Fine-Tune Longformer for Jointly Predicting Rumor Stance and Veracity

Anant Khandelwal
anant.iitd.2085@gmail.com
Senior Data Scientist, [24]7.ai
Bangalore, India

ABSTRACT
Increased usage of social media caused the popularity of news and events which are not even verified, resulting in spread of rumors all over the web. Due to widely available social media platforms and increased usage caused the data to be available in huge amounts. The manual methods to process such large data is costly and time-taking, so there has been an increased attention to process and verify such content automatically for the presence of rumors. A lot of research studies reveal that to identify the stances of posts in the discussion thread of such events and news is an important preceding step before identify the rumor veracity. In this paper, we propose a multi-task learning framework for jointly predicting rumor stance and veracity on the dataset released at SemEval 2019 RumorEval: Determining rumor veracity and support for rumors (SemEval 2019 Task 7), which includes social media rumors stem from a variety of breaking news stories from Reddit as well as Twitter. Our framework consists of two parts: a) The bottom part of our framework classifies the stance for each post in the conversation thread discussing a rumor via modelling the multi-turn conversation and make each post aware of its neighboring posts. b) The upper part predicts the rumor veracity of the conversation thread with stance evolution obtained from the bottom part. Experimental results on SemEval 2019 Task 7 dataset show that our method outperforms previous methods on both rumor stance classification and veracity prediction.

KEYWORDS
Rumor Stance, Rumor Veracity, Longformer, Roberta, Transformer, Multi-turn Conversation, Model Averaging

1 INTRODUCTION
Social media platforms like Twitter, Reddit etc generates large amount of data continuously[15, 26, 59]. Nowadays, any novel breaking news appear first on these platforms[23]. Wide usage of social media platforms results in rapid spread and propagation of unverified content i.e. rumor [4, 23, 24, 29, 40, 59, 61–63]. The definition of rumor is "a claim whose truthfulness is in doubt and has no clear source, even if its ideological or partisan origins and intents are clear" [16]. Rumors bring about harmful effects like spreading fear or even euphoria, cause people to make wrong judgement, cause damages to political events, economy and social stability[45]. The massive increase of the social media data rendered the manual methods to debunk rumor, difficult and costly. Machine Learning and Deep Learning based methods to identify such phenomena have attracted more attention to the research community in recent years. Rumor resolution pipeline contain several components such as rumor detection, rumor tracking and stance classification leading to the final outcome of determining the veracity of a rumour [61]. In this paper we concentrate on task A and B released in SemEval 2019 Task 7 (RumorEval 2019) i.e. Stance classification and Veracity prediction respectively[13].

The fine-grained definition for stance classification and veracity prediction is provided by the organizers of RumorEval 2019[13]. The detailed description for each of the two tasks is described as follows:

- **Sub-Task A**: Given a conversation thread discussing the rumor is classified as True, False and Unverified.

This task released the dataset consists of conversations threads from Twitter and Reddit. Reddit threads tend to be longer and more diverse, causing posts in discussion threads to be loosely connected to source post, making the task more challenging[13].

A lot of the systems submitted to RumorEval 2019[13] have used an ensemble model for the task. For example, the best performing system in Subtask B (eventAI)[25], which implemented an ensemble of classifiers (SVM,RF,LR) with features obtained from LSTM attention network and other range of features characterizing posts and conversation. The second best performing system in subtask A (BUT-FIT)[11] uses an ensemble of BERT[9] models (with pre-training) for different parameter settings. Best performing system in task A (BLCU-NLP)[56] and the third best (CLEARumor)[5] also uses pre-trained representation with OpenAI GPT[44] and ELMO[38] respectively. Most of the systems uses single post or pair of posts(source & response) as their input, BLCU-NLP[56] uses the complete conversation thread starting from source post with subsequent replies in the temporal order. They also augment the
training data with more conversations from external public datasets to increase the generalizability of the model. Most of the above-discussed systems either show high performance on (a) Sub-Task A or (b) Sub-Task B (given in the RumorEval 2019). This may be due to the different text style, level of complexities and lengths of conversations obtained from Twitter and Reddit.

Apart from the systems submitted to RumorEval 2019, there are other studies which have observed that stance classification towards rumors is viewed as an important preceding step of rumor veracity prediction [12, 20, 27, 30, 42, 43, 49, 60], especially in the context of Twitter conversations[62]. The approach proposed in[49] is based on the fact that temporal dynamics of stances indicate rumor veracity. They also illustrated the stance distribution of tweets for each category of rumor veracity. It has been observed that there can be conversations which starts with supporting stance but as the stance evolve with the conversation, the deny stance indicates the false rumor. Similar fact is observed in case study (see Figure 2). Based on this observation, we propose to use the post-level(sentence-level) encoder to learn the temporal dynamics of stance evolution for effective veracity prediction(see Section 4.4).

Further, most of the top performing methods[13] at RumorEval 2019 only able to tackle stance classification or veracity prediction separately but not both, which is sub-optimal and limits the generalization of models. But as we have observed previously, these two tasks are inter-related because the stance evolution can provide indicative clues to facilitate veracity prediction. Thus, these two tasks can be jointly learned to make better use of their interrelation.

