DUCK: Rumour Detection on Social Media by Modelling User and Comment Propagation Networks

Anonymous ACL submission

Abstract

Social media rumours, a form of misinformation, can mislead the public and cause significant economic and social disruption. Motivated by the observation that the user network — which captures *who* engage with a story — and the comment network — which captures *how* they react to it — provide complementary signals for rumour detection, in this paper, we propose DUCK (rumour detection with user and comment networks) for rumour detection on social media. We study how to leverage transformers and graph attention networks to jointly model the contents and structure of social media conversations, as well as the network of users who engaged in these conversations. Over four widely used benchmark rumour datasets in English and Chinese, we show that DUCK produces superior performance for detecting rumours, creating a new state-of-the-art. Source code for DUCK is available at: ANONYMISED.

1 Introduction

Social media platforms bring easy access to the wealth of information. On the flip side, social media has also accelerated the spread of misinformation (Starbird et al., 2014; Jin et al., 2017). Rumours, a form of misinformation typically defined as stories or statements with unverified truth value (Allport and Postman, 1947), can mislead the public and cause significant economic and social disruption.

Since the seminal work on prediction of information credibility on social media by Castillo et al. (2011), automatic rumour detection on social media has attracted significant research, which aims to detect rumorous posts (in contrast to news articles by credible news sources) or determine the veracity — true, false or unverified — of rumours. Although the task is related to fake news detection, the use of social media for propagation means that various social context features can be leveraged for detection. This is a contrast to FEVER-style fake news detection (Thorne et al., 2018) which relies mainly on a source of world knowledge (e.g. Wikipedia) to fact-check stories.

Early studies of rumour detection focus on supervised learning algorithms incorporating features manually engineered from post contents, user profiles as well as information propagation patterns (Castillo et al., 2011; Liu et al., 2015; Kwon et al., 2013; Ma et al., 2015; Rath et al., 2017). Recent neural approaches typically explore fusing different feature representations for rumour detection. Sequence processing models such as BERT are used to encode the contents of social media conversations, e.g. source posts and the stream of comments (Kochkina et al., 2017; Tian et al., 2020), while graph models have been experimented to model the structure of social media conversations (Bian et al., 2020; Ma et al., 2018; Lu and Li, 2020). Although some approaches use a combination of content and user features for rumour detection, how to leverage pretrained sequence models and graph models to effectively model them remains under-explored.

In this paper we propose DUCK (rumour detection with user and comment networks), a framework that jointly models the user and comment propagation networks. Our study presents an extensive exploration on how we can best model these networks, and compared to previous studies, there are several key differences: (1) we model comments both as a: (i) stream to capture the temporal nature of evolving comments; and (ii) network by following the conversational structure (see Figure 1 for an illustration); (2) our comment network uses sequence model to encode a pair of comments before feeding them to a graph network, allowing our model to capture the nuanced characteristics (e.g. agreement or rebuttal) exhibited by a reply; and (3) when modelling the users who en-

gage with a story via graph networks, we initialise the user nodes with encodings learned from their profiles *and* characteristics of their "friends" based on their social networks.

We conduct experiments on four widely used benchmark rumour datasets in English and Chinese, and show that DUCK produces superior performance, creating a new state-of-the-art. Although both users and comments provide complementary signals for our task, the comments have a stronger impact. Also, when modelling the network of users who engage with a story, incorporating the social relations of users proves to be very beneficial. Source code for DUCK is available at: ANONYMISED.

2 Related Work

Early studies of rumour detection focus on supervised learning algorithms incorporating engineered features from post contents, user profiles as well as information propagation patterns (Castillo et al., 2011; Liu et al., 2015; Kwon et al., 2013; Ma et al., 2015; Rath et al., 2017).

Recent research focus on neural models to automatically generate various features for rumour detection. Sequence processing models leverage the textual contents from the source posts and user reply comments for rumour detection. Signals such as writing style, stance and opinions as well as emotions are extracted from the text for rumour detection. Shu et al. (2017) introduce linguistic features to represent writing styles and other features based on sensational headlines from Twitter to detect misinformation. To detect rumours as early as possible, Zhou et al. (2019) incorporate reinforcement learning to dynamically decide how many responses are needed to classify a rumour. Tian et al. (2020) explore the relationship between a source tweet and its comments by transferring stance prediction model to classify rumours. Most of these approaches model user comments as a sequence of posts and ignore the conversational structure among the comments.

