Tree LSTMs with Convolution Units to Predict Stance and Rumor Veracity in Social Media Conversations

Sumeet Kumar
Carnegie Mellon University
5000 Forbes Ave,
Pittsburgh, PA 15213, USA
sumeetku@cs.cmu.edu

Kathleen M. Carley
Carnegie Mellon University
5000 Forbes Ave,
Pittsburgh, PA 15213, USA
kathleen.carley@cs.cmu.edu

Abstract

Learning from social-media conversations has gained significant attention recently because of its applications in areas like rumor detection. In this research, we propose a new way to represent social-media conversations as binarized constituency trees that allows comparing features in source-posts and their replies effectively. Moreover, we propose to use convolution units in Tree LSTMs that are better at learning patterns in features obtained from the source and reply posts. Our Tree LSTM models employ multi-task (stance + rumor) learning and propagate the useful stance signal up in the tree for rumor classification at the root node. The proposed models achieve state-of-the-art performance, outperforming the current best model by 12% and 15% on F1-macro for rumor-veracity classification and stance classification tasks respectively.

1 Introduction

Online misinformation, commonly called 'fake news', has become a serious problem in society (Ferrara, 2015) to the extent that they are impacting election decisions (Allcott and Gentzkow, 2017). Many machine-learning approaches have been proposed to identify and contain the fake-news shared on online social-media platforms (Jin et al., 2016; Rubin et al., 2016; Rubin and Lukoianova, 2015; Schifferes et al., 2014; Tacchini et al., 2017; Volkova et al., 2017; Vosoughi et al., 2018). One approach that combines machine-learning and human-intelligence by exploiting stance in reply posts has gained significant attention recently (Zubiaga et al., 2016a, 2015). In this approach, we first identify the stance – categorized as 'supporting', 'denying', 'commenting' and 'querying' – in the replies to the original post and then use the stance signal to find rumor veracity i.e. if a rumor is true or false. Prior work has confirmed that replies to a 'false' (misleading) rumor contain specific patterns, e.g. more replies deny the claim made in the source post (Zubiaga et al., 2016b). This approach is promising as people are reasonably good at pointing out misinformation (Babcock et al., 2019) and if such posts could be automatically found, the post could go through enhanced scrutiny before it gets circulated widely.

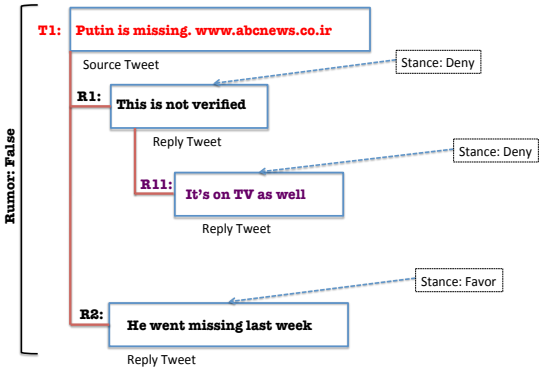

Figure 1: Twitter threads with stance and rumor-veracity labels. The conversation tree shown above has two branches a) T1–R1–R11 and b) T1-R2. R1 and R2 are 1^{st} level reply tweets and R11 is a 2^{nd} level reply tweet. Stance labels for each reply is relative to the tweet it is replied to i.e. stance for R11 is with-respect-to R1. There is a rumor-veracity label on the root tweet (T1 in the example above). The goal of this research is to learn the root tweet's veracity based on pattern in replies.

In this research, we extend this line of work on rumor-veracity and stance learning by proposing a new way to represent conversation trees and new LSTM cells that could be used to detect rumors more effectively. In past, researchers have explored various models to learn from tree structured

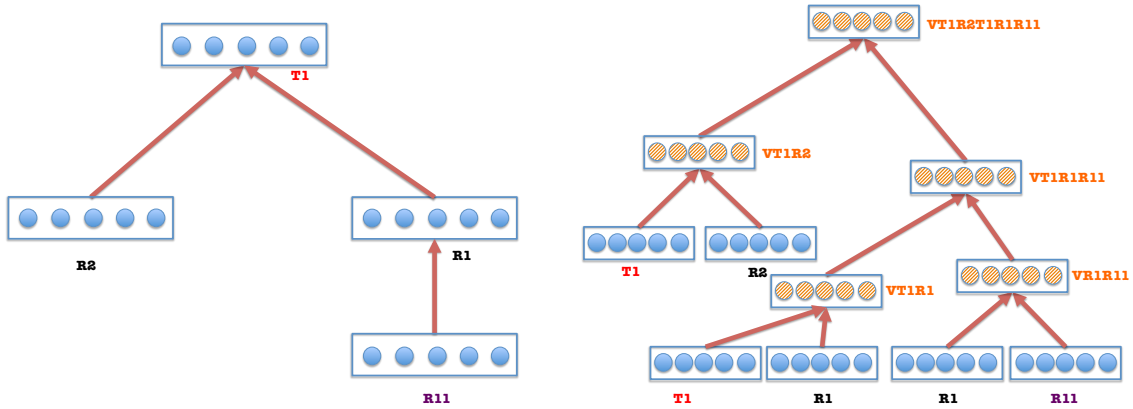

Figure 2: Normal tree structure (left) and the modified binarized constituency tree (BCTree) structure for the conversation shown in Fig. 1. On left, a tree with structure representing the original thread in which a node can have any number of children. On right, a binary tree structure where source post and reply posts are all leaf nodes such that each reply is placed next to the tweet it was made against and connected to a virtual parent node. E.g. R11 was made against R1 so are connected to VR1R11.

