Semantic Oppositeness Assisted Deep Contextual Modeling for Automatic Rumor Detection in Social Networks

Nisansa de Silva, Dejing Dou
Department of Computer and Information Science,
University of Oregon,
Eugene, USA.
{nisansa,dou}@cs.uoregon.edu

Abstract

Social networks face a major challenge in the form of rumors and fake news, due to their intrinsic nature of connecting users to millions of others, and of giving any individual the power to post anything. Given the rapid, widespread dissemination of information in social networks, manually detecting suspicious news is sub-optimal. Thus, research on automatic rumor detection has become a necessity. Previous works in the domain have utilized the reply relations between posts, as well as the semantic similarity between the main post and its context, consisting of replies, in order to obtain state-of-the-art performance. In this work, we demonstrate that semantic oppositeness can improve the performance on the task of rumor detection. We show that semantic oppositeness captures elements of discord, which are not properly covered by previous efforts, which only utilize semantic similarity or reply structure. Our proposed model learns both explicit and implicit relations between the main tweet and its replies, by utilizing both semantic similarity and semantic oppositeness. Both of these employ the self-attention mechanism in neural text modeling, with semantic oppositeness utilizing word-level self-attention, and with semantic similarity utilizing post-level self-attention. We show, with extensive experiments on recent data sets for this problem, that our proposed model achieves state-of-the-art performance. Further, we show that our model is more resistant to the variances in performance introduced by randomness.

1 Introduction

Social media changed the ecosystem of the World Wide Web by making it possible for any individual, regardless of their level of knowledge of web technologies, to create and maintain *profiles* online. At the same time, various social media provided these individuals with means to tap into the information disseminated by others (e.g., Facebook by *adding friends*, Twitter by *following*). By virtue of other mechanisms, such as *Facebook pages* and *Twitter lists*, the reach of each individual was then extended to the range of thousands-to-millions of users. New content, in the form of *posts*, is created on social media sites each passing second.

The rapidity of this post creation is such, that it is possible to claim that social media reflect a near real-time view of the events in the real world (Veyseh et al., 2019). While it was, indeed, beneficial in terms of *volume* of data, to have private individuals be content creators and propagators of information, this created significant issues, from the perspective of *veracity* of the data. This gave rise to a challenge of detecting *fake news* and *rumors* (which, in this study, we refer to as the task of *rumor detection*). The need for rumor detection has come to the forefront, in light of its momentous impacts on political events (Jin et al., 2017) and social (Jin et al., 2014) or economic (Domm, 2013) trends.

Manual intervention on this task would require extensive analysis of and reasoning about various sources of information, resulting in long response times, which are intolerable, given the impact of these rumors, and the rate at which they spread. Thus, *automatic rumor detection*, toward which we contribute in this paper, has become an important area of contemporary research. Cao et al. (2018) define any piece of information, of which the veracity status was questionable at the time of posting, as a rumor. They further claim that a rumor may later be verified to be true or false by other authorized sources. We follow their definition in this work; thus, we define the task of rumor detection as: Given a piece of information from a social network, predict whether the piece of information is a rumor or not using the conversations which were induced by the said piece of information. The initial piece of information could be a tweet or a user

Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics, pages 405–415
April 19 - 23, 2021.

post, and the induced conversation would be the replies from other users (which we use as contextual information). Following the conventions in the literature, in this work, we refer to a main post and its replies as a *thread*.

In this paper, we utilize the semantic oppositeness proposed by (de Silva and Dou, 2019) to improve the rumor detection task, which has so far been restricted to only considering semantic similarity. We further prove that semantic oppositeness is well-suited to be applied to this domain, under the observation that rumor threads are more discordant than those of non-rumors. We further observe that, within rumor threads, false rumor threads continue to be clamorous; while true rumor threads settle into inevitable acquiescence. We claim that semantic oppositeness can help in distinguishing this behavior as well.

We propose word-level self-attention mechanism for the semantic oppositeness to augment the tweet level self-attention mechanism for the semantic similarity. We model the explicit and implicit connections within a thread, using a relevancy matrix. Unlike a regular adjacency matrix, our relevancy matrix recognizes the coherence of each sub-tree of conversation rooted at the main post, while acknowledging that, by definition, for this task, the main tweet must be directly related all the rest of the tweets, regardless of the degrees of separation that may exist between them. We conduct extensive experiments to compare our proposed model with the state-of-the-art studies conducted on the same topic. To the best of our knowledge, this work is the first to utilize semantic oppositeness in rumor detection. In summary, our contributions in this paper include:

- We introduce a novel method for rumor detection, based on both semantic similarity and semantic oppositeness, utilizing the main post and the contextual replies.
- We model the explicit and implicit connections within a thread, using a relevancy matrix, which is then used to balance the impact semantic similarity and semantic oppositeness have on the overall prediction.
- We conduct experiments on recent rumor detection data sets and compare with numerous state-of-the-art baseline models to show that we achieve superior performance.

The remainder of this paper is organized as follows: Section 2 presents related work, and then Section 3 provides a formal definition of the problem, along with our proposed solution. It is followed by Section 4 discussing experiments and results. Finally the Section 5 concludes the paper.