Graph neural models leverage information propagation patterns for rumour detection. Liu and Wu (2018) experiment with using convolutional and recurrent neural networks to process user features in the retweet propagation path of stories, and Ma et al. (2018) present a tree-structure recursive neural network to model information propagation for rumour detection. Bian et al. (2020) proposed a bi-directional graph network to model the upward and downward information propagation structure among user comments to distinguish false from true rumours.

There are also studies combining signals from contents, users and propagation networks for rumour and fake news detection (Lu and Li, 2020; Nguyen et al., 2020). An ensemble deep learning architecture is presented in Lu and Li (2020), which incorporates source post content and retweet network. Nguyen et al. (2020) propose to learn representations for misinformation detection based on the heterogeneous graph of news, news sources, users and their stances in comments. All these studies largely model the superficial characteristics of comments and users, e.g. comments are represented using static features such as bag-of-words (Bian et al., 2020; Nguyen et al., 2020) and users with simple features extracted from their profiles (Liu and Wu, 2018; Lu and Li, 2020). Deeper interactions, such as the relation between a post and its reply and the social relations (e.g. "following") of users, remain under-explored. Table 1 summarises the differences between previous studies and our work.

Beyond rumour detection, recent studies explore combining modern pretrained language models and graph models for modelling texts and their interactions. Using the FEVER dataset, Zhong et al. (2020) explore pretrained models to perform semantic role labelling to understand the relation between clauses in evidence passages and then encode the network with graph models to detect fake news. Liu et al. (2020) use BERT to encode a pair of claim and evidence passage and then propose a kernel graph network to model the fully connected network of evidence passages. Although these two studies combine sequence and graph models, their task has a different data structure and hence their methods cannot be trivially adapted to the rumour detection task.

3 Problem Statement

Let $X = \{x_0, x_1, x_2, ..., x_n\}$ be a set of stories, where a story x_i consists of a source post and its reply comments, defined as $x_i = \{(c_0, u_0, p_0, t_0), ..., (c_m, u_m, p_m, t_m)\}$, where c refers to the textual content of the post (empty string if it's a repost/retweet), u the user ID who made the post, p the parent post ID that the current post replies to (null if it's a source post, e.g. $p_0 =$ null), and t the timestamp of the post. Each story

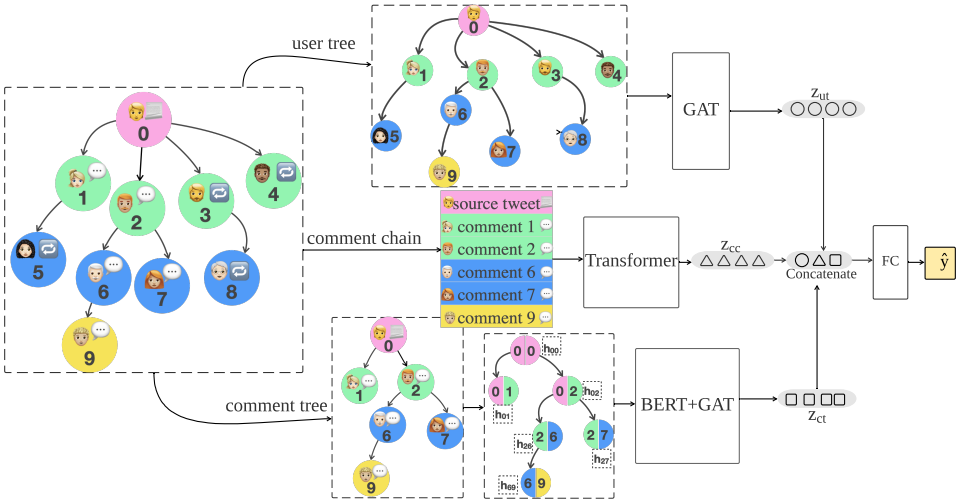


Figure 1: Overall architecture of DUCK. The structure of user tree differs from that of comment tree, as the former captures both comments (💬) and reposts/retweets (🔁), while the latter considers only comments.

	S	C	CN	UF	UN
RvNN (Ma et al., 2018)	✓	✓	✓		
RNN+CNN (Liu and Wu, 2018)				p	rt
Multitask (Li et al., 2019)	✓	✓	✓	p	
stance-BERT (Tian et al., 2020)	✓	✓			
Bi-GCN (Bian et al., 2020)	✓	✓	✓		
GCAN (Lu and Li, 2020)	✓			p	rt
DUCK (our work)	✓	✓	✓	p+s	rt+rp

Table 1: Information used in various studies. S: source post, C: comments, CN: comment network, UF: user features, UN: user network, p: user profile, s: social relations, rt: repost/retweet, and rp: reply.

x_i is associated with a ground-truth label $y_i \in Y$, where Y represents the label set (binary or 4 classes depending on the rumour dataset). Our goal is to learn a classifier from the labelled rumour dataset, that is $f : X \rightarrow Y$.