data (Wang et al., 2007; Gildea, 2004). For rumor veracity classification, prior research have found that the approach that performs the best on social-media conversations is a sequence model (like the Long Short Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) as discussed in (Zubiaga et al., 2018)). Sequential classifiers like LSTMs are good at learning temporal structure and are biased to use prior inputs to predict outputs (Eck and Schmidhuber, 2002). However, when it comes to comparison tasks like stance classification in threaded discussions, each reply is made against a post or a response to a source post (see Fig. 1). So, we ask, is the regular sequential model apt to learn the relationship between a source post and its replies in conversations? Would a model that can learn the contrast between a source and the reply tweets be more appropriate for rumor classification? To this end, we propose a new tree structure that is obtained from social-media conversation trees but allows for easy comparison of the source and its replies. Additionally, we use a convolution unit to learn patterns in local features for stance classification, and the tree model propagates the signal up the tree for the rumor classification at the root of the tree.

To evaluate our models, we use a human-labeled Twitter dataset that contains stance labels and rumor labels for around two thousand rumour threads related to five different events. Our proposed models achieve the state-of-the-art performance, outperforming the current best model by 12% and 15% on F1-macro for rumor classification and stance classification tasks respectively.

2 Models for Tree Structured Social Media Conversations

Tai et al. 2015 proposed a tree structured LSTM networks and showed its utility on two tasks of semantic relatedness and sentiment classification. In their work, the tree LSTM is composed of sentence sub-phrases using a given syntactic structure. The benefits of using a recursive tree approach was discussed by Li et al. (Li et al., 2015) where the authors concluded that tree models are more suitable for root level identification. Social-media conversations are naturally structured as trees. Can Tree LSTMs be used for classifying node labels in such conversations trees? In this work, we try to answer this question by modeling conversations as trees where each node in the tree is a sentence representation (Fig. 2). Node labels in tree structured conversations can be learned using: a) branches of the tree as input to an LSTM (Branch LSTM Model) as used in many prior research e.g. (Zubiaga et al., 2016a, 2018) b) using the entire tree as the input (Tree LSTM Model) c) modifying the structure of the tree to better capture the inherent correlations in conversations for a given task (Binarized Constituency Tree LSTM Model). We discuss these formulations next.

2.1 Branch LSTM Model

In branch LSTM, the encodings of source-tweet text and the replies text along a tree branch are used as the input and the stance-labels are used as the output (as illustrated in Fig. 3). Using a simple text encoder (like mean of a word vectors), at each step, the LSTM gets a sentence embedding and predicts a label. The process is repeated for all nodes in the thread. For example, if we take the thread (T1-R1-R11) (see an example thread in Fig. 1), the LSTM takes the R11 as the input in the first time step, R1 as the input in the second time step and T1 as the input in the third time step.

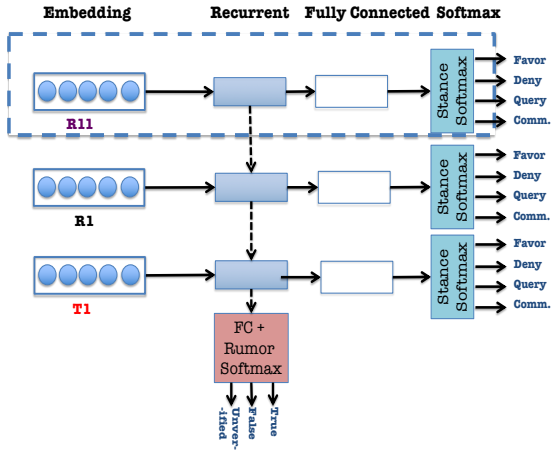

Figure 3: Branch LSTM: Recurrent Neural Network (RNN) architecture for sequence labeling. T1 , R1 and R11 are embeddings. At each time step, the LSTM uses a sentence embedding vector as input to output a stance label. At the root node T1, the RNN outputs a rumor-veracity label.

Modelling tree conversations as branches of the tree has two limitations: a) repetition of input as many branches share nodes (e.g. root node is present in all branches) b) no communication between branches during the learning process. The LSTM uses branches independently. Thus, there is no communication between branches during training and inference. We expect that not all branches are useful to predict the veracity of a rumor post and a few branches might have stronger signal. The branch LSTM weighs all branches equally and therefore, is likely to under perform when there are many uninformative branches in a tree. This problem is solved in Tree LSTM.

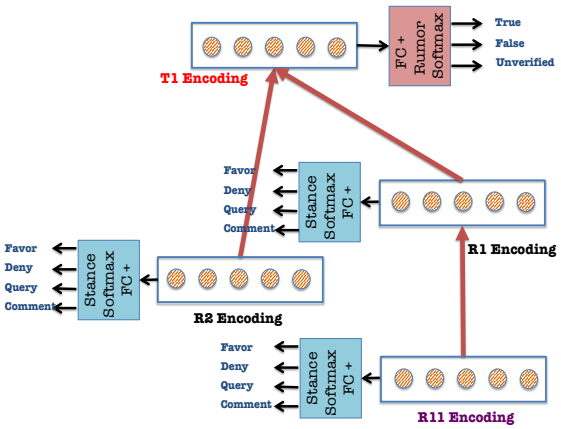

Figure 4: Tree LSTM model: Latent vectors at all nodes (except the root node) are used to predict stance label and the latent vector at the root node is used to predict the rumor-veracity label of the conversation.

