** or **Pro-Rep**, both the models perform comparably, in case of minority classes SANDS(C_2) outperforms BERT by a notable margin. Similar is the case on StanceIN, shown in Figure 3(b), where the best performing external baseline is BICE. For rarely represented labels
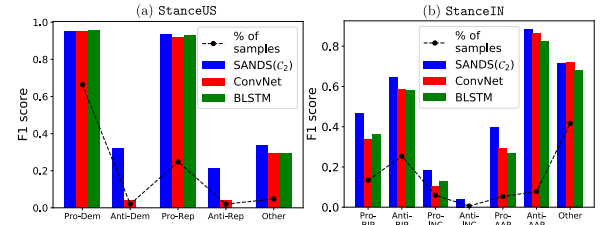
Figure 4: Comparison of class-wise F1 scores of SANDS(C_2) with the supervised counterparts, ConvNet and BLSTM on (a) StanceUS and (b) StanceIN. All of these frameworks use 1500 labelled data instances for training. SANDS provides better prediction performance on samples from minority classes compared to the supervised ConvNet and BLSTM.

like **Pro-INC** or **Anti-INC**, BICE did not predict even a single sample on the test data.

While Figure 3 provides evidence for the superior performance of SANDS compared to the external baselines for handling biased label distribution, one may suspect whether this betterment is intrinsic to the training methodology of SANDS, or the classifiers C_1 and C_2 themselves handle the biased samples. To seek further insights, we compare SANDS(C_2) with the purely supervised versions of the classifiers, ConvNet and BLSTM, for class-wise performance. In Figure 4, we can observe that both classifiers suffer heavily due to label imbalance. The supervised BLSTM model can not predict any sample from the two minority classes in StanceUS, while ConvNet performs very poorly. In both cases, SANDS provides substantial betterment. Similarly on StanceIN, where both the supervised classifiers could not predict any samples with the **Anti-INC** label while SANDS(C_2), (although poorly) provides some.

All of the supervised classifiers (external as well as ConvNet and BLSTM) take the class imbalance manifested within training samples into account with some sample weighting strategy (similar to ω_s in Section 3.3.1). Instead of this, these classifiers perform miserably on minority labels compared to SANDS. The vast majority of unlabelled data used in SANDS provides C_1 and C_2 with stronger feature signals for such labels that are absent in the small amount of labeled training data. Moreover, the loss weighing parameter ω_u (see Section 3.3.2) devised in SANDS forces the models to pay attention to the samples with under-represented stance labels.

5.3 Importance of network and feature context

The relative importance of distant supervision from network dynamics and synthetic difference of contextual views imposed by the classifier designs can be interpreted from the performances of SANDS/Net. and SANDS/Cont. classifiers in Table 2. For both datasets, the performance drops for each of the ablation variants from the original setting are evident. However, without distant network supervision, the classifiers suffer more severely: 0.07 (0.09) points drop in macro-F1 score C_1 (C_2) compared to 0.0 (0.02) points drop in the absence of contextual view difference on StanceUS.

The absence of network supervision affects the performance more drastically in the case of StanceIN. The predictions of the convolutional classifier (C_1) are essentially random. While both the ablation variants show unstable behavior while training progresses, the fluctuations in SANDS/Net. models on StanceIN are much significant. This difference of effects of network supervision over US and