4 Methodology

The overall architecture of our rumour detection approach is presented in Figure 1. It consists of four modules: (1) **comment tree**: models the comment network by following the reply-to structure using a combination of BERT (Devlin et al., 2019) and graph attentional networks (Veličković et al., 2017); (2) **comment chain**: models the comments as a stream using transformer-based sequence models; (3) **user tree**: incorporates social relations to model the user network using graph attentional networks; (4) **rumour classifier**: combines the output from comment tree, comment chain and user tree to classify the source post. Note that the network structure of the user tree differs from that of the comment tree as the former captures both comments and reposts/retweets but the latter considers only comments (Figure 1).

4.1 Comment Tree

Here we aim to model the conversational structure of the comments that a source post generates. Previous studies typically model this via graph networks, but most use simple features (e.g. bag-of-words) to represent the text (Bian et al., 2020), which fail to capture the nuanced relationships (e.g. agreement) between a parent post and its child/reply post.

To capture the relations of crowd opinions in the comment tree, we propose to use a pretrained language model (BERT; (Devlin et al., 2019)) and graph attention network (GAT; (Vieweg et al., 2010)) to model comment tree; see Figure 2 for an illustration. We first process the set of parent-child posts with BERT (Devlin et al., 2019) before feeding them to a graph network to model the full conversational structure. The self-attention mechanism between the words in the parent and child posts would produce a more fine-grained analysis of their relationship, which representations such as bag-of-words cannot model. Using the comment tree in Figure 2 as an example, this means we would first process the following pairs of posts using BERT: $\{(0,0),(0,1),(0,2),(2,6),(2,7),(6,9)\}$, where $(0,0)$ is a pseudo pair created for the source

post.[1] Formally:

$$h_{p+q} = \text{BERT}(\text{emb}([CLS], c_p, [SEP], c_q)) \quad (1)$$

where c represents the text, emb() the embedding function and h the contextual representation of the $[CLS]$ token produced by BERT.

To model the conversational network structure, we use graph attentional networks (GAT; (Veličković et al., 2017)). Different to graph convolutional networks (Kipf and Welling, 2016a), GAT iteratively learns node encodings via multi-head attention with neighbouring nodes, and has the advantage to infer encodings for new nodes after it's trained. To compute $h_i^{(l+1)}$, the encoding for node i at iteration $l+1$:

$$e_{ij}^{(l)} = \text{LR}\left(a^{(l)^T}\left(W^{(l)}h_i^{(l)} \oplus W^{(l)}h_j^{(l)}\right)\right)$$
$$h_i^{(l+1)} = \sigma\left(\sum_{j\in\mathcal{N}(i)} \text{softmax}\left(e_{ij}^{(l)}\right) z_j^{(l)}\right) \quad (2)$$

where LR denotes the LeakyReLU activation function, $\oplus$ the concatenation operation, $\mathcal{N}(i)$ the neighbours of node i, $e_{ij}^{(l)}$ the unnormalised attention score between node i and j, and a and W are learnable parameters. Note that $h_i^{(0)}$ represents the encodings produced by BERT (Equation 1).

To aggregate the node encodings to get a graph representation (z_{ct}), we explore four methods:

root: Uses the root encoding to represent the graph as the source post is ultimately what we are classifying:

$$z_{ct} = h_0^L \quad (3)$$

where L is the number of GAT iterations.

¬**root**: Mean-pooling over all nodes except the root:

$$z_{ct} = \frac{1}{m}\sum_{i=1}^{m} h_i^L \quad (4)$$

where m is the number of replies/comments.

▲: Mean-pooling of the root node and its immediate neighbours:

$$z_{ct} = \frac{1}{|\mathcal{N}(0)|}\sum_{i\in\mathcal{N}(0)} h_i^L \quad (5)$$

all: Mean-pooling of all nodes:

$$z_{ct} = \frac{1}{m+1}\sum_{i=0}^{m} h_i^L \quad (6)$$

[1]Preliminary experiments found that the pseudo pair is important because it allows us to maintain the original network structure.