Based on the above context we can divide the problem into following sub-problems a) to leverage the stance evolution for effective veracity prediction b) to identify the stance of current post, given the neighbouring posts in the conversation thread c) to extract the features for each post in conversation thread and also peculiar to tree-structured conversation, which can provide extra clues for determining stance and veracity like user features (verified user, profile pic present, previous history etc.), structural features(like retweet count, number of hash tags, number of question marks, url count etc) content features (like false synonym, false antonym etc) and psycho-linguistic features ( emotion feature using Emolex[32, 39], Emotion Sentiment[10, 41], post-depth in conversation tree, Speech-act features like order, accept etc). Our approach is based on four main ideas:

- **Multi-turn Conversational Modelling:** The goal is to learn the stance evolution during the conversation to determine the rumor veracity accurately. Also, the stance of a post depends on the neighbouring post. So we choose the variant of Transformer[47] called Longformer[6] (trained from RoBERTa[28] weights with maximum position embeddings upto 4096) as a base model which allows us to configure the window of self-attention while modelling the whole conversation at one go. With sliding window based attention mechanism it can capture signals from neighbouring post, and due to stacked layers it has large receptive field like in the case of stacked CNN’s[55]. Additionally we have used, various sentence encoders(like LSTM[18], Transformer[47] etc.) to establish the inter-sentence relation between posts to learn the fine-tuned representation specific to rumor stance and veracity.
- **Exploiting word-level, post-level and psycho-linguistic features:** The goal is to minimize the direct dependencies on in-depth grammatical structure of conversations from different social media. We have extracted several stylistic and structural features characterizing Twitter or Reddit language. In addition, we utilize conversational-based features to capture the tree-structure of dataset. We have also used affective and emotion based feature by extracting information from several resources like LIWC[37], EmoSenticNet[10, 41], Emolex[32, 39] and ANEW[35]. Additionally, speech-act features from[53] having 229 verbs across 37 categories. We use the term “NLP Features” to represent them in the entire paper.
- **Jointly Learning the Rumor Stance and Veracity Classification:** Since the stance evolution across the conversation provide indicative clues to predict the rumor veracity, we have learned the two tasks jointly to establish the inter-relationship between them.
- **Ensembling:** We have trained different models by varying the type of sentence encoder and learning rate for each configuration and save them resulting in a pool of models. To increase the F1-measure and reduce over-fitting we have used the Top-$N$ fusion strategy used in[11] to select the best models from the pool of saved models.

The intuition behind the “NLP features” are the following:

- **Structural Features:** Like in [22, 25, 36], these are extracted to capture the data characteristics like average word length, ratio of capital words. And, count of chars, words, urls, hash tags, question mark, periods etc. And, boolean feature indicating presence of exclamation mark, negative words, media content like images video etc, question mark, and additional to previous work also extracted 37 pos tags 1. Further, boolean features include flag to indicate post is the source post. Additionally, feature indicating presence of rumor words like gossip, hoax etc. or words indicating doubt in certainty of assertion like unconfirmed.
- **Content Features:** We have extracted the count of false synonyms, antonyms and question word for each post and concatenated with that of source post every time unlike[56] which extracted these only for one post at a time. Also, leveraged external resources[3] to identify presence of cue[3] and swear words 2.
- **Conversational features:** These feature depict the tree-structure of conversation thread. Different from[36] we have used embeddings instead of count statistics. We have leveraged embeddings based on Paragram[54] instead of Word2vec as in [22]. Specifically, we extracted average word embeddings baed on Paragram, paragram vector similarity with source, prev and other content of conversation (obtained by concatenating all other posts in time sequence) and normalized depth of each post in the tree.

1https://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html & additional 1ÃŽÂŠ1ÃŽÂŠ from https://spacy.io/api/annotation/post-tagging
2https://www.cs.cmu.edu/~bglou/resources/bad-words.txt
• **Affective features**: Like in [36] we have extracted three features from DAL[52] and three feature from affective norm rating (ANEW)[35], and an additional AFINN[34] sentiment score different from[36].

• **Emotion Features**: These features help to focus on emotional responses to true and false rumors, similar to[36] i.e. EmoLex[32, 39] and EmoSenticNet[10, 41].

• **LIWC(Linguistic Inquiry and Word Count)**: We have extracted 2 sentiment category feature (PosEMO and NegEMO) and 11 categories[37] specific to each stance as described in[36], which extracted these eleven as dialogue-act features.

• **Speech-Act features**: Different from[36] which extracted eleven categories from LIWC[36], we have extracted speech-act features specific to verb from[53] consisting of 37 categories compiling a total of 229 verbs representing different speech acts.

The model architecture of our proposed system is shown in Figure 1. The proposed system does not use any data augmentation techniques like[56], which is the top performer in RumorEval 2019 (in Sub-Task A). They also used the features different for Twitter and Reddit. This means the performance achieved by our system totally depends on the training dataset provided by RumorEval 2019. Also, we have used the unified set of features for both Twitter and Reddit. This also proves the effectiveness of our approach. Our system outperforms all the previous state of the art approaches used for rumor stance and veracity prediction on RumorEval 2019 data. The remaining part of this paper is organized as follows: Section 2 is an overview of related work. Section 3 define the problem for each task, Section 4 describes our proposed method in detail. Section 5 discusses the experimental evaluation of the system, and finally, Section 6 concludes this paper.