2 Related Work

Semantic oppositeness is the mathematical counterpart of semantic similarity (de Silva and Dou, 2019). While implementations of semantic similarity (Jiang and Conrath, 1997; Wu and Palmer, 1994) are more widely used than those of semantic oppositeness, there are a number of studies which work on deriving or using semantic oppositeness (de Silva et al., 2017; Paradis et al., 1982; Mettinger, 1994; Schimmack, 2001; Rothman and Parker, 2009; de Silva, 2020). However, it is noted that almost all of these studies are reducing oppositeness from a scale to either bipolar scales (Schimmack, 2001) or simple anonymity (Paradis et al., 1982; Jones et al., 2012). The study by de Silva et al. (2017) proves that this reduction is incorrect and proposes an alternative oppositeness function. Their follow-up study, de Silva and Dou (2019) creates a word embedding model for this function. In this study, we use the oppositeness embeddings created by them.

Rumor detection task has been approached on three fronts, according to Cao et al. (2018): feature engineering, propagation-based, and deep learning. In the *feature engineering* approach, posts are transformed into feature representations by hand-designed features and sent to a statistical model to be classified. In addition to textual information, structural evidences (Castillo et al., 2011; Yang et al., 2012) and media content (Gupta et al., 2012) are also utilized. Given that this approach depends heavily on the quality of the hand-designed feature sets, it is neither scalable, nor transferable to other domains. The *propagation-based* approach is built on the assumption that the propagation pattern of a rumor is significantly different to that of a non-rumor. It has been deployed to detect rumors in social networks (Ma et al., 2017). However, this method does not pay any heed to the information in the post content itself. As expected, *deep learning* approach, automatically learns effective features (Ma et al., 2016, 2018; Veyseh et al., 2019). Ma et al. (2016) claim that these discovered features capture the underlying representations of

the posts, and hence, improve the generalization performance, while making it easy to be adapted into a new domain or a social medium for the purpose of rumor detection.Yang et al. (2020) propose a slide window-based system for feature extraction. None of these state-of-the-art work attempt to check rumour veracity akin to attempts by Hamidian and Diab (2019a) and Derczynski et al. (2017). Instead, they attempt to do classification on the already established baseline. Thus, our work also follow the approach of the former rather than the latter. The work by Hamidian and Diab (2019b) does focus on rumor detection and classification. However, they are not using the data sets common to the state-of-the-art work mentioned above to evaluate their approach.

Our work is most related to the rumor detection model on Twitter by means of deep learning to capture contextual information (Veyseh et al., 2019). However, we also derive inspiration from earlier work on the same topic (Ma et al., 2018), which utilized the tree-like structures of the posts, and the work by de Silva and Dou (2019), which introduced the oppositeness embedding model. The early work by Ma et al. (2018) uses Recursive Neural Networks (RvNN) for the construction of the aforementioned tree-like structures of the posts, based on their *tf-idf* representations.

The following work by Veyseh et al. (2019) acknowledges the usefulness of considering the innate similarities between replies, but further claims that only considering the replies along the tree-like structure only exploits the explicit relations between the main posts and their replies, and thus ignores the implicit relations among the posts from different branches based on their semantics. Under this claim, they disregard the tree-like structure entirely. In our work, we preserve the idea of considering semantic similarities to discover the implicit relationships among posts, as proposed by (Veyseh et al., 2019).

However, we augment the model and reintroduce the explicit relationships proposed by Ma et al. (2018) in a balancing of information between implicit and explicit. Further, we note that all these prior works have been solely focused on the similarity between the posts and have ignored the oppositeness metric. To the best of our knowledge, we are the first to utilize oppositeness information in the rumor detection task.

3 Methodology

We use a recent work (Veyseh et al., 2019) on rumor detection as our baseline. Their work, in turn, was heavily influenced by the earlier work on rumor detection in Twitter (Ma et al., 2018). A tweet set I is defined as shown in Equation 1, where R_0 is the initial tweet and $R_1, R_2, \ldots, R_T$ are replies, such that T is the count of replies. Each tweet R_i is a sequence of words $W_1, W_2, ..., W_n$, such that n is the count of words. We tokenize the tweets; and in this work, *tokens* and *words* are used interchangeably. We also define the relevance matrix M, which carries the information of the tree structure of the tweet tree in Equation 2, where $A \star B$ denotes that A and B belong to the same tree in the forest obtained by eliminating the initial tweet. We show the process in Fig 1 as well. Our input is the pair $P = (I, M)$, which differs from (Veyseh et al., 2019), where only I was used as the input. The entire data set is represented by D.

Following the convention of (Veyseh et al., 2019) which is our baseline, we classify each pair (I, M) into four labels: 1) Not a rumor (NR); 2) False Rumor (FR); 3) True Rumor (TR); and 4) Unrecognizable (UR), It should be noted that the distinction between "*False Rumor*" and "*True Rumor*" is drawn from the truthfulness of R_0.

$$I = (R_0, R_1, R_2, \ldots, R_T) \tag{1}$$

$$m_{i,j} = \begin{cases} 1 & \text{if } R_i = R_0 \vee R_j = R_0 \\ 1 & \text{if } R_i \star R_j \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

In simpler terms, we can represent Veyseh et al. (2019) as a trivial *relevance matrix* where all elements are set at 1. The success of Veyseh et al. (2019) over previous state-of-the-art methods attest to the success of using a *relevance matrix* over vanilla *adjacency matrix*. In this work what we do with the above described *relevance matrix* M is to augment the implicit relationship consideration using the high level structure of the explicit relationships, hence bringing in the best-of-both-worlds. In summary, the set of edges in the *relevancy matrix* is a super-set of the set of edges in the *adjacency matrix*. In addition to the edges that were in the *adjacency matrix*, the *relevancy matrix* also has edges that carry implicit connection information. Thus, by definition, the *relevancy matrix* is more