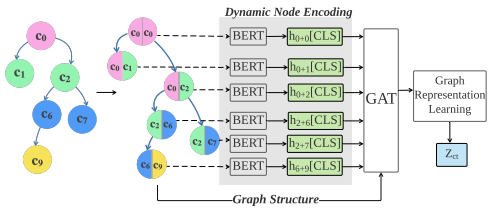


Figure 2: The architecture of BERT+GAT.

4.2 Comment Chain

Here we model the posts as a stream in the order they are posted. As such, we have a chain or list structure (rather than a tree structure); see "comment chain" in Figure 1.

We explore three ways to model the comment chain, using: (1) one-tier transformer; (2) longformer (Beltagy et al., 2020); and (3) two-tier transformer.

One-tier transformer: Given a source post (c_0) and the comments ($\{c_1, ..., c_m\}$), we simply concatenate them into a long string and feed it to BERT and use the contextual representation of the $[CLS]$ token as the final representation:

$$z_{cc} = \text{BERT}(\text{emb}([CLS], c_0, [SEP], c_1, ..., c_{m'}))$$

where m' ($< m$) is the number of comments we can incorporate without exceeding BERT's maximum sequence length (384 in our experiments).

Longformer: To circumvent the sequence length limit, we experiment with using a longformer, which can process up to 4,096 subwords, allowing us to use most if not all the comments. Longformer has a similar architecture as the one-tier transformer, but uses a sparser attention pattern to process longer sequences more efficiently. We use a pretrained longformer[2], and follow the same approach as before for modelling the comment chain:

$$z_{cc} = \text{LF}(\text{emb}([CLS], c_0, [SEP], c_1, ..., c_{m''}))$$

where $m'' \approx m$.

Two-tier transformer: An alternative approach to tackling the sequence length limit is to model the comment chain using two tiers of transformers: one for processing the posts independently, and another for processing the sequence of posts using representations from the first transformer. Formally:

$$h_i = \text{BERT}(\text{emb}_1([CLS], c_i))$$
$$z_{cc} = \text{transformer}(\text{emb}_2([CLS]), h_0, h_1, ..., h_m)$$

[2]https://huggingface.co/transformers/model_doc/longformer.html

where BERT and transformer denote the first- and second-tier transformers respectively. The second-tier transformer has a similar architecture to BERT, but has only 2 layers and its parameters are initialised randomly.

4.3 User Tree

Moving away from the post content, here we model the network of users that interact with a source post ("user tree" in Figure 1). Previous studies found that the user characteristics are different for those that engage with rumours vs. those who don't (Vosoughi et al., 2018; Shu et al., 2018), motivating us to model the user network. Note that unlike previous studies, our user network captures all users who reply to or repost the source post (previous studies use only the reposts, see Table 1).

We explore three methods to model the user network. All methods use a GAT (Veličković et al., 2017) to model the network (following Equation 2), and we aggregate the node encodings by mean-pooling over all nodes to produce the graph representation:

$$z_{ut} = \frac{1}{m+1}\sum_{i=0}^{m} h_i^L$$

where L is number of GAT iterations.

The main difference between the three methods is in how they initialise the user nodes ($h_i^{(0)}$):

GAT$_{\text{rnd}}$: This is the base method where we initialise the user nodes with random vectors.

GAT$_{\text{prf}}$: Following Liu and Wu (2018), this method initialises the user nodes based on features derived from their user profiles: username, user screen name, user description, user account age, number of followers and following users, number of posts and favourite posts, whether the profile is protected, whether the account is GPS-enabled, and the time difference with the source post.[3]

GAT$_{\text{prf+rel}}$: This method initialises the user nodes with representations learned by a variational graph autoencoder (GAE; Kipf and Welling (2016b)) based on the user features (defined above) *and* their social relations (based on "follow" relations).[4] Intuitively, GAE is an unsupervised graph learning algorithm that takes in an adjacency matrix as input, and learns node representations that can reconstruct the adjacency matrix in the output. Note that the network structure of the GAT and GAE is intrinsically different — the former captures the users that engage with a source post while the latter the network of users who follow one another. The idea for using GAE-learned encodings to initialise user nodes is that they are more informative, since they capture information about a user and their peers.

[3]The last feature is technically not user profile information, but it is a form of user characteristic as it captures how quickly they engage with a post.

[4]For the unseen or isolated users, we initialise them based on their user features (used in GAT$_{\text{prf}}$), and project them via a learned matrix into the same space as the GAE-initialised user nodes.