2.2 Tree LSTM Model

A typical social-media conversations consists of a post (source post), its reply and reply to the replies. This is a tree structure with the source post as the root node and the replies as the child nodes. Models for such tree structures was explored in (Tai et al., 2015) where authors suggested a modification of the LSTM cell to accommodate an unknown number of inputs at a node. For a general tree with any number of child nodes, they suggested 'Child Sum Unit' that sums the hidden vectors of child nodes (as in eqn. 8). We generalize this formulation to accommodate other operations as shown in Fig. 4.

$$\tilde{h} = O_{k \in C(j)} h_k \tag{1}$$

where $C(j)$ denotes the set of children of node j and O_k is an operator that acts on the hidden vector h_k of child k to output $\tilde{h}$. Using this, we define the LSTM transition equations as follows:

$$i_j = \sigma\left(W^{(i)} x_j + U^i \tilde{h_j} + b^{(i)}\right) \tag{2}$$

$$f_{jk} = \sigma\left(W^{(f)} x_j + U^{(f)} h_k + b^{(f)}\right) \tag{3}$$

$$o_j = \sigma\left(W^{(o)} x_j + U^o \tilde{h_j} + b^{(o)}\right) \tag{4}$$

$$u_j = tanh\left(W^{(u)} x_j + U^{(u)} \tilde{h_j} + b^{(u)}\right) \tag{5}$$

$$c_j = i_j \odot u_j + \sum_{k \in C(j)} f_{jk} \odot c_k \quad (6)$$

$$h_j = o_j \odot tanh(c_j) \quad (7)$$

Except wherever specified, the notations used are of standard Tree LSTM as described in Tai et al. 2015.

2.2.1 Child Sum Tree Unit

The child-sum unit involves using sum of all h_k vectors which means $O = \sum$. Therefore

$$\tilde{h} = \sum_{k \in C(j)} h_k \quad (8)$$

2.2.2 Child Max-Pooling Unit

The child max-pooling unit involves using the maximum of all h_k vectors across a dimension. Therfore

$$\tilde{h} = \max_P {}_{k \in C(j)} h_k \quad (9)$$

2.2.3 Child Convolve + MaxPooling Tree Unit

Child convolve uses convolution operation of the set of child hidden vectors i.e. $O = \circledast$ where $\circledast$ denotes vector convolution operation. As a normal tree node can have any number of child nodes, convolution operation using all child nodes requires a max-pooling operation to preserve the dimension of $\tilde{h}$.

$$\tilde{h} = \max_P \circledast_{k \in C(j)} h_k \quad (10)$$

where $\circledast$ denotes vector convolution operation and $\max_P$ denotes max pooling operation. A 2d convolution over h matrix results in another matrix and the max pooling operator maps the matrix to vector containing the maximum value of each column in the matrix.

A neural-network model (like an LSTM) expects a pre-defined size of input. Using an operation that reduces the children hidden layer matrix $\tilde{h}$ to fixed dimension vector like in equation 8 or in equation 10 attempts to solve the problem. However, these reduction operators have limitations e.g. 'sum' weighs all children equally and 'convolve+maxpool' only picks the convoluted features with maximum value. Ideally this importance factor should be learned from data itself, which is what we intend to achieve using Binarized Constituency Tree (BCTree) LSTM Model.

2.3 Binarized Constituency Tree (BCTree) LSTM Model

Social media conversations are in the format of a tree where a node can have many children. Converting this tree structure to another tree structure in which each node always contain two children creates a consistent format which is convenient for matrix operations needed to train neural networks. Additionally, for tasks like stance learning, where its important to compare a reply against its source post, a source reply-pair should be placed such that the contrast features can be effectively learned. To achieve this, we modify the original structure to a binary tree which we call Binarized Constituency Tree (BCTree).

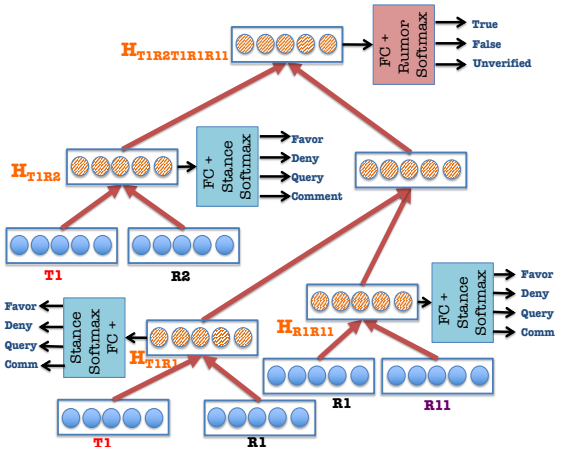

Figure 5: BCTree LSTM model: Latent vectors at virtual parent node of each leaf node is used to predict stance labels (e.g. H_{R1R11} to predict stance of $R11$) and the latent vector at the root node is used to predict the rumor-veracity label of the conversation.

In BCTree, all source posts and their replies appear as leaf nodes (Fig. 5). A reply is always paired with its source (this requires source node to be duplicated) and they are connected to a new (virtual) parent node. To construct a BCTree from a tree, we replace all parent node with a new virtual node. The original parent node and a child node are then connected to the new virtual parent node. If a parent node has more than one child, additional virtual nodes are created to keep the tree binary.