2 RELATED WORK

Within NLP research the task to identify the false content, stance classification of news articles, fact checking etc. has gained momentum to build the automatic system[8, 13, 31]. Initial work on rumor detection and stance classification[27, 30, 42, 43, 59] was succeeded by more sophisticated systems[1, 8, 12, 13, 31, 337]. Stance analysis has been widely studied in different contexts[17, 46]. Specifically, the studies which classify stance towards rumors in social media[30, 42, 43]. Some proposed linguistic feature based technique[14, 58], some used conversational thread in temporal sequence[11, 56, 62], while some used it in tree-structured format[36, 49]. In RumorEval 2019[13], the trend has been toward the neural approaches, with almost all the system used NN(Neural Network) based approaches except only two systems for task B(verbatim classification). The baseline system[22] used LSTM to model the sequential branches in the conversation which is also ranked first in the SemEval 2017 contest. For task A, top performing systems[13] used approaches based on pre-trained models. Like in BUT-FIT[11], which ranked second, used ensemble of BERT[9] based models. Similarly, the best performing system, BLCU-NLP[56] and third best performing system, CLEARumor[5] also used the pre-trained models with BLCU-NLP[56] used OpenAI GPT[44] and CLEARumor[5] used ELMo[38]. Difference lies in all system in their input structure, while BLCU-NLP[56] uses the inference chain i.e. conversation from source post to replies (direct or indirect) arranged in time sequence. Also, linguistic feature has been used along with each post content and trained jointly for both task A and B. For task B, the best performing system (eventAI)[25] approached the problem using an ensemble of classifiers (SVM, LR, RF), including NN with three connected layers. Besides, other features it uses post representation obtained using LSTM with attention. The second ranked system[13] also uses the similar ensemble with sophisticated features and feature selection using RF. There are some previous studies which support the fact that it is necessary to solve the task A as a first step before veracity identification[12, 27, 30, 42, 43, 49, 59, 62].

3 PROBLEM DEFINITION

Consider a conversation thread $C$ originating with a source post $t_1$ followed by a number of reply posts $\{t_2, t_3, \ldots, t_{|C|}\}$ that replies $t_1$ directly or indirectly, and each post $t_i \in [1, |C|]$ has a particular stance category. This paper focuses on two tasks: a) Rumor Stance Classification, aiming to determine the stance of each post $t_i$ in $C$, which belongs to [Supporting, Denying, Querying, Commenting], & b) Rumor Veracity Prediction, with the aim of identifying the veracity of the rumor, belonging to [True, False, Unverified].

4 PROPOSED METHOD

We proposed the multi-task learning framework for jointly predicting rumor stance and veracity. The joint architecture of our system is illustrated in Figure 1 that is composed of two components. The first component is to classify each post in the conversation thread into four different stance labels $\{support, comment, query, deny\}$. It models each post in the multi-turn conversation thread with pre-trained Longformer using sliding window based self-attention[6]. Analogous to CNNs[55], it has multiple stacked layers resulting in large receptive field. It outputs each of the post representation at the corresponding [CLS] token which is then concatenated with feature representation (obtained after passing the NLP Features through a linear layer). This feature augmented post representation is then input to Sentence Encoder (see Section 4.4) to classify each post into the four different stance labels $\{support, comment, deny, query\}$. The second component is to classify the rumor’s veracity into three labels $\{True, False, Unverified\}$ by taking the 1-dimensional mean-pooling of post representations at the output of Sentence Encoder.

4.1 Pre-processing

We have normalized the text to make it suitable for feeding to the Longformer[6]. Same pre-processor as in[11] is used to normalize the text except the tokenizer. Instead, we have used the tokenizer from Hugging Face PyTorch re-implementation of Longformer[5]. Specifically, we have used tweet-preprocessor4 to identify URLs and mentions and replace them with special tokens $\$URLS$ and $\$mentionS$, and spacy5 has been used to split each post into sentences and add the [EOS] token to specify the termination of sentence.

---

3 PROBLEM DEFINITION

Consider a conversation thread $C$ originating with a source post $t_1$ followed by a number of reply posts $\{t_2, t_3, \ldots, t_{|C|}\}$ that replies $t_1$ directly or indirectly, and each post $t_i \in [1, |C|]$ has a particular stance category. This paper focuses on two tasks: a) Rumor Stance Classification, aiming to determine the stance of each post $t_i$ in $C$, which belongs to [Supporting, Denying, Querying, Commenting], & b) Rumor Veracity Prediction, with the aim of identifying the veracity of the rumor, belonging to [True, False, Unverified].