	Twitter15	Twitter16	WEIBO	CoAID
#source nodes	1,490	818	4,664	143,009
#users	276,663	173,487	2,746,818	114,484
Comment graph				
#nodes	331,612	204,820	3,805,656	248,742
Avg. # of nodes/s	223	251	816	7
Max. # of nodes/s	1,768	2,765	1,768	228
Min. # of nodes/s	55	81	10	1
Avg. time delay/s	1,337	848	2460.7	15.4
User social network				
#nodes	39,869	19,211	2,746,818	1,601
#connections	3,086,741	1,232,100	–	25,605

Table 2: Statistics of rumour datasets. "s" denotes an story (source post and its comments).

4.4 Rumour Classifier

In each module (comment tree, comment chain and user tree), we explore a number of approaches to model its structure (e.g. there are several ways to aggregate the node encodings to produce z_{ct} for the comment tree and 3 different methods to produce z_{cc} for the comment chain). Given an optimal approach for each module (discussed in the Experiments and Results section), DUCK (Figure 1) combines the output from all modules to classify a source post and is trained using standard cross-entropy loss. Formally:

$$z = z_{ct} \oplus z_{cc} \oplus z_{ut}$$
$$\hat{y} = \text{softmax}(W_c z + b_c)$$
$$\mathcal{L} = -\sum_{i=1}^{n} y_i log(\hat{y}_i)$$

where n denotes the number of instances.

5 Experiments and Results

5.1 Datasets and models

We evaluate our method on four widely used rumour datasets: Twitter15 (Ma et al., 2017); Twitter16 (Ma et al., 2017); CoAID (Cui and Lee,

2020); and WEIBO (Ma et al., 2016). Twitter15 and Twitter16 are Twitter datasets with four rumour classes: true rumour, false rumour, non-rumour and unverified rumour. CoAID (Cui and Lee, 2020) collects of a set of COVID-19 news articles shared on Twitter, and they are annotated with two classes (true or fake). WEIBO (Ma et al., 2016) contains a collection of stories from Sina Weibo, a Chinese social media platform, and is annotated with two classes (rumour and non-rumour). Table 2 shows statistics of these datasets. For Twitter-based datasets (Twitter15/16 and CoAID), we crawl the tweets and additional user information (e.g. user profile metadata and followers) via the official Twitter API.[5] For WEIBO, the platform does not provide a means to crawl social relations them and so the user tree uses GAT_{prf}.

In terms of data partitioning, for Twitter15 and Twitter16 we follow previous studies (Ma et al., 2015, 2016) and report the average performance based on 5-fold cross-validation. For CoAID and WEIBO, we reserve 20% data as test and split the rest in a ratio of 4:1 for training and development partitions and report the average test performance over 5 runs (initialised with different random seeds). We use the development set of each dataset for tuning hyper-parameters.[6]

Implementation details for all models are given in the Appendix.

5.2 Results

In this section, we first present results for each of the modules (comment tree, comment chain and user tree) *separately* to understand the best approach for modelling them, and then present the final results where we compare our full model DUCK (Figure 1) to a number of benchmark systems. For the first set of results where we evaluate each module independently, we feed their representations (i.e. z_{ct}, z_{cc} and z_{ut}) to an MLP layer to do classification. Specifically, we are interested in the following questions:

- Q1 [Comment tree]: Does incorporating BERT to analyse the relation between parent and child posts help modelling the comment network, and what is the best way to aggregate comment-pair encodings to represent the comment graph?
- Q2 [Comment chain]: Does incorporating more comments help rumour detection when modelling them as a stream of posts?
- Q3 [User tree]: Can social relations help modelling the user network?
- Q4 [Overall performance]: Do the three different components complement each other and how does a combined approach compared to existing rumour detection systems?

For the first three questions, we present *development* performance using Twitter16 and CoAID as the representative datasets (as the trends are largely the same for the other datasets), while for the final question we report the *test* performance for all datasets. In terms of evaluation metrics, we present F1 scores for each class and macro-averaged F1 scores as the aggregate performance. All results are an average over 5 runs (5-fold cross-validation for Twitter15/16 and 5 independent runs with different random seeds for WEIBO and CoAID following Ma et al. (2016, 2017); Cui and Lee (2020)).