Because each node in a BCTree always has only two children, and therefore is consistent, many operators are trivially supported. E.g. we can use hidden vector concatenation. Similarly, for convolution, a convolution unit with kernel size 2 and

stride size 1 (comparing a source post and a reply) preserves the dimension of h_k (as BCTree node always have 2 children). Thus additional operation like 'Sum' or 'MaxPooling' is not needed.

2.3.1 Child Sum BCTree Unit

This uses the same operation as in the normal tree structure (see equation 8).

2.3.2 Child Concat BCTree Unit

$$\tilde{h} = \oplus_{k \in C(j)} h_k \tag{11}$$

where $\oplus$ denotes vector concatenation operation.

2.3.3 Child Convolve BCTree Unit

$$\tilde{h} = \circledast_{k \in C(j)} h_k \tag{12}$$

where $\circledast$ denotes vector convolution operation.

2.3.4 Combinations of BCTree Units

Because a BCTree has a uniform structure, any combination of the previous discussed units can also be combined together. Some possible combinations we try are 'Convolve + Concat', 'Convolve + Sum ' and 'Convolve + Concat + Sum '.

3 Experiments and Results

3.1 Datasets

We use Pheme 5 events dataset. This dataset was created as a part of the Pheme project [1] which aims to find and verify rumors shared on social-media platforms (Zubiaga et al., 2015, 2016b). The dataset consist of Twitter conversation threads on five different events and contains three types of annotations. Each thread is labeled as either rumor or non-rumor. Rumors are annotated for their veracity as 'true', 'false' or 'unverified' (see Tab. 1). For a subset of the true rumors, we also have stance labels for each reply in the threaded conversations. The stance labels are 'support', 'deny', 'comment' and 'query' (see Tab. 2). As we can observe in Tab. 2, this dataset is highly skewed towards 'comment'.

3.2 Feature Representation

We use four different models that have shown good results on various NLP tasks to extract text features.

[1] https://www.pheme.eu/

Events	True	False	Unverified
Charlie Hebdo (CH)	193	116	149
Sydney siege (SS)	382	86	54
Ferguson (FG)	10	8	266
Ottawa shooting (OS)	329	72	69
Germanwings-crash (GC)	94	111	33
Total	1008	393	571

Table 1: Conversation threads in the Pheme dataset

Events	Support	Deny	Query	Comment
CH	239	58	53	721
SS	220	89	98	700
FG	176	91	99	718
OS	161	76	63	477
GC	69	11	28	173
Total	865	325	341	2789

Table 2: Stance labels for Tweets in the conversations. Event codes are described in Tab. 1

3.2.1 Mean of Glove word vectors

To get word vectors, we used Glove (Pennington et al., 2014) and the mean of these word vectors are used as the sentence embedding. Before extracting the Glove word vectors, we perform some basic text cleaning which involves removing any @mentions, any URLs and the Twitter artifact (like 'RT') which gets added before a retweet. Some tweets, after cleaning did not contain any text (e.g. a tweet that only contains a URL or an @mention). For such tweets, we generate an embedding vector containing uniformly generated numbers between -0.5 and 0.5. The same text cleaning was performed before generating features for all embeddings described in the rest of the paper.

3.2.2 BERT embeddings

BERT [2] is not a ready to use model to generate embeddings in its original form. It is rather a model that can be tuned for a task (Devlin et al., 2018). We first tried to tune the model on our rumor classification task. But since the rumor classification dataset is relatively small, while evalu-

[2] https://github.com/huggingface/pytorch-pretrained-BERT

ating we found that tuning did not lead to a good performance. We then considered other datasets that can be used for tuning. Because natural language entailment task (which predicts entailment, contradiction, or neutral between two sentences) is similar to stance learning, we use the BERT model and tune it on Multi-Genre Natural Language Inference task (Williams et al., 2018). The tuned model is then used to generate BERT embedding which is the vector representation on the last layer of the Bert model. This tuned BERT model generates a 768 dimension vector for each sentence.

3.2.3 Skipthought (SKP) embeddings

We use the pre-trained model shared by the authors of Skipthought (Kiros et al., 2015) [3]. The model uses a neural-network that takes sentences as input and generate a 4800 dimension embedding for each sentence. Thus, on our dataset, for each post in Twitter conversations, we get a 4800 dimension vector.

3.2.4 DeepMoji (EMT) embeddings

We use the DeepMoji (Felbo et al., 2017) pre-trained model [4] to generate deepmoji vectors. Like skipthought, DeepMoji is a neural network model that takes sentences as input and outputs a 64 dimension feature vectors.

3.2.5 Skipthought and DeepMoji joint (SKPEMT) embeddings

Because DeepMoji and Skipthoughts are different types of encodings, we also tried a concatenated version of them which we call SKPEMT. This encoding is of size 4864 dimension.

3.3 Models Training

Following the convention in prior work (Zubiaga et al., 2018), we use event wise cross-validation, which means out of five events, four events are used to train a model and one event is used to validate the performance.