[1]Subhabrata Dutta, [2]Samiya Caur, [3]Soumen Chakrabarti, [2]Tanmoy Chakraborty
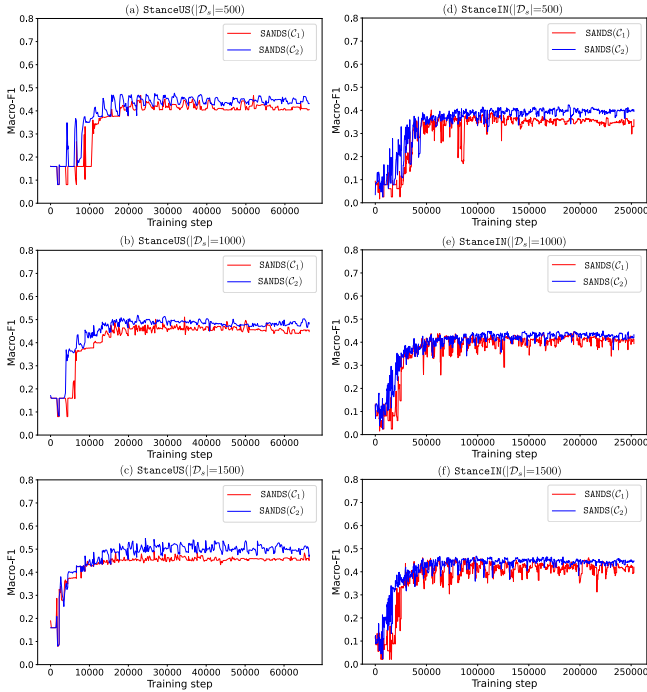
Figure 5: Macro-F1 scores of SANDS(C_1) and SANDS(C_2) on test set as training progress on different training splits for StanceUS and StanceIN. In each case, SANDS provides stable convergence.

India-based tweets further strengthens our previous observation — network dynamics play a more important role for highly noisy tweets in the Indian context.

5.4 Stability and convergence of learning

Over and above performance gains, a training protocol must provide stable, convergent learning of the underlying model(s). To investigate SANDS for this property, we continue training for a sufficient number of optimization steps and track the performance of each of the classifiers C_1 and C_2 on the test set. In Figure 5, we plot the resulting progress for each of the datasets with different amount of available labeled training data. We can observe that, in each case, both the models reach a stable convergence overcoming the periodic fluctuations at the initial phases. However, these fluctuations are not classifier-specific and vary with datasets as well as the size of labeled data. For example, on StanceIN, C_1 shows more fluctuations in performance even at the later stages, compared to C_2. However, in the US counterpart, this behavior reverses with a more stable convergence of C_1 than C_2. In both the datasets, with more labeled data, the classifiers reach stability faster. Such dependence is trivial, as with smaller labeled data the classifiers search around the parameter space in a more unguided manner.

5.5 Choice of classifier pairs

We experiment with multiple different choices of classifier designs for C_1 and C_2 with SANDS and summarize the results in Table 3. One can observe the differences in the performance of the same classifier when paired with two different ones; e.g., a convolutional

Classifier pair	StanceUS		StanceIN	
(C_1, C_2)	m-F1 C_1	m-F1 C_2	m-F1 C_1	m-F1 C_2
Conv, bi-LSTM	0.49	0.55	0.45	0.47
Conv, Conv	0.48	0.48	0.43	0.43
bi-LSTM, bi-LSTM	0.52	0.52	0.45	0.45
Conv, BERT	0.50	0.52	0.44	0.45
BERT, bi-LSTM	0.53	0.53	0.42	0.45

Table 3: Macro-F1 scores of different classifier-pairs with SANDS. The size of labelled dataset used for all these pairs is 1500.

% unlabeled	StanceUS		StanceIN	
data used	m-F1 C_1	m-F1 C_2	m-F1 C_1	m-F1 C_2
100	0.49	0.55	0.45	0.47
80	0.49	0.54	0.44	0.46
50	0.46	0.51	0.42	0.43
30	0.45	0.48	0.42	0.41
10	0.45	0.45	0.41	0.39

Table 4: Macro-F1 scores of SANDS with different amount of unlabeled data used in semi-supervised phase. The size of labeled dataset used is 1500.

classifier performs better when paired with BERT instead of bi-LSTM. However, neither of these two classifiers could outperform bi-LSTM when paired with a convolutional classifier. When the same design is chosen for both classifiers, we remove the view-difference enforced by feature signal encoding from local and global contexts, and the settings become essentially similar to the ablation variant SANDS/Cont., showing a very similar performance to SANDS/Cont.(C_1) and SANDS/Cont.(C_2).