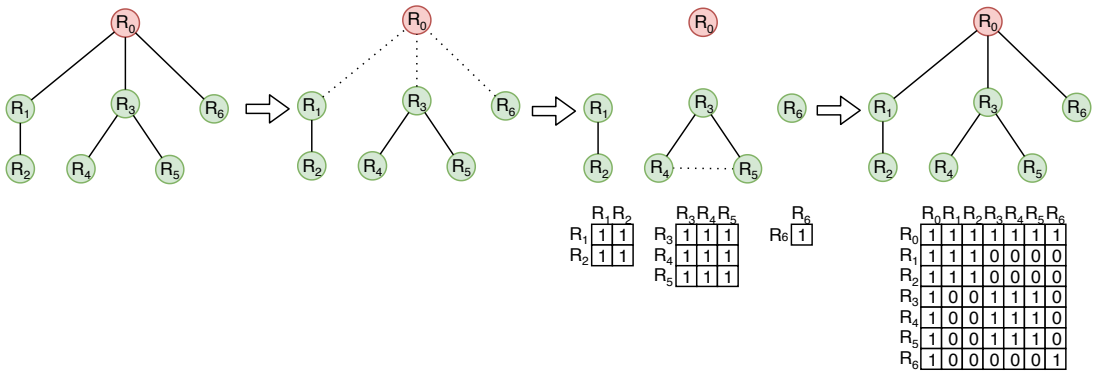


Figure 1: **Relevance matrix building:** 1) Original tweet reply tree; 2) Obtain the forest by temporarily removing the root (main tweet); 3) Consider each tree in the forest to be fully connected graphs, and obtain the relevance matrices; 4) Obtain the full Relevance matrix by putting together the matrices from the previous step and considering the main tweet to be connected to all the other tweets.

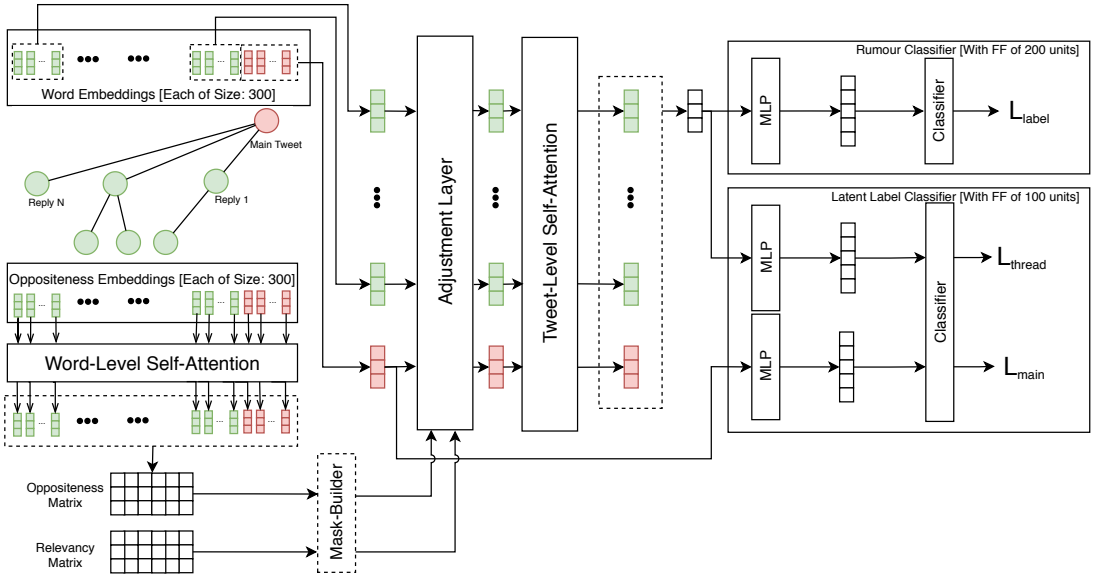


Figure 2: **The Proposed Model:** Red vectors and node represent the main (root) tweet, and green vectors and nodes represent replies. Pooling operations are shown in boxes with dashed lines.

descriptive of the thread compared to the *adjacency matrix*.

3.1 Formal Definition of Tweet Representation

Each tweet will have different number of words n; thus, we pad the short tweets with a special token, until all the tweets have the same word length N as defined by 3.

$$N = \underset{P_i \in D}{\operatorname{argmax}}(n_i) \tag{3}$$

We build the representative oppositeness list O using the oppositeness embeddings created by (de Silva and Dou, 2019) such that, for the i-th tweet R_i, with words $W_{i1}, W_{i2}, ..., W_{iN}$, the oppositeness embedding O_i is created as $o_{i1}, o_{i2}, ..., o_{im}$ where o_{ij} is the embedding of W_{ij}. Note that $m \leq N$ where all tokens might not have corresponding oppositeness embeddings.