We define the overall objective function using cross-entropy loss, as can be seen in equation 13, where $i \in n$ samples, j are classes, y is the (one-hot) true label, and p is the probability output for each label. In multi-task training, the total loss is the sum of loss for stance learning task and rumor learning task. As shown in Fig. 3, Fig. 4 and Fig. 5, we use the output of the softmax layer for classifying stance and rumor labels of nodes in trees.

$$L(y, p) = -\frac{1}{n} \sum_{i,j} y_{ij} \log(p_{ij}) \qquad (13)$$

All operations in our models are fully differentiable, so these models can be trained end-to-end. Because the dataset has unbalanced labels, we can use over sampling of minority classes to create balanced input to train models. For rumor, balancing is easy as each tree has one rumor label, so we over-sample minority labeled trees to balance the training set. For stance labels, balancing is not trivial. The stance classes can be balanced by creating duplicate nodes of minority classes and connecting the new nodes to the original parent nodes. However, this results in changing the structure of trees. Thus we only used balancing on original conversation trees for stance classification and not for rumor classification on BCTrees.

Our LSTM models are built using PyTorch [5] and DGL library [6]. The Branch LSTM models used feature vectors as input, adds an LSTM layer, a linear dense activation layer followed by a dropout (0.3) (Srivastava et al., 2014) and uses a softmax layer for the output (rumor or stance). The models are trained using stochastic gradient descent (SGD) optimization using a cross-entropy loss function. The size of LSTM hidden layer and learning rate were used as hyper-parameter. The learning rate we tried were in range .0001 to 0.01. The LSTM layer size we tried varied from 16 to 256. We found 64 to be the best hidden dimension vector size and 0.08 to be a good learning rate for training the branch LSTMs. Once we find the best value for these hyper parameters by initial experiments, they remain unchanged during training and evaluations of the model for all five events.

The training of tree models also followed the same pattern except they use an entire tree conversation. The convolution units use convolution kernels of size 2 (i.e. it used two hidden vectors at time) and stride of 1. We tried learning rate from 0.001 to 0.1, and .008 was found to work the best. We again used stochastic gradient descent (SGD) optimization with a cross-entropy loss function. For multi-task training, we used step wise training that alternates between rumor objective and stance objective. We train the models for 30 epochs.

[3] https://github.com/ryankiros/skip-thoughts
[4] https://github.com/huggingface/torchMoji
[5] https://pytorch.org/
[6] https://www.dgl.ai

Model↓ Event →	CH	SS	FG	OS	GC	Mean F1
Majority	0.189	0.190	0.197	0.192	0.175	0.188
Branch LSTM Models						
GLOVE	0.332	0.322	0.298	0.305	0.385	0.329
BERT	0.384	0.393	0.332	0.380	0.425	0.383
SKP	0.424	0.417	0.373	0.454	0.455	0.425
EMT	0.370	0.332	0.365	0.399	0.442	0.381
SKPEMT	0.428	0.424	0.397	0.463	0.468	0.436
Tree LSTM Models - 'Child Sum' Cell Type						
BERT	0.512	**0.580**	0.528	0.481	0.522	0.524
SKP	0.490	0.565	0.540	0.495	**0.568**	0.532
EMT	0.443	0.514	0.444	0.453	0.509	0.473
SKPEMT	0.509	0.577	0.524	**0.504**	0.529	0.529
Tree LSTM Models - 'Child Convolve + MaxPooling' Cell Type						
BERT	0.510	0.564	0.522	0.476	0.530	0.520
SKP	**0.514**	0.579	**0.553**	0.469	0.547	**0.532**
EMT	0.486	0.478	0.530	0.439	0.496	0.486
SKPEMT	0.480	0.574	0.497	0.477	0.598	0.525
Prior Research						
(Zubiaga et al., 2018)	0.465	0.446	0.373	0.475	0.543	0.460
(Zubiaga et al., 2016a)	0.427	0.495	0.390	0.457	0.523	0.458
(Lukasik et al., 2016)	0.326	0.323	0.260	0.323	NA	NA

Table 3: Stance learning results: F1-score (macro) and mean of F1-macro (Mean-F1) for different events.

To evaluate the trained models, we use F1-score which is defined as the harmonic mean of precision and recall. Rather than using accuracy, we use F1-score as the metric for evaluating the performance of the models for two reasons: a) Pheme dataset (the dataset we use) is skewed towards one class ('comment'), hence, a classifier that predicts the majority class can get a good accuracy. F1-score (macro) balances the classes and considers precision as well as recall. 2) Prior work on this dataset used F1-score (Zubiaga et al., 2018). Thus, the use of this measure allows to compare with prior research. The performance for a validation event is the F1-macro obtained by evaluating the model trained on all data except the validation event data. This step is performed for all five events, and the mean of F1-macro scores from all five events is used to compare the models. For the stance classification task, the F1-score (macro) is defined in Eqn. 14. For the rumor classification task, the F1-score (macro) is defined in Eqn. 15.

$$F1_{stance} = \frac{F1_{deny} + F1_{favor} + F1_{query} + F1_{com.}}{4} \quad (14)$$

$$F1_{rumor} = \frac{F1_{true} + F1_{false} + F1_{unverified}}{3} \quad (15)$$

3.4 Stance Classification Results

We present the results of evaluating the models for stance classification in Tab. 3. The Tree LSTM model that uses 'Child Convolve + Maxpooling' with skipthought features outperforms all other models (0.532 mean f1). The Tree LSTM model using 'Child sum' unit performs equally well on mean value but was worse on three events.