4 PROPOSED METHOD

We proposed the multi-task learning framework for jointly predicting rumor stance and veracity. The joint architecture of our system is illustrated in Figure 1 that is composed of two components. The first component is to classify each post in the conversation thread into four different stance labels $\{support, comment, query, deny\}$. It models each post in the multi-turn conversation thread with pre-trained Longformer using sliding window based self-attention[6]. Analogous to CNNs[55], it has multiple stacked layers resulting in large receptive field. It outputs each of the post representation at the corresponding [CLS] token which is then concatenated with feature representation (obtained after passing the NLP Features through a linear layer). This feature augmented post representation is then input to Sentence Encoder (see Section 4.4) to classify each post into the four different stance labels $\{support, comment, deny, query\}$. The second component is to classify the rumor’s veracity into three labels $\{True, False, Unverified\}$ by taking the 1-dimensional mean-pooling of post representations at the output of Sentence Encoder.

4.1 Pre-processing

We have normalized the text to make it suitable for feeding to the Longformer[6]. Same pre-processor as in[11] is used to normalize the text except the tokenizer. Instead, we have used the tokenizer from Hugging Face PyTorch re-implementation of Longformer[5]. Specifically, we have used tweet-preprocessor4 to identify URLs and mentions and replace them with special tokens $\$URLS$ and $\$mentionS$, and spacy5 has been used to split each post into sentences and add the [EOS] token to specify the termination of sentence.

---

https://huggingface.co/transformers/model_doc/longformer.html
https://github.com/s/preprocessor
https://spacy.io/
4.2 Feature Extraction

We have identified a novel combination of features which are highly
effective to provide indicative clues for rumor stance and veracity,
when learning the fine-tuned representation on top of pre-trained
model. We have introduced two new features in addition to the
previously available features. The first one is the use of Paragram
embeddings[54] to get each of the post representation, and hence-
forth similarity with previous, source and other posts joined in tem-
poral sequence. The second one are the speech-act categories[53]
containing the collection of 229 verbs divided into 37 categories.(see
Section 1) ) we decided to use the Longformer[6] which is the
large (Reddit conversations are usually larger than those of twitter
conversation as input. Since the conversations can be arbitrarily
size, hence fi ∈ R^d_2 ∀ i ∈ [1,|C|].

4.3 Encoding each utterance in conversation thread

As mentioned in Section 1, the nearest neighbors of a post provide
more informative signal for the stance of a post. Based on the above
information we proposed to model the structural and temporal
property to learn the stance feature representation of each post in
the conversation thread. For that it is required to give the complete
conversation as input. Since the conversations can be arbitrarily
large (Reddit conversations are usually larger than those of twitter
see Section 1) ) we decided to use the Longformer[6] which is the
current state-of-art for long contexts datasets like WikiHop[50],
HotPotQA[57], TriviaQA[21] etc.

To use Longformer[6] for encoding each post in the conversa-
tion thread, requires it to output the representation of each sen-
tence/post in the conversation. However, since Longformer (based
on Roberta[28]) is trained as a masked-language model, the out-
put vectors are grounded to tokens instead of sentences (or post-
level in this case). Therefore, we modify the input sequence of

Figure 1: Model Architecture: Each of the individual post is tokenized and the corresponding tokens are separated by [SEP]
token. All the post are arranged in the temporal order and the "NLP Features" extracted for each post, are passed through a
linear layer denoted as f_1, f_2, ..., f_n. Each of the feature f_i is concatenated with the post representation obtained at the output
of Longformer[6] at the corresponding [CLS] token. Finally, stance labels are denoted by Y_i^st & veracity label is denoted by Y_i^ver

---

Footnote:

56 from https://www.ling.upenn.edu/courses/Fall_2003/ling001/ling_treebank_pos.html & àAZàÀZ from https://spacy.io/api/annotation#pos-tagging
The structural features for each post extracted are: a) Feature vector indicating presence of 37 POS tags\(^3\), b) Presence of exclamation mark, negative words, media content like pic, video etc, URL, period, question mark, hashtag, c) Count of chars, words and ratio of capital letters. d) Whether post is the source post. e) Count of exclamation marks, question marks, and periods.

The features extracted to capture the content of each post are: a) 2 features indicating presence of cue[3] and swear words\(^3\) b) Count of false synonyms, antonyms and question word for each post concatenated with that of source post thus 6 features. c) Presence of rumor words like ‘gossip’, ‘hoax’ etc for both source and thread post. d) Presence of words indication unsure about assertion like ‘unconfirmed’, ‘unknown’ for both source and thread post.

There are seven features extracted from three resources a) Three from DAL[52] b) Three from ANEW[35] and c) One is the sentiment score from AFINN[34].

The various encoders we have experimented with are described as feature augmented vectors (post representation) using local attention based on sliding window.