Each word in each tweet is then converted to a representative vector by means of a set of pre-trained word embeddings, such that for the i-th tweet R_i, with words $W_{i1}, W_{i2}, ..., W_{iN}$ is converted $e_{i1}, e_{i2}, ..., e_{iN}$. We then apply max-pooling operation over the word embeddings along each dimension, resulting in a representative vector h_i coupled to R_i, as shown in Equation 4. At this point, note that the tweet set I of each pair P, which used to be $I = (R_0, R_1, R_2, \ldots, R_T)$, has been replaced by $I = (h_0, h_1, h_2, \ldots, h_T)$. It is this new representation which is passed to the fol-

lowing steps.

$$h_i = Elementwise\_Max(e_{i1}, e_{i2}, ..., e_{iN}) \quad (4)$$

3.2 Similarity-Based Contextualization

As discussed earlier, the Twitter data is organized as a tree rooted at the main tweet R_0 in each instance. The earlier work by Ma et al. (2018) proved that, in rumor detection, it is helpful to capture these relations among the main tweet and the replies. The subsequent work by Veyseh et al. (2019) noted that only considering explicit reply relation between the main tweet and other tweets neglects the implicit relations among the tweets, arising from their semantic similarities (i.e., by the virtue of discussing the same topic, tweets in two separate branches may carry mutually useful information). Following this hypothesis, they exploited such implicit semantic relations for the purpose of improving the performance of the rumor detection task. However, in doing so, they abandoned the information garnered from the tree structure. In this work we propose to continue to use the implicit information, but to augment it with the information derived from the tree structure.

We follow the self-attention mechanism of (Veyseh et al., 2019), which was inspired by the transformer architecture in (Vaswani et al., 2017), to learn the pairwise similarities among tweets for capturing the semantic relations between the tweets. The process starts with calculating the key (k_i) and query (q_i) vectors for each tweet, based on its representation h_i, as shown in Equation 5. (W and b follow the traditional notation of weights and bias).

$$k_i = W_k * h_i + b_k \qquad q_i = W_q * h_i + b_q \quad (5)$$

With the key and query vectors, we calculate the similarity a_{ij} between i-th and j-th tweets, using the dot product as shown in Equation 6, where γ is a normalization factor.

$$a_{i,j} = k_i \cdot q_j / \gamma \quad (6)$$

3.3 Oppositeness-Based Contextualization

Unlike in the case of similarity vectors, which were reduced to a single dimension at this point, the oppositeness representations are still at two dimensions. Thus the self-attention of oppositeness between tweets is handled at a word level, rather than at the sentence level. We build key ($k_i^{'}$) and query ($q_i^{'}$) vectors for each word based on its representation o_i, as shown in Equation 7. (W and b follow the traditional notation of weights and bias).

$$k_i^{'} = W_k * o_i + b_k \qquad q_i^{'} = W_q * o_i + b_q \quad (7)$$

Since the oppositeness embedding of (de Silva and Dou, 2019) is based on Euclidean distance, with the key and query vectors, we calculate the oppositeness op_{i_x,j_y} between x-th word of i-th tweet and y-th word of j-th tweet using the Euclidean distance, as shown in Equation 8 where $k_{i_x}^{'}$ is the key vector for x-th word of i-th tweet, $q_{j_y}^{'}$ is the query vector for y-th word of j-th tweet, and Euclidean distance $d(,)$ is calculated across the size of the oppositeness embedding.

$$op_{i_x,j_y} = d(k_{i_x}^{'}, q_{j_y}^{'}) \quad (8)$$

To obtain the abstract tweet-level oppositeness, we apply element-wise average-pooling on the $OP_{i,j}$ matrix, as shown in Equation 9, to create the oppositeness matrix $O^{"}$, where EA is $Elementwise\_Average$ operation, δ is the oppositeness embedding count of the i-th tweet, and ϱ is the oppositeness embedding count of the j-th tweet. Note that the dimensions of the oppositeness matrix $O^{"}$ is the same as the relevance matrix M.

$$o_{i,j}^{"} = EA\left(\begin{bmatrix} op_{i_0,j_0} & op_{i_1,j_0} & \dots & op_{i_\delta,j_0} \\ op_{i_0,j_1} & op_{i_1,j_1} & \dots & op_{i_\delta,j_1} \\ \dots & \dots & \dots & \dots \\ op_{i_0,j_\varrho} & op_{i_1,j_\varrho} & \dots & op_{i_\delta,j_\varrho} \end{bmatrix}\right) \quad (9)$$

Next we create the oppositeness mask Ω by average-pooling $O^{"}$ along rows and columns, as shown in Equation 10, where the definition of EA, is the same as Equation 9 and similar to Equation 3, n_i and n_j are natural lengths of the i-th and j-th tweets respectively.

$$\begin{aligned} \omega_{i,j} = 1 - EA(o_{i,0}^{"}, o_{i,1}^{"}, ..., o_{i,n_j}^{"}) \\ -EA(o_{0,j}^{"}, o_{1,j}^{"}, ..., o_{n_i,j}^{"}) \end{aligned} \quad (10)$$

3.4 Deriving Overall Thread Representations

Similar to the oppositeness mask Ω, we create the relevance mask Ψ by sum-pooling M along rows and columns, as shown in Equation 11, where ES, is $Elementwise\_Sum$ operation, and similar to

Equation 3, n_i and n_j are natural lengths of the i-th and j-th tweets respectively.

$$\psi_{i,j} = ES(m_{i,0}, m_{i,1}, ..., m_{i,n_j}) + ES(m_{0,j}, m_{1,j}, ..., m_{n_i,j}) \quad (11)$$