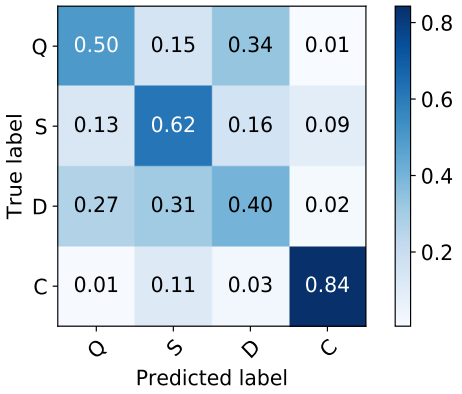

Figure 6: Normalized stance confusion matrix. Q, S, D and C labels indicate 'Query', 'Support', 'Deny' and 'Comments' respectively.

In Fig. 6, we show the confusion matrix for the best performing stance classifier. As we can observe, the model is best at classifying 'Comment' and is worst at classifying 'Denial'. The poor performance of the denial class could be partially attributed to the unbalance of classes ('Deny' being the smallest) in the dataset.

If we compare the stance classification results

based on feature types, we see that BERT and SKP are often comparable and EMT is slightly worse then them. SKPEMT performs better than EMT and BERT, but is as not as good as SKP. Because of space limitation, we do not present results for Glove features for Tree based models as, in almost all cases, the mean of Glove vectors as sentence representation performed worse than other features.

For stance learning, the BCTree based models did not work as well as the Tree LSTM based models. This is likely because we are not able to balance stance classes in BCT trees. BCTrees stance nodes can be balanced before binarizing, but that adds many additional new nodes. These new virtual nodes don't have stance labels and results in poor performance.

3.5 Rumor Classification Results

We present the rumor classification results in Table 4.

CellType ↓ Feature →	SKP	EMT	BERT	SKPEMT
Branch LSTM - Multitask				
	0.358	0.359	0.332	0.347
Tree LSTM - Multitask				
Sum	0.364	0.348	0.341	0.364
MaxPool	0.369	0.352	0.339	**0.375**
Convolve + MaxPool	**0.379**	**0.365**	**0.359**	0.370
BCTree LSTM - Multitask				
Sum	0.371	0.356	0.338	0.371
Convolve	0.367	0.335	0.337	0.362
Convolve+Sum	0.353	0.353	0.329	0.364
Convolve + Concat	0.370	0.354	0.340	0.364
MaxPool	0.353	0.354	0.326	0.352
Convolve+MaxPool	0.363	0.349	0.333	0.357
Concat + Sum	0.364	0.341	0.324	0.364
Convolve+Sum+Concat	0.366	0.343	0.342	0.354
Baselines and Prior Research				
(Kochkina et al., 2018)	0.329			
NileTMRG (Enayet and El-Beltagy, 2017)	0.339			
Majority	0.223			

Table 4: Rumor classification results: Mean F1-score from different cell-type and feature-type combinations. For NileTMRG, we used the results presented in (Kochkina et al., 2018), Tbl. 3.

For rumor classification, the best performing model uses 'Convolve + MaxPool' as units in Tree LSTM (Mean F1 of 0.379 using SKP features) and is trained in multi-task fashion. Other comparable models are 'sum' and 'Convolve + concat' units with BCTree LSTM. For SKPEMT features, the best performance was obtained using 'Maxpool' cell with a Tree LSTM model. We expected BCTree LSTM to work better than Tree LSTM. They are almost comparable but BCTree LSTM is slightly worse. This is likely because binarizing a tree creates many new nodes (without labels), and as height of trees increase it becomes more difficult for LSTMs to propagate useful information to the top root node for rumor-veracity classification.

If we compare the different types of features, SKP features outperformed others in almost all cases. It should be noted that SKP features are also higher in dimension (4800) in comparison to EMT 64 and BERT 768. If we compare, multi-task vs single-task, in almost all cases, performance improved by training in a multitask fashion.

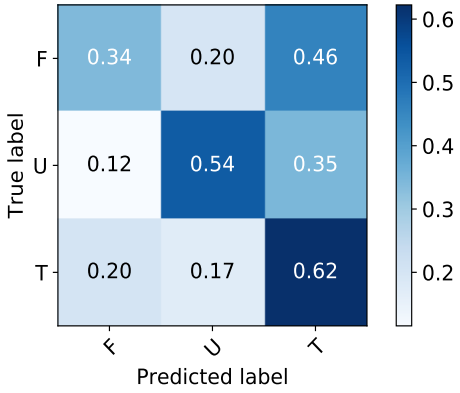

Figure 7: Normalized rumor confusion matrix. F, U and T labels indicate 'False', 'Unverified' and 'True' respectively.

Overall, for rumor classification, the best model is the LSTM model that uses 'Convolve + MaxPool' unit and trained on Tree LSTM using multi-task. This exceeds the best prior work by 12% in f1-score. For this model, we show the confusion matrix in Fig. 7. As we can observe, 'True' (T) and 'Unknown' (U) performs equally well and the 'False' (F) rumor is the most confusing class. The poor performance of 'False' rumors could be linked to the poor performance of 'Denials' stance in stance classification. Prior research have shown that a high number of denials is a good indicator of 'False' rumors, and therefore a model that is poor at predicting denials also performs poorly at predicting 'False' rumors.

4 Related Work

Stance learning and rumor detection lie at the intersection of many different fields. We highlight important related topics here.