5.2.1 Comment Tree

To understand the impact of using BERT for processing a pair of parent-child posts, we present an alternative method ("unpaired") where we use BERT to process each post independently before feeding their $[CLS]$ representation to the GAT. That is, Equation 1 is now modified to:

$$h_p = \text{BERT}(\text{emb}([CLS], c_p))$$

where h will be used as the initial node representation ($h^{(0)}$) in the GAT (Equation 2). We report the performance of this alternative model ("unpaired")[7] and the different aggregation methods ("root", "¬root", "▲" and "all"; equations 3, 4, 5 and 6 respectively) in Table 3.

Comparing the aggregation methods, "all" performs the best, followed by "▲" and "root" (0.88 vs. 0.87 vs. 0.86 in Twitter16; 0.87 vs. 0.86 vs. 0.85 in CoAID in terms of Macro-F1). We can see that the root and its immediate neighbours contain most of the information, and not including the root node impacts the performance severely (both Twitter16 and CoAID drops to 0.80 with ¬root). Does processing

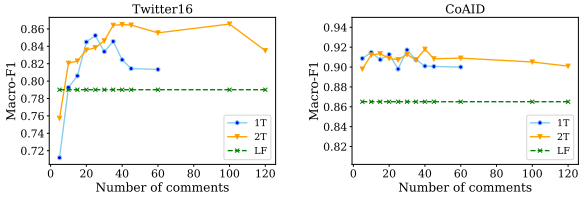
[5] https://developer.twitter.com/en/docs/twitter-api/v1

[6] For Twitter15/16 during tuning we use only one of the folds and reserve 1/4 of the training data as development set and train the model using the rest (3/4) of the training data.

[7] The "unpaired" approach uses "all" as the aggregation method.

Variants	Twitter16					CoAID		
	F1	FR	TR	NR	UR	F1	T	F
unpaired	0.83	**0.92**	0.87	0.73	0.82	0.83	0.98	0.67
root	0.86	0.85	0.92	**0.85**	0.83	0.85	0.98	0.71
¬root	0.80	0.82	0.91	0.77	0.79	0.80	0.97	0.64
▲	0.87	0.89	**0.95**	0.74	0.88	0.86	**0.99**	0.74
all	**0.88**	0.89	0.94	0.79	**0.90**	**0.87**	0.98	**0.75**

Table 3: Results for the comment tree. "FR", "TR", "NR" and "UR" in Twitter16 denote false, true, non- and unverified rumours respectively; and "T" and "F" in CoAID means true and fake classes.



(a) Twitter16 (b) CoAID

Figure 3: Results (macro-F1) for the comment chain over varying number of comments.

the parent-child posts together with BERT help? The answer is evidently yes, as we see a substantial drop in performance when we process the posts independently: "unpaired" produces a macro-F1 of only 0.83 in both Twitter16 and CoAID. Given these results, our full model (DUCK) will be using "all" as the aggregation method for computing the comment graph representation.

5.2.2 Comment Chain

Recall that we explore using transformer models – one-tier transformer, longformer and two-tier transformer – for modelling the comments of a story as a sequence. Fig. 3 plots the results where we vary the number of included comments to answer Q2.[8] Note that for longformer we always use all the comments, since it is designed to process long sequences.

Both one-tier and two-tier transformers see a performance gain when the number of comments increases and a drop when there are too many comments (noting that the trend is flatter in CoAID). However, due to one-tier transformer's sequence length limit, it can take no more than 60 comments on average. Two-tier transformer is able to process more comments, and produces the best performance. Interestingly, even though longformer is able to include most of the comments, it performs worse than both one-tier and two-tier transformer, suggesting that the sparser attention pattern that longformer introduces has a negative impact. With these results, we will use the two-tier transformer to model the comment chain in DUCK.

[8]For one-tier and two-tier transformers, if the number of comments is set to 10, that means we will concatenate 10 comments (with the source post) into a long string, and any text that exceeds BERT's maximum sequence length will be truncated (and so for some stories the models may use less than 10 comments, if earlier comments are very long).

Variants	Twitter16					CoAID		
	F1	FR	TR	NR	UR	F1	T	F
GAT_{rnd}	0.47	0.57	0.38	0.48	0.47	0.61	0.59	0.46
GAT_{prf}	0.63	0.64	0.67	0.56	0.60	0.80	0.97	0.62
$GAT_{prf+rel}$	**0.69**	**0.74**	**0.72**	**0.64**	**0.68**	**0.85**	**0.98**	**0.71**

Table 4: Results for the user tree.