5.6 Variation in the size of unlabeled data

Finally, we seek to explore how the amount of unlabeled data used in SANDS effects the performance of each of the classifiers. We experiment with 80%, 50%, 30%, and 10% of the original amount of unlabeled data for both the datasets and present the results in Table 4. In both cases, the effect of data reduction manifests more sharply over the performance of C_2, i.e., the bi-LSTM classifier. This is at par with the fact that C_2 received a sharper gain in predictive power from supervised to semi-supervised setting (see Section 5.1). Also, the dip in performance with decreasing amount of unlabeled data is more evident on StanceIN compared to its US counterpart.

6 CONCLUSION

We presented SANDS, a novel stance classification framework that uses distant supervision by social network properties to exploit large-scale unlabeled data along with very few manually labeled data, to predict political stance expressed over tweets. We experimentally validated the superiority of our proposed method in comparison to several supervised as well as semi-supervised classification frameworks on two tweet stance datasets we collected and annotated from US and India-based users. Particularly in settings with small annotated data with noisy textual representations and highly imbalanced class-wise sample distribution, SANDS improved upon other models by substantial margins. With different ablation experiments, our work provides significant insights into the complex interplay of text-based and network-propagated signals for stance classification.

REFERENCES

[1] Isabelle Augenstein, Tim Rocktäschel, Andreas Vlachos, and Kalina Bontcheva. 2016. Stance Detection with Bidirectional Conditional Encoding. In *EMNLP*. 876–885.

[2] Avrim Blum and Tom M. Mitchell. 1998. Combining Labeled and Unlabeled Data with Co-Training. In *COLT*, Peter L. Bartlett and Yishay Mansour (Eds.). 92–100.

[3] Jing Chen, Jun Feng, Xia Sun, and Yang Liu. 2020. Co-Training Semi-Supervised Deep Learning for Sentiment Classification of MOOC Forum Posts. *Symmetry* 12, 1 (2020), 8.

[4] Minmin Chen, Kilian Q. Weinberger, and Yixin Chen. 2011. Automatic Feature Decomposition for Single View Co-training. In *ICML*, Lise Getoor and Tobias Scheffer (Eds.). Omnipress, 953–960.

[5] Daejin Choi, Selin Chun, Hyunchul Oh, Jinyoung Han, et al. 2020. Rumor propagation is amplified by echo chambers in social media. *Scientific reports* 10, 1 (2020), 1–10.

[6] Costanza Conforti, Jakob Berndt, Mohammad Taher Pilehvar, Chryssi Giannitsarou, Flavio Toxvaerd, and Nigel Collier. 2020. Will-They-Won't-They: A Very Large Dataset for Stance Detection on Twitter. In *ACL*. 1715–1724.

[7] Kareem Darwish, Peter Stefanov, Michaël Aupetit, and Preslav Nakov. 2020. Unsupervised User Stance Detection on Twitter. *ICWSM* 14, 1 (May 2020), 141–152.

[8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL*. 4171–4186.

[9] Jiachen Du, Ruifeng Xu, Yulan He, and Lin Gui. 2017. Stance Classification with Target-Specific Neural Attention Networks. In *IJCAI*. 3988–3994.

[10] Sebastian Dungs, Ahmet Aker, Norbert Fuhr, and Kalina Bontcheva. 2018. Can Rumour Stance Alone Predict Veracity?. In *Proceedings of the 27th International Conference on Computational Linguistics*. Association for Computational Linguistics, Santa Fe, New Mexico, USA, 3360–3370. https://www.aclweb.org/anthology/C18-1284

[11] Subhabrata Dutta, Dipankar Das, Gunkirat Kaur, Shreyans Mongia, Arpan Mukherjee, and Tanmoy Chakraborty. 2019. Into the Battlefield: Quantifying and Modeling Intra-Community Conflicts in Online Discussion. In *CIKM*. 1271–1280.