At this point we diverge from (Veyseh et al., 2019) in two ways and utilize the related relevance mask M as a weighting mechanism, with proportion constant α (where $0 < \alpha < 1$), as well as the oppositeness mask Ω, to obtain augmented attention $a^{'}_{i,j}$ as shown in Equation 12.

$$a^{'}_{i,j} = a_{i,j}\omega_{i,j}\left[(\psi_{i,j} - \alpha)^2 + \alpha\psi_{i,j}\right] \quad (12)$$

We utilize the augmented similarity values $a^{'}_{i,j}$ for each tweet pair in the thread to compute abstract representations for the tweets based on the weighted sums, as shown in Equation 13.

$$h'_i = \Sigma_j a^{'}_{i,j} * h_j \quad (13)$$

Next, we apply the max-pooling operation over the processed tweet representation vectors h'_i to obtain the overall representation vector h' for the input pair P.

$$h' = Elementwise\_Max(h'_0, h'_1, h'_2, ..., h'_T) \quad (14)$$

Finally, the result is sent through a 2-layer feed-forward neural network capped with a softmax layer, with the objective of producing the probability distribution $P(y|R_0, R_1, R_2, \ldots, R_T; \theta)$ over the four possible labels, where θ is the model parameter. On this, we optimize the negative log-likelihood function, in order to train the model, as shown in Equation 15, where y^* is the expected (correct) label for I.

$$L_{label} = -\log P(y^*|R_0, R_1, R_2, \ldots, R_T; \theta) \quad (15)$$

3.5 Main Tweet Information Preservation

The Veyseh et al. (2019) study noted that the model by Ma et al. (2018) treats all tweets equally. This was deemed undesirable, given that the main tweet of each thread incites the conversation, and thus, arguably, carries the most important content in the conversation, which should be emphasized, to produce good performance. To achieve this end, it was proposed to bring forward the information in the main tweet independently of and separately from that of the collective twitter thread, in order to provide a check. We, in this work, also provide this sanctity check, to enhance the obtained results.

The basic idea is that, by virtue of definition, if a main tweet is a rumor (or not), unique trait and information pertaining to that class should be in the main tweet itself. Thus, the latent label (L_{thread}) obtained by processing the thread representation h' above should be the same as a potential latent label (L_{main}) obtained by processing the representation of the main tweet h_0. To calculate L_{main}, we use a 2-layer feed-forward neural network with a softmax layer in the end, where it assigns the latent labels drawn from K possible latent labels. Next, we use another 2-layer feed-forward neural network with a softmax layer in the end, assigning the same K number of possible latent labels as shown in the negative log-likelihood function to match it with the thread.

$$L_{main} = argmax_L P(L|R_0) \quad (16)$$

$$L_{thread} = -\log P'(L_{main}|R_0, \ldots, R_T) \quad (17)$$

Finally, the loss function to train the entire model is defined as in Equation 18, where the L_{label} is obtained from Equation 15, and β is a hyper-parameter which controls the contribution of the main tweet information preservation loss to final loss.

$$Loss = L_{label} + \beta L_{thread} \quad (18)$$

4 Experiments

We use the *Twitter 15* and *Twitter 16* data sets introduced by Ma et al. (2017) for the task of rumor detection. Some statistics of the data sets as given by Ma et al. (2017) are shown in Table 1. We use Glove (Pennington et al., 2014) embedding to initialize the word vectors and oppositeness embedding (de Silva and Dou, 2019) to initialize the oppositeness embeddings. Both embedding vectors are of size 300. Key and query vectors in Equations 5 and Equations 7 employ 300 hidden units. The rumor classifier feed-forward network has two layers of 200 hidden units. The feed-forward layer in the main tweet information preservation component has two layers, each with 100 hidden units,

and it maps to three latent labels. The proportion constant α, which balances the explicit and implicit information, is set at 0.1. The loss function uses a trade-off parameter of $\beta = 1$, with an initial learning rate of 0.3 on the *Adagrad* optimizer. For the purpose of fair results comparison, we follow the convention of using 5-fold cross validation procedure to tune the parameters (such as node and layer counts) set by Ma et al. (2018).

Statistic	Twitter15	Twitter16
Number of NR	374	205
Number of FR	370	205
Number of TR	372	205
Number of UR	374	203
Avg. Num. of Posts/Tree	223	251
Max Num. of Posts/Tree	1,768	2,765
Min Num. of Posts/Tree	55	81

Table 1: Statistics of the Data Sets.

4.1 Comparison to the State-of-the-Art Models

We compare the proposed model against the state-of-the-art models on the same data sets. The performance is compared by means of overall accuracy and F1 score per class. We observe that there are two types of models against which we compare. The first type are the *feature-based models*, which used feature engineering to extract features for Decision Trees (Zhao et al., 2015; Castillo et al., 2011), Random Forest (Kwon et al., 2013), and SVM (Ma et al., 2015; Wu et al., 2015; Ma et al., 2017). The second type of models are *deep learning models*, which used Recurrent Neural Networks or Recursive Neural Networks to learn features for rumor detection. We compare our model to GRU-RNN proposed by Ma et al. (2016), BU-RvNN and TD-RvNN proposed by Ma et al. (2018), and Semantic Graph proposed by Veyseh et al. (2019). Results for Twitter 15 and Twitter 16 are shown in Tables 2 and 3, respectively.