As illustrated in Figure 1, we insert a [CLS] token before each sentence and a [SEP] token after each sentence. In vanilla Longformer (which is trained from Roberta[28] checkpoint), the [CLS] token is used to aggregate features from one sentence or a pair of sentences using global attention[6]. We modify the model by using multiple [CLS] tokens to get sentence vector (post representation) using local attention based on sliding window. After obtaining the sentence vectors from Longformer[6] each of the vector \(t_i \in R^{d_i}\) has been concatenated with their corresponding feature representation \(f_i \in R^{d_i}\) given as

\[
k_i = t_i \oplus f_i \in R^{d_i+d_i} \forall i \in [1,|C|]
\]

### 4.4 Sentence Encoder

The feature augmented vectors \(k_i \in R^{d_i+d_i}\) are then fed to the encoder-specific layers stacked on top of Longformer[6]. These encoder layers are jointly fine-tuned with Longformer[6] to learn the temporal evolution of stance in the conversation (see Figure 1). The various encoders we have experimented with are described as follows:

- **Identity Encoder.** We have used this encoder for the sake of comparison with other encoders and to compare the robustness of the representation we get at the output of Longformer[6] versus other encoders. Encoded representation in this case is just \(k_i\) only given as:

\[
h_i = k_i \in R^{m_i} \forall i \in [1,..|C|]
\]

In this case, \(m = d_1 + d_2\).

Inter-Sentence Transformer. Inter-Sentence Transformer applies additional Transformer[47] layers only on feature augmented post representation i.e. \(k_i\) to extract the relation between post and their corresponding stances. The encoded representation for a layer \(l\) is given as follows:

\[
\tilde{g}_l = LN(g_l^{l-1} + MHATT(g_l^{l-1}))
\]

\[
g_l^l = LN(g_l^{l-1} + FFN(\tilde{g}_l^{l-1}))
\]

where \(g_0 = k_i \oplus \text{PosEMB}(k_i)\), where \(K = [k_1, k_2, ..., k_{|C|}]\). Here, the symbols like PosEMB, LN, FFN, MHATT are the function to add Positional Embeddings, Layer Normalization, Feed-Forward Network, Multi-Head Attention respectively. These function implementation is taken as it is described in Transformer[47]. Let there be \(L\) number of layers, final output is given as follows:

\[
h_l = g_L^l \in R^m \forall i \in [1,..|C|]
\]

In experiments, we implemented Transformers with \(L = 1, 2, 3\) and found Transformer[47] with 2 layers performs the best.

Recurrent Neural Network (RNN). Although, the pre-trained transformers achieved state-of-art on several tasks. But, RNN stacked over transformer seem to achieve better results[7]. We have used LSTM[19] in this case to learn the task specific representation. Similar to[2] we have also applied Layer normalization per-gate to each LSTM cell.

\[
\begin{pmatrix}
F_l \\
I_l \\
O_l \\
C_l
\end{pmatrix} = LN_h(W_h r_{l-1}) + LN_x(W_x k_l)
\]

\[
C_l = \sigma(F_l) \odot C_{l-1} + \sigma(k_l) \odot \tanh(G_{l-1})
\]

\[
r_l = \sigma(O_l) \odot \tanh(LN_c(C_l))
\]

Longformer[6] to make it possible for extracting post representations.

**Encoding Multiple Sentences.** As illustrated in Figure 1, we insert a [CLS] token before each sentence and a [SEP] token after each sentence. In vanilla Longformer (which is trained from Roberta[28] checkpoint), the [CLS] token is used to aggregate features from one sentence or a pair of sentences using global attention[6]. We modify the model by using multiple [CLS] tokens to get sentence vector (post representation) using local attention based on sliding window. After obtaining the sentence vectors from Longformer[6] each of the vector \(t_i \in R^{d_i}\) has been concatenated with their corresponding feature representation \(f_i \in R^{d_i}\) given as

\[
k_i = t_i \oplus f_i \in R^{d_i+d_i} \forall i \in [1,|C|]
\]

### Table 1: NLP Features

| Feature Name            | Description                                                                 | Feature Count |
|-------------------------|------------------------------------------------------------------------------|---------------|
| Structural Features     | The structural features for each post extracted are: a) Feature vector indicating presence of 37 POS tags\(^3\), b) Presence of exclamation mark, negative words, media content like pic, video etc, URL, period, question mark, hashtag, c) Count of chars, words and ratio of capital letters. d) Whether post is the source post. e) Count of exclamation marks, question marks, and periods. | 51            |
| Content Features        | The features extracted to capture the content of each post are: a) 2 features indicating presence of cue[3] and swear words\(^3\) b) Count of false synonyms, antonyms and question word for each post concatenated with that of source post thus 6 features. c) Presence of rumor words like ‘gossip’, ‘hoax’ etc for both source and thread post. d) Presence of words indication unsure about assertion like ‘unconfirmed’, ‘unknown’ for both source and thread post. | 12            |
| Conversational Features | Feculiar to tree-structured conversation we have extracted various features a) Faragram Embedding[34] representation for each post in the conversation.(300) b) Similarity between thread post and source post c) Similarity between thread post and previous post d) Similarity with concatenated posts (other than source and previous) in time order e) Normalized Depth of post in the tree structured conversation. | 305           |
| Affective Features      | There are seven features extracted from three resources a) Three from DAL[52] b) Three from ANEW[35] and c) One is the sentiment score from AFINN[34]. | 7             |
| Emotion Features        | Extracted 8 primary emotion based on Plutchik model[32, 39], extra 2 signals is also extracted indicating positive and negative emotion. Additionally, 6 basic emotion categories from EmoSentSensNet[10, 41]. | 16            |
| LIWC                    | There are 11 categories of LIWC[37] are further grouped according to four stance categories as a) agree-accept(Support) - Assert, Certain, Effect; b) Reject(Deny) - Negate, Inhibit; c) Info-request(Query) - You, Cause; d) Opinion(Comment) - Future, Sad, Insight, Cogmech. Additionally, two categories indicating positive and negative sentiment is also extracted. | 13            |
| Speech-Act Features     | Certain verbs like ask, demand, promise report etc. that categorize the different speech acts, also able to indicate the stance category which in turn indicate the rumor veracity. Extracted 37 categories[53] which compiled a total of 229 verbs across different categories. | 37            |
| Total Features          |                                                                                       | 441           |
where \( F_i, I_i, O_i \) are forget gates, input gates, output gates; \( G_i \) is the hidden vector and \( C_i \) is the memory vector; \( r_i \) is the output vector; \( L_{h}, L_{en}, L_{g} \) are the layer normalization operations at output, input and memory vector respectively; Bias terms are not shown. The final output vector after linear layer is given as:

\[
h_i = r_i \in R^m \forall i \in [1, 2, ..., |C|]
\]  

(9)

### 4.5 Rumor Stance and Veracity Classification

After getting the fine-tuned post representation \( h_i \) from the sentence encoder, it has been used to classify the stance of each post and rumor veracity each has been described as follows:

**Stance Classification.** For stance classification each of the fine-tuned representation \( h_i \in R^m \forall i \in [1, 2, ..., |C|] \) is to be classified among four labels \{support, comment, deny, query\} numbered as \{0, 1, 2, 3\}. For each post \( t_i \) in the conversation \( C \), we apply softmax to obtain its predicted stance distribution:

\[
\hat{y}_i^{st} = \text{softmax}(W_{st}h_i + b_{st})
\]

(10)

where \( W_{st} \in R^{4x(m)} \) and \( b_{st} \in R^4 \) are weight matrix and bias respectively. The loss function of \( C \) for stance classification is computed by cross-entropy criterion:

\[
L_{stance} = \frac{1}{|C|} \sum_{i=1}^{|C|} - (y_i^{st})^T \log \hat{y}_i^{st}
\]

(11)

where gold label \( y_i^{st} \) is the one-hot vector denoting the stance label for the post \( t_i \). For batched training the cross-entropy loss is the average cross-entropy over the number of examples in a batch.

**Veracity Classification.** The fine-tuned post representation vectors \( \{h_1, h_2, ..., h_{|C|}\} \) are the output sequence that represents the temporal feature. We then transform this temporal sequence to a vector \( v \) by a 1-dimensional global mean-pooling to capture the stance evolution. The mean-pooled representation \( \hat{h} \) is then used for veracity classification after passing through linear layer and softmax normalization. For veracity classification, there are three labels \{True, False, Unverified\} numbered as \{0, 1, 2\}. The predicted distribution over veracity labels is given as:

\[
\hat{h} = \text{mean-pooling}(h_1, h_2, ..., h_{|C|}),
\]

(12)

\[
\hat{y}_i^{ver} = \text{softmax}(W_{ver} \hat{h} + b_{ver})
\]

(13)

where \( W_{ver} \in R^{3x(m)} \) and \( b_{ver} \in R^3 \) are weight matrix and bias respectively. The cross-entropy loss function of \( C \) for veracity classification is given as:

\[
L_{veracity} = -y_i^{ver} \log \hat{y}_i^{ver}
\]

(14)

where the gold label \( y_i^{ver} \) is the one-hot vector denote the veracity label for the rumor started with post \( t_0 \).

### 4.6 Jointly Learning Two Tasks

As mentioned in Section 1, the stance evolution indicates the rumor veracity so we should leverage the interrelation between the two task i.e. stance classification and subsequent task which is veracity classification. We have trained these two tasks jointly by adding the loss function for each task with a trade-off parameter \( \lambda \) and optimize them jointly. Specifically the joint loss \( L \) is given as:

\[
L = L_{veracity} + \lambda L_{stance}
\]

(15)

### 4.7 Ensembling

Our overall architecture of model consists of Longformer[6] as a base model on top of that various sentence encoder has been put to learn the varying features for stance classification and on top of that we leverage the stance evolution for veracity classification. So, we have trained and finally saved 50 best models by varying the learning rates and encoders. We have used the Top-\( N_s \) fusion strategy as described in[11] inorder to increase the F1 measure and reduce overfitting. This procedure iteratively selects 1 model after random shuffling the pool of models and adds it to ensemble, if it increase the ensemble’s F1 by averaging the output probabilities, effectively approximating the Bayesian model averaging. Specifically, in Top-\( N_s \) strategy we take the average of pre-softmax scores instead of output probabilities.