[12] Nabeel Gillani, Ann Yuan, Martin Saveski, Soroush Vosoughi, and Deb Roy. 2018. *Me, My Echo Chamber, and I: Introspection on Social Media Polarization*. 823–831.

[13] Miha Grčar, Darko Cherepnalkoski, Igor Mozetič, and Petra Kralj Novak. 2017. Stance and influence of Twitter users regarding the Brexit referendum. *Computational social networks* 4, 1 (2017), 1–25.

[14] Lara Grimminger and Roman Klinger. 2021. Hate Towards the Political Opponent: A Twitter Corpus Study of the 2020 US Elections on the Basis of Offensive Speech and Stance Detection. In *Proceedings of the Eleventh Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis, WASSA@EACL 2021, Online, April 19, 2021*, Orphée De Clercq, Alexandra Balahur, João Sedoc, Valentin Barrière, Shabnam Tafreshi, Sven Buechel, and Véronique Hoste (Eds.). Association for Computational Linguistics, 171–180. https://www.aclweb.org/anthology/2021.wassa-1.18/

[15] Divam Gupta, Kushagra Singh, Soumen Chakrabarti, and Tanmoy Chakraborty. 2019. Multi-task Learning for Target-Dependent Sentiment Classification. In *PAKDD*, Vol. 11439. Springer, 185–197.

[16] Itai Himelboim, Stephen McCreery, and Marc Smith. 2013. Birds of a Feather Tweet Together: Integrating Network and Content Analyses to Examine Cross-Ideology Exposure on Twitter. *Journal of Computer-Mediated Communication* 18, 2 (2013), 154–174.

[17] Long Jiang, Mo Yu, Ming Zhou, Xiaohua Liu, and Tiejun Zhao. 2011. Target-dependent Twitter Sentiment Classification. In *ACL*. 151–160.

[18] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net. https://openreview.net/forum?id=SJU4ayYgl

[19] Svetlana Kiritchenko and Stan Matwin. 2011. Email classification with co-training. In *CASCON*, Joanna W. Ng, Christian Couturier, Marin Litoiu, and Eleni Stroulia (Eds.). 301–312.

[20] Sumeet Kumar and Kathleen Carley. 2019. Tree LSTMs with Convolution Units to Predict Stance and Rumor Veracity in Social Media Conversations. In *ACL*. 5047–5058.

[21] Srijan Kumar, William L. Hamilton, Jure Leskovec, and Dan Jurafsky. 2018. Community Interaction and Conflict on the Web. In *Proceedings of the 2018 World Wide Web Conference* (Lyon, France) *(WWW '18)*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 933–943. https://doi.org/10.1145/3178876.3186141

[22] Mirko Lai, Marcella Tambuscio, Viviana Patti, Giancarlo Ruffo, and Paolo Rosso. 2019. Stance polarity in political debates: A diachronic perspective of network homophily and conversations on Twitter. *Data & Knowledge Engineering* 124 (2019), 101738.

[23] Walid Magdy, Kareem Darwish, Norah Abokhodair, Afshin Rahimi, and Timothy Baldwin. 2016. #ISISisNotIslam or #DeportAllMuslims? Predicting Unspoken

[24] Views. In *WebSci*. 95–106.

[25] Saif Mohammad, Svetlana Kiritchenko, Parinaz Sobhani, Xiaodan Zhu, and Colin Cherry. 2016. A Dataset for Detecting Stance in Tweets. In *LREC*.

[25] Saif M. Mohammad, Parinaz Sobhani, and Svetlana Kiritchenko. 2017. Stance and Sentiment in Tweets. *ACM Trans. Internet Technol.* 17, 3, Article 26 (June 2017), 23 pages.