It is evident from these tables that, in the rumor detection task, the deep learning models outperform feature-based models, proving that automatically learning effective features from data is superior to hand-crafting features. We also note that the *Semantic Oppositeness Graph*, along with the Semantic Graph, and other RvNN models with GRU-RNN, generally do well, which attests to the utility of structural information, be it in the form of reply structure or be it in the form of semantic relations, in helping to improve performance. We further notice that Veyseh et al. (2019) which uses implicit information, outperforms TD-RvNN (Ma et al., 2018), which only uses explicit information. *Semantic Oppositeness Graph*, which uses explicit information, implicit information, and semantic oppositeness outperforms all the other models in accuracy, while outperforming all the other models in three out of four classes, in terms of F1 Score. The one class in which *Semantic Oppositeness Graph* loses out to Veyseh et al. (2019) is in the case of the *Unrecognizable* (UR) class. We argue that this is not an issue, given that the unrecognizable class consists of tweets which were too ambiguous for human annotators to tag as one of: not a rumor (NR), false rumor (FR), or true rumor (TR). We assert that Tables 2 and 3 clearly demonstrate the effectiveness of the proposed *Semantic Oppositeness Graph* method in the task of rumor detection.

4.2 Model Stability Analysis

While comparing our system with Veyseh et al. (2019), which we use as our main baseline, we noticed that their system has a high variance in results, depending on the random weight initialization. This was impactful in such a way that in some random weight initializations, the accuracy of their system could fall as low as 24% from the reported high 70% results in their paper. Given that we use their system as our baseline and the basis for our model, we decided to do a stability analysis between their system and ours. For this purpose, we created 100 random seeds and trained four models with each seed, resulting in a total of 400 models. The models were: 1) Veyseh et al. (2019) on *twitter 15*, 2) Veyseh et al. (2019) on *twitter 16*, 3) *Semantic Oppositeness Graph* on *twitter 15*, 4) *Semantic Oppositeness Graph* on *twitter 16*. Then we normalized the results of the Veyseh et al. (2019) models to the values reported in their paper (also shown in the relevant row on Tables 2 and 3). Each result is reported in the format of (μ, σ) for the 5 fold cross-validation to explore how random weight initialization affects the two models.

From the results in Tables 4 and 5, it is evident that our *Semantic Oppositeness Graph* has higher mean values for accuracy, not a rumor (NR), false rumor (FR), and true rumor (TR), while having comparably reasonable values for Unrecognizable (UR) class. But more interesting are the standard

Model	Accuracy	F1 NR	F1 FR	F1 TR	F1 UR
DTR (Zhao et al., 2015)	0.409	0.501	0.311	0.364	0.473
DTC (Castillo et al., 2011)	0.454	0.733	0.355	0.317	0.415
RFC (Kwon et al., 2013)	0.565	0.810	0.422	0.401	0.543
SVM-TS (Ma et al., 2015)	0.544	0.796	0.472	0.404	0.483
SVM-BOW (Ma et al., 2018)	0.548	0.564	0.524	0.582	0.512
SVM-HK (Wu et al., 2015)	0.493	0.650	0.439	0.342	0.336
SVM-TK (Ma et al., 2017)	0.667	0.619	0.669	0.772	0.645
GRU-RNN (Ma et al., 2016)	0.641	0.684	0.634	0.688	0.571
BU-RvNN (Ma et al., 2018)	0.708	0.695	0.728	0.759	0.653
TD-RvNN (Ma et al., 2018)	0.723	0.682	0.758	0.821	0.654
Semantic Graph (Veyseh et al., 2019)	0.770	0.814	0.764	0.775	**0.743**
Semantic Oppositeness Graph (SOG)	**0.796**	**0.825**	**0.820**	**0.814**	0.742

Table 2: Model Performance on Twitter 15.

Model	Accuracy	F1 NR	F1 FR	F1 TR	F1 UR
DTR (Zhao et al., 2015)	0.414	0.394	0.273	0.630	0.344
DTC (Castillo et al., 2011)	0.465	0.643	0.393	0.419	0.403
RFC (Kwon et al., 2013)	0.585	0.752	0.415	0.547	0.563
SVM-TS (Ma et al., 2015)	0.574	0.755	0.420	0.571	0.526
SVM-BOW (Ma et al., 2018)	0.585	0.553	0.655	0.582	0.578
SVM-HK (Wu et al., 2015)	0.511	0.648	0.434	0.473	0.451
SVM-TK (Ma et al., 2017)	0.662	0.643	0.623	0.783	0.655
GRU-RNN (Ma et al., 2016)	0.633	0.617	0.715	0.577	0.527
BU-RvNN (Ma et al., 2018)	0.718	0.723	0.712	0.779	0.659
TD-RvNN (Ma et al., 2018)	0.737	0.662	0.743	0.835	0.708
Semantic Graph (Veyseh et al., 2019)	0.768	0.825	0.751	0.768	**0.789**
Semantic Oppositeness Graph (SOG)	**0.826**	**0.843**	**0.843**	**0.878**	0.774

Table 3: Model Performance on Twitter 16.