### 5 EXPERIMENTS

In this section, we first evaluate the performance of rumor stance classification and then veracity prediction (Section 5.3). We then give a detailed analysis of our proposed method (Section 5.4 and 5.5).

| Support | Deny | Query | Comment | Total |
|---------|------|-------|---------|-------|
| Twitter Train | 1004 | 415 | 464 | 3685 | 5568 |
| Reddit Train | 23 | 45 | 51 | 1015 | 1134 |
| Total Train | 1027 | 460 | 515 | 4700 | 6702 |

### Table 2: Task A Corpus

| True | False | Unverified | Total |
|------|-------|------------|-------|
| Twitter Train | 145 | 74 | 106 | 325 |
| Reddit Train | 9 | 24 | 7 | 40 |
| Total Train | 154 | 98 | 113 | 365 |

### Table 3: Task B Corpus

5.1 Data & Evaluation Metric

We have used the data released at RumorEval 2019 for both subtask A(stance classification) and B(veracity classification)[13]. For each task the distribution of train and test is shown in Table 2 and 3. We have used the same evaluation metric as in RumorEval 2019[13] i.e. Macro-averaged F1 score and RMSE (For task a only).

5.2 Implementation Details

We implemented our models in Pytorch using Hugging Face implementation of Longformer\(^6\) with pre-trained parameters loaded from

\(^6\)https://huggingface.co/transformers/model_doc/longformer.html
“longformer-base-4096” having 12 transformer layers, hidden unit size of $d = 768$, 12 attention heads, vocab size = 50265, max length = 4096. Longformer, sentence encoder and task related classification layers are jointly fine-tuned. For conversation which does not fit into max length of 4096 we have created multiple examples using sliding window at post level (adding one post at a time and remove from source end). Since the sentence encoder and classification layers has to be trained from scratch, while Longformer is already pre-trained this may cause unstability for example pre-trained one may overfit the data while sentence encoder underfits. Therefore, we have used two Adam optimizers with $\beta_1 = 0.9$ and $\beta_2 = 0.999$ for pre-trained (P) and other components (OC) respectively, each with different warmup-steps and learning rates:

$$lr_p = \frac{lr_p \cdot \text{min(\text{step}^{-0.5}, \text{step} \cdot \text{warmup}_p^{1.5})}}{} \quad (16)$$

$$lr_{OC} = \frac{lr_{OC} \cdot \text{min(\text{step}^{-0.5}, \text{step} \cdot \text{warmup}_{OC}^{1.5})}}{} \quad (17)$$

The size of the Linear Layer used to obtain feature representation is taken as 128. In case of transformer as sentence encoder the output hidden size is same as input one (in our case 768+128) and number of layers = 2, while in case of Bi-LSTM output size is 512 where 256-dimensional vector is obtained from each direction. Also, for joint training $\lambda = 0.7$ gives the best results.

5.3 Evaluation Strategy
In this section we evaluate our proposed approach against the top-5 systems submitted at RumorEval 2019[13] for Task A and Task B. We have conducted the separate experiments, to properly investigate the performance of a) each of the classifiers used in the ensemble. b) impact of NLP features on each of those classifiers and finally, c) the performance of our proposed system. In Table 5, 4 and 6, models named as Longformer + Identity Encoder, Longformer + Transformer and Longformer + BiLSTM are corresponding models with different types of sentence encoders (see Section 4.4) over the base model Longformer (Figure 1) without concatenating NLP features (see Section 4.2) at the fine-tuned post representation as shown in figure. Similarly, Longformer + Identity Encoder + NLP Features, Longformer + Transformer + NLP Features, Longformer + BiLSTM + NLP Features are corresponding models with NLP features. Specifically, for the model named Longformer + Identity Encoder is only a base model (Longformer) without any encoder to judge the performance boost as compared to when we use BiLSTM or Transformer encoder. Here, NLP Features have been applied to Linear Layer (of size 128), output of which has been concatenated to fine-tuned representation obtained at the output of sentence encoder (see Section 4.4). In
5.4 Results and Discussion

In this paper, we have evaluated our models using the same guidelines as in RumorEval 2019 contest paper[13]. Specifically, they have used macro-averaged F1 to evaluate the performance on both Task A and B. Additionally, they have used RMSE score for Task B to judge the confidence scores. We followed the same guidelines as provided by the baseline system[22] to calculate the score for our proposed system. Table 5, 4 and 6 presents the comparative experimental results for the proposed method in this paper with respect to the state-of-the-art. The top-5 systems[13] given in Table 5 and 4 are the best performing systems as per the published results in RumorEval 2019 paper[13]. From the results given in the Table 5, 4 and 6 it is clear that our proposed method shows the best performance among all the approaches. These results also state the importance of NLP Features and Sentence Encoder. We will discuss the effect of each in the following sections.

**Effect of NLP Features.** To understand the importance of NLP Features, we conduct an ablation study: we only input the post representation to the classification layer with or without sentence encoder(ref. Table 5, 4). The results state that the sentence encoder only models the temporal variation of post representation but not able to capture the cause for the particular category for stance or veracity.

**Effect of Sentence Encoder.** We have studied the impact of using the sentence encoder with or without NLP Features using the model named as Longformer + Identity Encoder and Longformer + Identity Encoder + NLP Features. In both cases, the performance in terms of F1 is less than as compared to when we use the encoder (either BiLSTM or Transformer). The results state that the sentence encoders helps to learn the stance evolution to determine the proper category of veracity and neighbouring posts helps in determining the stance category.