[26] Jonas Mueller and Aditya Thyagarajan. 2016. Siamese recurrent architectures for learning sentence similarity. In *AAAI*, Vol. 30.

[27] Subhabrata Mukherjee and Ahmed Hassan Awadallah. 2020. Uncertainty-aware Self-training for Few-shot Text Classification. In *NeurIPS*. 1–14.

[28] Peter Stefanov, Kareem Darwish, Atanas Atanasov, and Preslav Nakov. 2020. Predicting the Topical Stance and Political Leaning of Media using Tweets. In *ACL*. 527–537.

[29] Hao Tang, Donghong Ji, Chenliang Li, and Qiji Zhou. 2020. Dependency Graph Enhanced Dual-transformer Structure for Aspect-based Sentiment Classification. In *ACL*. 6578–6588.

[30] Xiaojun Wan. 2011. Bilingual Co-Training for Sentiment Classification of Chinese Product Reviews. *Computational Linguistics* 37, 3 (2011), 587–616.

[31] Bo Wang, Maria Liakata, Arkaitz Zubiaga, and Rob Procter. 2017. TDParse: Multi-target-specific sentiment recognition on Twitter. In *EACL*. 483–493.

[32] Chang Xu, Cécile Paris, Surya Nepal, and Ross Sparks. 2018. Cross-Target Stance Classification with Self-Attention Networks. In *ACL*. 778–783.

[33] Kaizhou Xuan and Rui Xia. 2019. Rumor Stance Classification via Machine Learning with Text, User and Propagation Features. In *2019 International Conference on Data Mining Workshops (ICDMW)*. 560–566. https://doi.org/10.1109/ICDMW.2019.00085

[34] Min Yang, Wenting Tu, Jingxuan Wang, Fei Xu, and Xiaojun Chen. 2017. Attention Based LSTM for Target Dependent Sentiment Classification. *AAAI* 31, 1 (2017).

[35] Li Zeng, Kate Starbird, and Emma Spiro. 2016. #Unconfirmed: Classifying Rumor Stance in Crisis-Related Social Media Messages. *ICWSM* 10, 1 (Mar. 2016).

[36] Dengyong Zhou, Olivier Bousquet, Thomas Navin Lal, Jason Weston, and Bernhard Schölkopf. 2003. Learning with Local and Global Consistency. In *NIPS*, Sebastian Thrun, Lawrence K. Saul, and Bernhard Schölkopf (Eds.). 321–328.

[37] Elena Zotova, Rodrigo Agerri, Manuel Nuñez, and German Rigau. 2020. Multilingual Stance Detection in Tweets: The Catalonia Independence Corpus. In *LREC*. 1368–1375.

Semi-supervised Stance Detection of Tweets
Via Distant Network Supervision
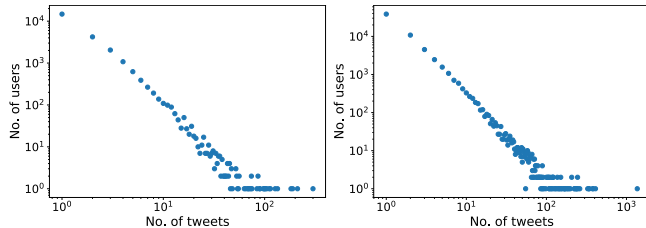
(Appendix / Supplementary Material)

Figure 6: Log-log distribution of tweets posted by users in `StanceUS` (left) and `StanceIN` (right).

A DATASET PREPARATION

To collect US political stances, we first identified a set of 100 users who expressed support towards the Democratic (Dem) or the Republican (Rep) Parties explicitly on their profile information (50 users each). We also kept track of daily trending hashtags (political as well as non-political) and collected tweets that mentioned them. We then created the full social network comprised of the said 100 users and the users that posted using the trending hashtags. We continued this procedure from 1st January, 2020 to 3rd April, 2020. We also collected tweets posted by a user in our collection within this time-frame which was not collected while searching for trending hashtags to ensure continuity of tweeting activity. Finally, we end up with a total of $59,684$ tweets from $24,490$ users, with an average of 2.44 tweets and 41.79 followees per user. We refer to this dataset as **`StanceUS`**.