Model	Accuracy	F1 NR	F1 FR	F1 TR	F1 UR
Veyseh et al. (2019)	(0.770,0.138)	(0.814,0.133)	(0.764,0.198)	(0.775,0.118)	(0.743,0.129)
SOG (This work)	(0.796,0.089)	(0.825,0.080)	(0.820,0.109)	(0.814,0.093)	(0.742,0.100)

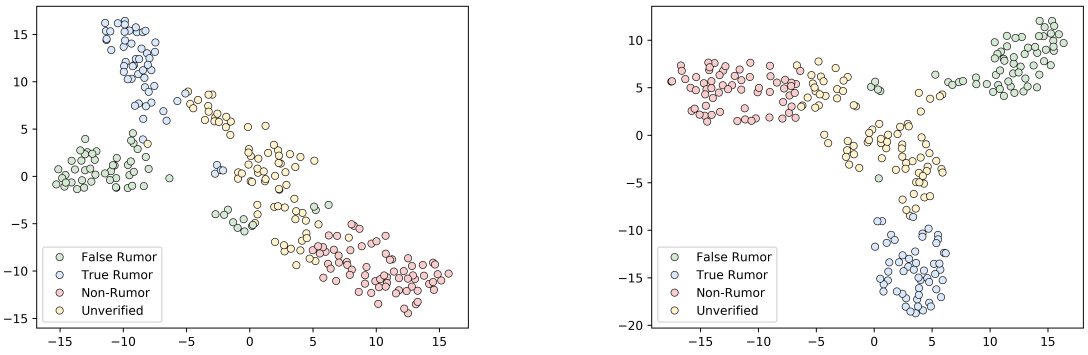
Table 4: Model Variance Performance on Twitter 15.

Model	Accuracy	F1 NR	F1 FR	F1 TR	F1 UR
Veyseh et al. (2019)	(0.768,0.103)	(0.825,0.226)	(0.751,0.103)	(0.768,0.096)	(0.789,0.184)
SOG (This work)	(0.826,0.082)	(0.843,0.153)	(0.843,0.091)	(0.878,0.074)	(0.774,0.114)

Table 5: Model Variance Performance on Twitter 16.

deviation values. It is evident that in all cases, our model has smaller standard deviation values than that of Veyseh et al. (2019). This is proof that our system is comparatively more stable in the face of random weight initialization. We argue that this stability comes from the introduction of the oppositeness component, which augments the decision-making process with the oppositeness information, as a counterpart for the already existing similarity information, preventing the predictions from having a swinging bias.

For a demonstration, consider the subset of three words *increase*, *decrease*, and *expand* from the example given by de Silva and Dou (2019). If the main tweet (R_0) were to say "*A will increase B*", R_1 replied with "*A will decrease B*", and R_2 replied



(a) Without the Oppositeness Component

(b) With the Oppositeness Component

Figure 3: t-SNE diagrams for thread representations.

with "*A will expand B*", then the purely semantic similarity based model will position R_0 and R_1 closer than R_0 and R_2, given that the word contexts in which *increase* and *decrease* are found are more similar than the word contexts in which *increase* and *expand* are found. This would result in the neural network having to learn the opposite semantics between *increase* and *expand* by itself, during the training, making it more vulnerable to issues of bad initial weight selection. This, in turn, will result in greater variance among the trained models as is the case of Veyseh et al. (2019). However, a system with an oppositeness component will already have the opposite semantics between *increase* and *decrease*, as well as *increase* and *expand* already calculated. Thus, such a system would have pre-knowledge that the word pair *increase* and *decrease*, despite being used in more common contexts, is more semantically opposite than the word pair *increase* and *expand*, which is used in less common contexts. Hence the neural network does not have to learn that information from scratch during the training, resulting in it being less vulnerable to issues of bad initial weight selection. Analogously, this, in turn, will result in lesser variance among the trained models; hence, explaining the better stability demonstrated by *Semantic Oppositeness Graph* in comparison to Veyseh et al. (2019) in Tables 4 and 5.

4.3 Impact of the Oppositeness Component

Finally, to emphasize the effect the oppositeness component has on the model, we draw the t-SNE diagrams for the final representations of the threads. Figure 3a shows the data points clustering when the model is trained without the oppositeness component, and Fig. 3b shows the data points clustering when the model is trained with the oppositeness component. Note that all other variables, including the seed for the weight initializer, are the same in the two models. These diagrams prove that the oppositeness component helps improve the separability of the classes. Specially note how the *False Rumor* and *True Rumor* classes are now more clearly separated. We postulate that this derives from the fact that the oppositeness component would help in distinguishing the continuous discord happening in a *False Rumor* thread from the subsequent general agreement in a *True Rumor* thread.

5 Conclusion

Rumors and fake news are a significant problem in social networks, due to their intrinsic nature of connecting users to millions of others and giving any individual the power to post anything. We introduced a novel method for rumor detection, based on semantic oppositeness, in this paper. We demonstrated the effectiveness of our method using data sets from Twitter. Compared to previous work, which only used explicit structures in the reply relations or semantic similarity, our model learns both explicit and implicit relations between a main tweet and its replies, by utilizing both semantic similarity and semantic oppositeness. We proved, with extensive experiments, that our proposed model achieves state-of-the-art performance, while being more resistant to the variances in performance introduced by randomness.

References

Juan Cao, Junbo Guo, Xirong Li, Zhiwei Jin, Han Guo, and Jintao Li. 2018. Automatic Rumor Detection on Microblogs: A Survey. *arXiv preprint arXiv:1807.03505*.

Carlos Castillo, Marcelo Mendoza, and Barbara Poblete. 2011. Information Credibility on Twitter. In *WWW*, pages 675–684. ACM.