5.2.3 User Tree

Recall that we use GAT to represent the reply and repost user network, and we investigate different node encodings to initialise GAT: (1) random initialisation (GAT_{rnd}); (2) user profile metadata (GAT_{prf}); and (3) user profile metadata and social relations ($GAT_{prf+rel}$). Results are shown in Table 4. Unsurprisingly, random initialisation performs the worse, and we see a substantial improvement when user profile information is incorporated, and again an improvement when we incorporate user social relations (6% and 5% increase in macro-F1 for Twitter16 and CoAID). Our results highlight the importance of incorporating social relations, and DUCK therefore uses $GAT_{prf+rel}$ for modelling the reply and retweet user network.[9]

5.2.4 Overall Rumour Detection Performance

We next compare the rumour detection performance of DUCK that combines comment tree, comment chain and user tree models (Figure 1) to the following state-of-the-art methods: (1) **RvNN** (Ma et al., 2018)[10]: uses a GRU to process text content and recursive networks to model the comment network; (2) **RNN+CNN** (Liu and Wu, 2018): uses CNN and RNN to model the retweet user network where user representations are initialised with user profile features; (3) **stance-BERT** (Tian et al.,

[9]With the exception of WEIBO where we can't crawl users' followers, and so it uses GAT_{prf}.

[10]https://github.com/majingCUHK/Rumor_RvNN

	Twitter15					Twitter16					CoAID			WEIBO		
Model	F1	FR	TR	NR	UR	F1	FR	TR	NR	UR	F1	T	F	F1	NR	R
RvNN	0.72	0.76	0.82	0.68	0.65	0.74	0.74	0.84	0.66	0.71	0.78	0.98	0.57	0.91	0.91	0.91
RNN+CNN	0.53	0.51	0.30	0.36	0.64	0.56	0.54	0.40	0.59	0.67	–	–	–	0.92	0.91	0.92
stance-BERT	0.82	0.82	0.85	0.87	0.71	0.83	0.82	0.88	0.83	0.77	0.90	**0.99**	0.81	–	–	–
Bi-GCN	0.86	0.85	0.91	0.84	0.82	0.86	0.86	**0.93**	0.79	0.86	0.83	**0.99**	0.68	0.96	0.96	0.96
GCAN	0.69	0.75	0.75	0.63	0.68	0.72	0.73	0.78	0.67	0.72	–	–	–	0.92	0.92	0.92
DUCK$_{\neg CT}$	0.82	0.72	0.91	0.82	0.85	0.84	0.88	0.81	0.88	0.79	0.91	**0.99**	0.82	0.93	0.93	0.93
DUCK$_{\neg CC}$	0.85	0.91	0.86	0.81	0.82	0.85	0.84	0.91	0.78	0.87	0.87	0.98	0.75	0.94	0.94	0.94
DUCK$_{\neg UT}$	0.88	**0.92**	0.84	**0.91**	0.85	0.89	**0.91**	0.91	0.87	0.88	0.91	**0.99**	0.83	0.97	0.97	0.97
DUCK	**0.90**	0.91	**0.93**	0.88	**0.88**	**0.91**	0.89	**0.93**	**0.93**	**0.91**	**0.92**	**0.99**	**0.85**	**0.98**	**0.98**	**0.98**

Table 5: Overall rumour detection results. "CT", "CC" and "UT" denote comment tree, comment chain and user tree respectively, and "R" and "NR" in WEIBO denote rumour and non-rumour.

2020): fine-tunes a BERT pretrained with stance annotations for rumour detection and comments are modelled as a chain (similar to our one-tier transformer model); (4) **Bi-GCN** (Bian et al., 2020)[11]: uses a bidirectional graph convolutional network to model the comment network in a top-down (i.e. nodes are combined starting from the leaf comments) and bottom-up manner (i.e. nodes are combined starting from the root); and (5) **GCAN** (Lu and Li, 2020)[12]: uses graph networks to model the retweet user network and a CNN to model the source post with co-attention between the two networks. For a summary of the different features used by these benchmark systems and our model, see Table Table 1.

All benchmark results are produced using the author-provided code, with the exception of RNN+CNN and stance-BERT where we implement ourselves. Note that we only have English results (Twitter15, Twitter16 and CoAID) for stance-BERT as it uses stance annotations from SemEval-2016 (Mohammad et al., 2016), and GCAN and RNN+CNN do not have results for CoAID as it does not contain retweets.