For the Indian counterpart, we followed the same procedure. However, we focused mostly on the capital city area of New Delhi to ensure most tweets to be written in English. The targeted political bodies for this demographic area were Bharatiya Janta Party (BJP), Indian National Congress (INC), and Aam Admi Party (AAP). Our crawling period ranges from 26th October, 2019 to 2nd March, 2020, resulting in a total of $176,619$ tweets from $63,230$ users, with an average of 2.79 tweets and 22.93 followees per user. We refer to this dataset as **`StanceIN`**. Figure 6 shows the distribution of the number of tweets posted by users, evidently following power-law.

For initial supervised training and final evaluation of SANDS as well as the baselines, we proceed to manually annotate a randomly selected subset of the collected data according to the stance they express. For US-based tweets, we annotate each tweet with one among the following labels: **Pro-Dem**, **Anti-Dem**, **Pro-Rep**, **Anti-Rep**, and, **Other**. In case of India-based tweets, the labels are **Pro-BJP**, **Anti-BJP**, **Pro-INC**, **Anti-INC**, **Pro-AAP**, **Anti-AAP**, and **Other**. We employ three expert annotators for this task (their ages range between 25–35 years). All of them are from a linguistics background and familiar with political events and entities in both demographics. In case of disagreement among the annotators, we select the majority label (if two of them agree), else discard it altogether. After a 5-week long annotation process, we finally end up with $3,822$ and $4,185$ annotated tweets from US and India with inter-annotator agreements 0.84 and 0.78 Cohen's κ, respectively.

The distribution of annotated samples among different classes is shown in Table 1.

To investigate the effects of the size of the labeled training data, we train and test SANDS and all the baselines on three different training sets of size 500, 1000, and 1500.

B TWEET PREPROCESSING

For both `StanceUS` and `StanceIN`, we use the same strategy for cleaning the tweets. The text of each tweet is converted to lowercase. The URLs in the tweet are removed. All URLs are first found using regular expression search and then, replaced by a space. We further, created a list of hashtags used in the tweet and removed any hashtags or mentions that were present in the tweet. Finally, we remove any punctuation or numbers present in the tweet.

In order to create the vocabulary of words, we begin by tokenizing the tweets and creating a set of all the tokens. From this set, all non-ASCII terms are removed, along with the terms that occur only once. For obtaining the vocabulary of hashtags, from the set of all hashtags, the hashtags that occur less than or equal to five times are removed to exclude any rare hashtags. After this, we obtained a vocabulary of size 16166 and hashtag vocabulary of size 2134 for `StanceUS`. The maximum length for the tweets (number of terms in the cleaned tweet) in the `StanceUS` is 59 and the maximum hashtag length is 30. For `StanceIN`, the word vocabulary size is 31939 and the size of hashtag vocabulary obtained is 3480. For the cleaned tweets in `StanceIN`, the maximum tweet length is found to be 61 and the maximum length for the hashtags in a given tweet is 28.

C PARAMETER DETAILS

In both the models (convolutional and bi-LSTM) in our setting, the word embedding layers are initialized with pretrained GloVe word vectors of dimension 200. The hashtag embedding layers were initialized randomly with size 128. We keep all these embeddings trainable throughout the training process.

For the self-attention based module to represent hashtags (see Section 3.2.1), we set the dimensions of the query, key, and value vectors to 128, which then trivially becomes the dimensionality of Z^H as well.

In the three successive convolution-maxpool blocks (see Section 3.2.2), we keep the filter sizes as 128, 64, and 32, respectively, in all the branches of different window sizes. In the bi-LSTM model, we set the hidden size of the LSTM cell to 200 (100 in each direction and concatenated).