5.5 Case Study

An example of the “false” rumor identified by our model is illustrated in Figure 2. It illustrates the conversation thread starting from source post followed by subsequent replies in the temporal order. As seen, the stance evolution contains a sequence "query → comment → comment → comment → deny". Since, the source post does not indicate of why this should be a false rumor, our model captures the stance evolution using fine-tuned representations obtained at the output of sentence encoder (see Section 4.4), and hence accumulated information using average pooling correctly identifies the rumor veracity.

6 CONCLUSION AND FUTURE WORK

In this paper we have briefly described the multi-task approach for joint prediction of rumor stance and veracity for data obtained from various social media platforms (in our case Twitter and Reddit). We presented ensemble of deep learning models having the same architecture, but varying the parameters. Our approach outperforms the previous approaches by sufficient margin and able to generalize across different social media. In future, we can extend our model for multilingual setting[51] (IberEval is the counterpart of RumorEval 2019 for other languages[13]). Moreover, we can leverage more sophisticated resources like pre-trained model which was trained specifically to handle data from different social media platforms. Further, we can explore other methods like diffusion process of multilingual setting[51].
[44] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding with unsupervised learning. Technical report, OpenAI (2018).

[45] Victoria L. Rubin. 2017. Deception detection and rumor debunking for social media. In The SAGE Handbook of Social Media Research Methods. Sage, 342.

[46] Swapna Somasundaran and Janyce Wiebe. 2009. Recognizing Stances in Online Debates. In Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP. Association for Computational Linguistics, Suntec, Singapore, 226–234. https://www.aclweb.org/anthology/P09-1026

[47] 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 Advances in neural information processing systems. 5998–6008.

[48] Soroush Vosoughi, Deb Roy, and Sinan Aral. 2018. The spread of true and false news online. Science 359, 6380 (2018), 1146–1151.

[49] Penghui Wei, Nan Xu, and Wenji Mao. 2019. Modeling conversation structure and temporal dynamics for jointly predicting rumor stance and veracity. arXiv preprint arXiv:1909.08211 (2019).

[50] Johannes Welbl, Pontus Stenetorp, and Sebastian Riedel. 2018. Constructing datasets for multi-hop reading comprehension across documents. Transactions of the Association for Computational Linguistics 6 (2018), 287–302.

[51] Wuming Wen, Songwen Su, and Zhou Yu. 2018. Cross-Lingual Cross-Platform Rumor Verification Pivoting on Multimedia Content. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics, Brussels, Belgium, 3487–3496. https://doi.org/10.18653/v1/D18-1385

[52] Cynthia Whissell. 2009. Using the revised dictionary of affect in language to quantify the emotional undertones of samples of natural language. Psychological reports 105, 2 (2009), 509–521.

[53] Anna Wierzbicka. 1987. English speech act verbs: A semantic dictionary. Academic Pr.

[54] John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2015. Towards universal paraphrasable sentence embeddings. arXiv preprint arXiv:1511.08198 (2015).

[55] Felix Wu, Angela Fan, Alexei Baevski, Yann N Dauphin, and Michael Auli. 2019. Pay less attention with lightweight and dynamic convolutions. arXiv preprint arXiv:1901.10430 (2019).

[56] Ruoyao Yang, Wanying Xie, Chunhua Liu, and Dong Yu. 2019. BLCU_NLP at SemEval-2019 Task 7: An Inference Chain-based GFT Model for Rumour Evaluation. In Proceedings of the 13th International Workshop on Semantic Evaluation. 1090–1096.

[57] Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. arXiv preprint arXiv:1809.09600 (2018).

[58] Li Zeng, Kate Starbird, and Emma S Sprio. 2016. # unconfirmed: Classifying rumor stance in crisis-related social media messages. In Tenth International AAAI Conference on Web and Social Media.

[59] Qiao Zhang, Shuyuan Zhang, Jian Dong, Jimhua Xiong, and Xueqi Cheng. 2015. Automatic detection of rumor on social network. In Natural Language Processing and Chinese Computing. Springer, 113–122.

[60] Zhe Zhao, Paul Resnick, and Qiaozhu Mei. 2015. Enquiring Minds: Early Detection of Rumors in Social Media from Enquiry Posts. In Proceedings of the 24th International Conference on World Wide Web (Florence, Italy) (WWW ’15). International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 1395âĂ–1405. https://doi.org/10.1145/2736277.2741637

[61] Arkaitz Zubiaga, Elena Kochkina, Maria Liakata, Rob Procter, and Michal Lukasik. 2016. Detection and resolution of rumours in social media: A survey. ACM Computing Surveys (CSUR) 51, 2 (2018), 1–36.

[62] Arkaitz Zubiaga, Elena Kochkina, Maria Liakata, Rob Procter, and Michal Lukasik. 2016. Stance classification in rumours as a sequential task exploiting the tree structure of social media conversations. arXiv preprint arXiv:1609.09028 (2016).

[63] Arkaitz Zubiaga, Maria Liakata, Rob Procter, Geraldine Wong Sak Hoi, and Peter Tolmie. 2016. Analysing how people orient to and spread rumours in social media by looking at conversational threads. PloS one 11, 3 (2016), e0150989.