4.1 Stance Learning

Computational approaches of Stance learning – which involves finding people's attitude about a topic of interest – have primarily appeared in two flavors. 1) Recognizing stance in debates (Somasundaran and Wiebe, 2010; Ozer et al., 2016) 2) Conversations on online social-media platforms. Since our research focuses on conversations on social-media platforms, we discuss some important contributions here. Mohammad et al. built a stance dataset using Tweets and organized a SemEval competition in 2016 (Task 6). Many researchers (Augenstein et al., 2016; Liu et al., 2016; Wei et al., 2016) used the dataset and proposed algorithms to learn stance from this text data. In almost the same time frame, work on stance in conversations appeared in the context of fake-news and misinformation identification, we discuss this in the next section.

4.2 Rumor and Misinformation Identification

Finding misinformation on social-media platforms has been an active area of research in recent years (Hassan et al., 2015; Lukasik et al., 2015; Dang et al., 2016; Volkova et al., 2017; Zubiaga et al., 2018; Zhou et al., 2019; Sharma et al., 2019). Rumor detection that uses stance in the reply posts was in initiated by the Pheme project [7] and was popularized as a SemEval 2017 task 8 [8]. The task involved predicting stance ('supporting', 'denying', 'commenting' and 'querying') in replies to rumor posts on Twitter and the dataset is described in (Zubiaga et al., 2015, 2016b). A number of researchers used this dataset and proposed many algorithms. For example, (Derczynski et al., 2017) proposed an LSTM that uses branches in conversation trees to classify stance in reply posts, and (Kochkina et al., 2018) used sequential classifiers for joint stance and rumor classification. More recently (Ma et al., 2018) suggested two tree structured neural-networks to find rumors i.e. if a post is rumor or not. In this work, we focus on rumor-veracity and stance learning objectives. Our work extends this thread of research by showing that convolution operations that compare source and reply tweets are more effective in learning stance and rumor-veracity.

4.3 LSTM and Convolutional Neural Networks

Deep neural networks (DNN) have shown great success in many fields (Hinton et al., 2012). Researchers have used DNNs for various NLP tasks like POS tagging, named entity recognition (Collobert and Weston, 2008). Convolution neural networks (LeCun et al., 2010) are popular in computer vision tasks for quite some time but lately they have shown potential in NLP tasks as well (Zhang et al., 2015). Yoon Kim (Kim, 2014) used convolution neural networks (CNN) for various NLP tasks. To the best of our knowledge, this is the first work that uses a convolution unit in LSTMs.

5 Conclusion

In this work, we explored a few variants of LSTM cells for rumor-veracity and stance learning tasks in social-media conversations. We also proposed a new Binarized Constituency Tree structure to model social-media conversations. Using a human labeled dataset with rumor-veracity labels for source posts and stance labels for replies, we evaluated the proposed models and compared their strengths and weaknesses. We find that using convolution unit in LSTMs is useful for both stance and rumor classification. We also experimented with different types of features and find that skipthoughts and BERT are competitive features while skipthoughts have slight advantage for rumor-veracity prediction task.

Acknowledgments

We are thankful to anonymous reviewers for their valuable feedback. This work was supported in part by the ONR Award No. N00014182106, ONR Award No. N0001418SB001 and the Center for Computational Analysis of Social and Organization Systems (CASOS). The views and conclusions contained in this document are those of the authors only. Funding to attend this conference was partly provided by the CMU GSA/Provost Conference funding.

[7] https://www.pheme.eu/

[8] http://www.aclweb.org/anthology/S17-2006

References

Hunt Allcott and Matthew Gentzkow. 2017. Social media and fake news in the 2016 election. *Journal of Economic Perspectives*, 31(2) 211-36.

Isabelle Augenstein, Andreas Vlachos, and Kalina Bontcheva. 2016. Usfd at semeval-2016 task 6: Any-target stance detection on twitter with autoencoders. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 389–393.

Matthew Babcock, Ramon Alfonso Villa Cox, and Sumeet Kumar. 2019. Diffusion of pro- and anti-false information tweets: the black panther movie case. *Computational and Mathematical Organization Theory*, 25(1):72–84.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM.

Anh Dang, Michael Smit, Abidalrahman Moh'd, Rosane Minghim, and Evangelos Milios. 2016. Toward understanding how users respond to rumours in social media. In *2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 777–784. IEEE.

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. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 69–76, Vancouver, Canada. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Douglas Eck and Juergen Schmidhuber. 2002. Finding temporal structure in music: Blues improvisation with lstm recurrent networks. In *Neural Networks for Signal Processing, 2002. Proceedings of the 2002 12th IEEE Workshop on*, pages 747–756. IEEE.

Omar Enayet and Samhaa R El-Beltagy. 2017. Niletmrg at semeval-2017 task 8: Determining rumour and veracity support for rumours on twitter. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 470–474.

Bjarke Felbo, Alan Mislove, Anders Søgaard, Iyad Rahwan, and Sune Lehmann. 2017. Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Emilio Ferrara. 2015. Manipulation and abuse on social media. *ACM SIGWEB Newsletter*, (Spring):4.

Daniel Gildea. 2004. Dependencies vs. constituents for tree-based alignment. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*.

Naeemul Hassan, Chengkai Li, and Mark Tremayne. 2015. Detecting check-worthy factual claims in presidential debates. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 1835–1838. ACM.

Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. 2012. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Zhiwei Jin, Juan Cao, Yongdong Zhang, and Jiebo Luo. 2016. News verification by exploiting conflicting social viewpoints in microblogs. In *AAAI*, pages 2972–2978.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar. Association for Computational Linguistics.

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *Advances in neural information processing systems*, pages 3294–3302.