Each training iteration in our method consists of a supervised pass followed by a semi-supervised one (as described in Section 3.3). In the supervised pass, we set the dropout probability $p_{\text{dropout}} = 0.1$ (see Section 3.2) with a batch size of 128. Due to the constraints described in Section 3.4, the mini-batches in the semi-supervised pass are of variable sizes and we set the upper bound on the size to
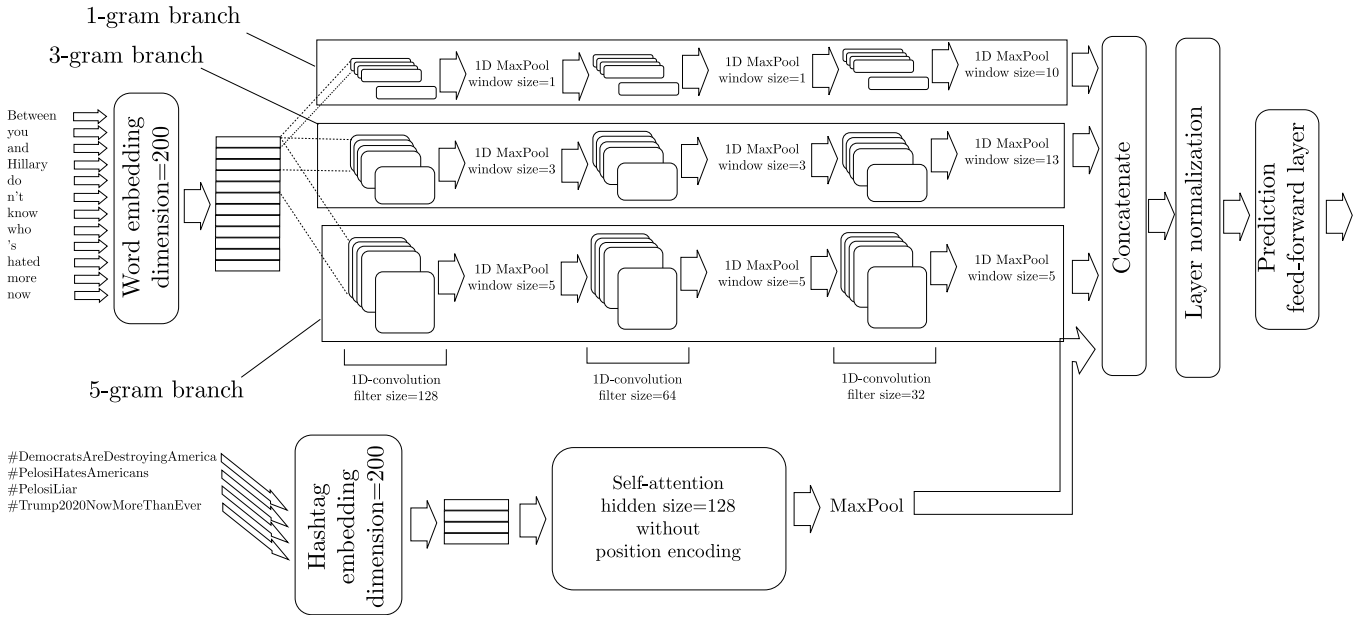
Figure 7: Fully expanded view of the convolutional model

be 512. We set $p_{\text{dropout}} = 0.3$ in this pass. We use Adam optimizer with a learning rate of 10^{-4} for both the models.

As follower-network information for Twitter users is often kept private due to profile settings, sometimes it is not possible to collect the information about all the followees of a given user. To ensure that the pseudo-labels computed from the tweets posted by the followees $f(u_i)$ (see Section 3.3.2) do not get biased due to the small size of followees, we use an empirically set *min_degree* threshold to ignore pseudo-labels for users having followees less than that value. We set *min_degree* = 15 for StanceUS and 20 for StanceIN.