Leon Derczynski, Kalina Bontcheva, Maria Liakata, Rob Procter, Geraldine Wong Sak Hoi, and Arkaitz Zubiaga. 2017. Semeval-2017 task 8: Rumoureval: Determining rumour veracity and support for rumours. *arXiv preprint arXiv:1704.05972*.

Patti Domm. 2013. False Rumor of Explosion at White House Causes Stocks to Briefly Plunge; AP Confirms Its Twitter Feed Was Hacked. *https://www.cnbc.com/id/100646197*.

Manish Gupta, Peixiang Zhao, and Jiawei Han. 2012. Evaluating event credibility on twitter. In *SDM*, pages 153–164.

Sardar Hamidian and Mona Diab. 2019a. Gwu nlp at semeval-2019 task 7: Hybrid pipeline for rumour veracity and stance classification on social media. In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 1115–1119.

Sardar Hamidian and Mona T Diab. 2019b. Rumor detection and classification for twitter data. *arXiv preprint arXiv:1912.08926*.

Jay J Jiang and David W Conrath. 1997. Semantic similarity based on corpus statistics and lexical taxonomy. In *COLING*, pages 19–33.

Z. Jin, J. Cao, Y. Jiang, and Y. Zhang. 2014. News credibility evaluation on microblog with a hierarchical propagation model. In *ICDM*, pages 230–239.

Zhiwei Jin, Juan Cao, Han Guo, Yongdong Zhang, Yu Wang, and Jiebo Luo. 2017. Detection and analysis of 2016 US presidential election related rumors on twitter. In *SBP-BRiMS*, pages 14–24.

Steven Jones, M Lynne Murphy, Carita Paradis, and Caroline Willners. 2012. *Antonyms in English: Construals, constructions and canonicity*. Cambridge University Press.

Sejeong Kwon, Meeyoung Cha, Kyomin Jung, Wei Chen, and Yajun Wang. 2013. Prominent features of rumor propagation in online social media. In *ICDM*, pages 1103–1108. IEEE.

Jing Ma, Wei Gao, Prasenjit Mitra, Sejeong Kwon, Bernard J. Jansen, Kam-Fai Wong, and Meeyoung Cha. 2016. Detecting rumors from microblogs with recurrent neural networks. In *IJCAI*, pages 3818–3824.

Jing Ma, Wei Gao, Zhongyu Wei, Yueming Lu, and Kam-Fai Wong. 2015. Detect rumors using time series of social context information on microblogging websites. In *CIKM*, pages 1751–1754. ACM.

Jing Ma, Wei Gao, and Kam-Fai Wong. 2017. Detect rumors in microblog posts using propagation structure via kernel learning. In *ACL*, pages 708–717.

Jing Ma, Wei Gao, and Kam-Fai Wong. 2018. Rumor detection on twitter with tree-structured recursive neural networks. In *ACL*, pages 1980–1989.

Arthur Mettinger. 1994. *Aspects of semantic opposition in English*. Oxford University Press.

Michel Paradis, Marie-Claire Goldblum, and Raouf Abidi. 1982. Alternate antagonism with paradoxical translation behavior in two bilingual aphasic patients. *Brain and language*, 15(1):55–69.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, pages 1532–1543.

Lori Rothman and M Parker. 2009. Just-About-Right (JAR) Scales. *West Conshohocken, PA: ASTM International*.

Ulrich Schimmack. 2001. Pleasure, displeasure, and mixed feelings: Are semantic opposites mutually exclusive? *Cognition & Emotion*, 15(1):81–97.

Nisansa de Silva. 2020. *Semantic Oppositeness for Inconsistency and Disagreement Detection in Natural Language*. Phd dissertation, College of Arts and Sciences, University of Oregon. Available at `https://www.cs.uoregon.edu/Reports/PHD-202012-deSilva.pdf`.

Nisansa de Silva and Dejing Dou. 2019. Semantic oppositeness embedding using an autoencoder-based learning model. In *Database and Expert Systems Applications*, pages 159–174.

Nisansa de Silva, Dejing Dou, and Jingshan Huang. 2017. Discovering inconsistencies in PubMed abstracts through ontology-based information extraction. In *Proceedings of the 8th ACM International Conference on Bioinformatics, Computational Biology, and Health Informatics*, pages 362–371. ACM.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*, pages 5998–6008.

Amir Pouran Ben Veyseh, My T. Thai, Thien Huu Nguyen, and Dejing Dou. 2019. Rumor detection in social networks via deep contextual modeling. In *ASONAM*.

Ke Wu, Song Yang, and Kenny Q Zhu. 2015. False rumors detection on sina weibo by propagation structures. In *ICDE*, pages 651–662.

Z. Wu and M. Palmer. 1994. Verbs semantics and lexical selection. In *ACL*, pages 133–138.

Fan Yang, Yang Liu, Xiaohui Yu, and Min Yang. 2012. Automatic detection of rumor on sina weibo. In *ACM SIGKDD Workshop on Mining Data Semantics*, pages 1–7.

Haolin Yang, Zhiwei Xu, Limin Liu, Jie Tian, and Yujun Zhang. 2020. Adaptive slide window-based feature cognition for deceptive information identification. *IEEE Access*, 8:134311–134323.

Zhe Zhao, Paul Resnick, and Qiaozhu Mei. 2015. Enquiring minds: Early detection of rumors in social media from enquiry posts. In *WWW*, pages 1395–1405.