Elena Kochkina, Maria Liakata, and Arkaitz Zubiaga. 2018. All-in-one: Multi-task learning for rumour verification. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3402–3413. Association for Computational Linguistics.

Yann LeCun, Koray Kavukcuoglu, and Clément Farabet. 2010. Convolutional networks and applications in vision. In *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*, pages 253–256. IEEE.

Jiwei Li, Thang Luong, Dan Jurafsky, and Eduard Hovy. 2015. When are tree structures necessary for deep learning of representations? In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2304–2314, Lisbon, Portugal. Association for Computational Linguistics.

Can Liu, Wen Li, Bradford Demarest, Yue Chen, Sara Couture, Daniel Dakota, Nikita Haduong, Noah Kaufman, Andrew Lamont, Manan Pancholi, et al. 2016. Iucl at semeval-2016 task 6: An ensemble model for stance detection in twitter. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 394–400.

Michal Lukasik, Trevor Cohn, and Kalina Bontcheva. 2015. Classifying tweet level judgements of rumours in social media. In *EMNLP*.

Michal Lukasik, PK Srijith, Duy Vu, Kalina Bontcheva, Arkaitz Zubiaga, and Trevor Cohn. 2016. Hawkes processes for continuous time sequence classification: an application to rumour stance classification in twitter. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 393–398.

Jing Ma, Wei Gao, and Kam-Fai Wong. 2018. Rumor detection on twitter with tree-structured recursive neural networks. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1980–1989, Melbourne, Australia. Association for Computational Linguistics.

Saif M Mohammad, Parinaz Sobhani, and Svetlana Kiritchenko. 2017. Stance and sentiment in tweets. *ACM Transactions on Internet Technology (TOIT)*, 17(3):26.

Mert Ozer, Nyunsu Kim, and Hasan Davulcu. 2016. Community detection in political twitter networks using nonnegative matrix factorization methods. In *Advances in Social Networks Analysis and Mining (ASONAM), 2016 IEEE/ACM International Conference on*, pages 81–88. IEEE.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Victoria Rubin, Niall Conroy, Yimin Chen, and Sarah Cornwell. 2016. Fake news or truth? using satirical cues to detect potentially misleading news. In *Proceedings of the Second Workshop on Computational Approaches to Deception Detection*, pages 7–17.

Victoria L Rubin and Tatiana Lukoianova. 2015. Truth and deception at the rhetorical structure level. *Journal of the Association for Information Science and Technology*, 66(5):905–917.

Steve Schifferes, Nic Newman, Neil Thurman, David Corney, Ayse Göker, and Carlos Martin. 2014. Identifying and verifying news through social media: Developing a user-centred tool for professional journalists. *Digital Journalism*, 2(3):406–418.

Karishma Sharma, Feng Qian, He Jiang, Natali Ruchansky, Ming Zhang, and Yan Liu. 2019. Combating fake news: A survey on identification and mitigation techniques. *ACM Trans. Intell. Syst. Technol.*, 10(3):21:1–21:42.

Swapna Somasundaran and Janyce Wiebe. 2010. Recognizing stances in ideological on-line debates. In *Proceedings of the NAACL HLT 2010 Workshop on Computational Approaches to Analysis and Generation of Emotion in Text*, pages 116–124. Association for Computational Linguistics.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Eugenio Tacchini, Gabriele Ballarin, Marco L. Della Vedova, Stefano Moret, and Luca de Alfaro. 2017. Some like it hoax: Automated fake news detection in social networks. *CoRR*, abs/1704.07506.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1556–1566, Beijing, China. Association for Computational Linguistics.

Svitlana Volkova, Kyle Shaffer, Jin Yea Jang, and Nathan Hodas. 2017. Separating facts from fiction: Linguistic models to classify suspicious and trusted news posts on twitter. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 647–653.

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

Wei Wang, Kevin Knight, and Daniel Marcu. 2007. Binarizing syntax trees to improve syntax-based machine translation accuracy. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*.

Wan Wei, Xiao Zhang, Xuqin Liu, Wei Chen, and Tengjiao Wang. 2016. pkudblab at semeval-2016 task 6: A specific convolutional neural network system for effective stance detection. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 384–388.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume*

1 (Long Papers), pages 1112–1122, New Orleans, Louisiana. Association for Computational Linguistics.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pages 649–657.

Kaimin Zhou, Chang Shu, Binyang Li, and Jey Han Lau. 2019. Early rumour detection. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1614–1623, Minneapolis, Minnesota. Association for Computational Linguistics.

Arkaitz Zubiaga, Elena Kochkina, Maria Liakata, Rob Procter, and Michal Lukasik. 2016a. Stance classification in rumours as a sequential task exploiting the tree structure of social media conversations. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2438–2448, Osaka, Japan. The COLING 2016 Organizing Committee.

Arkaitz Zubiaga, Elena Kochkina, Maria Liakata, Rob Procter, Michal Lukasik, Kalina Bontcheva, Trevor Cohn, and Isabelle Augenstein. 2018. Discourse-aware rumour stance classification in social media using sequential classifiers. *Information Processing & Management*, 54(2):273–290.

Arkaitz Zubiaga, Maria Liakata, Rob Procter, Kalina Bontcheva, and Peter Tolmie. 2015. Crowdsourcing the annotation of rumourous conversations in social media. In *Proceedings of the 24th International Conference on World Wide Web*, pages 347–353. ACM.

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