We present the results in Table 5. DUCK (our model) performs very strongly, outperforming all benchmark systems consistently over all datasets, creating a new state-of-the-art for rumour detection. In terms of datasets, WEIBO appears to be the "easier" dataset, where most systems produce a macro-F1 over 90%. We also observe that models that use the comment texts (stance-BERT and Bi-GCN) tend to do better than those that only use the user network (RNN+CNN and GCAN), although the strong performance of DUCK indicates that combining both types of information works best, suggesting that they complement each other. Another thing of note is CoAID, the only dataset where the class distribution is heavily imbalanced. Here we see that most systems struggle with the minority class ("F"), but our combined approach appears to handle this well.

To understand the impact of each module in DUCK, we present variants where we remove one module, e.g. DUCK$_{\neg CT}$ means comment tree removed. Results suggest that comment tree has the largest impact, followed by comment chain as they produce the largest performance drop when removed. This finding is similar to what we saw earlier, where systems like stance-BERT and Bi-GCN that use comments tend to perform better.

6 Conclusion

We presented DUCK, a social media rumour detection approach that models both the network of users who interact with a story as well as their comments/opinions. Our approach is unique in how we model the comment as a graph (with BERT and GAT) and also as a stream (with transformers) and the user networks together with their peer relations (with GAT and GAE). We conduct extensive experiments over four popular rumour benchmark datasets to evaluate DUCK. We found that the comment network contains the strongest signal for predicting rumours, and social relations are important for modelling the user network. DUCK substantially outperforms all benchmark methods consistently, creating a new state-of-the-art.

[11] https://github.com/TianBian95/BiGCN
[12] https://github.com/l852888/GCAN

References

Gordon W Allport and Leo Postman. 1947. The psychology of rumor.

Iz Beltagy, Matthew E Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*.

Tian Bian, Xi Xiao, Tingyang Xu, Peilin Zhao, Wenbing Huang, Yu Rong, and Junzhou Huang. 2020. Rumor detection on social media with bi-directional graph convolutional networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 549–556.

Carlos Castillo, Marcelo Mendoza, and Barbara Poblete. 2011. Information credibility on Twitter. In *Proceedings of the 20th international conference on World wide web*, pages 675–684. ACM.

Limeng Cui and Dongwon Lee. 2020. Coaid: Covid-19 healthcare misinformation dataset. *arXiv preprint arXiv:2006.00885*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. 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 4171–4186, Minneapolis, Minnesota.

Zhiwei Jin, Juan Cao, Han Guo, Yongdong Zhang, Yu Wang, and Jiebo Luo. 2017. Detection and analysis of 2016 us presidential election related rumors on twitter. In *International conference on social computing, behavioral-cultural modeling and prediction and behavior representation in modeling and simulation*, pages 14–24. Springer.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Thomas N Kipf and Max Welling. 2016a. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Thomas N Kipf and Max Welling. 2016b. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*.

Elena Kochkina, Maria Liakata, and Isabelle Augenstein. 2017. Turing at semeval-2017 task 8: Sequential approach to rumour stance classification with branch-lstm. *arXiv preprint arXiv:1704.07221*.

Sejeong Kwon, Meeyoung Cha, Kyomin Jung, Wei Chen, and Yajun Wang. 2013. Prominent features of rumor propagation in online social media. In *2013 IEEE 13th International Conference on Data Mining*, pages 1103–1108. IEEE.

Quanzhi Li, Qiong Zhang, and Luo Si. 2019. Rumor detection by exploiting user credibility information, attention and multi-task learning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1173–1179.

Xiaomo Liu, Armineh Nourbakhsh, Quanzhi Li, Rui Fang, and Sameena Shah. 2015. Real-time rumor debunking on twitter. In *Proceedings of the 24th ACM international on conference on information and knowledge management*, pages 1867–1870.

Yang Liu and Yi-Fang Brook Wu. 2018. Early detection of fake news on social media through propagation path classification with recurrent and convolutional networks. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Zhenghao Liu, Chenyan Xiong, Maosong Sun, and Zhiyuan Liu. 2020. Fine-grained fact verification with kernel graph attention network. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7342–7351.

Yi-Ju Lu and Cheng-Te Li. 2020. Gcan: Graph-aware co-attention networks for explainable fake news detection on social media. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 505–514.

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

Jing Ma, Wei Gao, Zhongyu Wei, Yueming Lu, and Kam-Fai Wong. 2015. Detect rumors using time series of social context information on microblogging websites. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 1751–1754. ACM.

Jing Ma, Wei Gao, and Kam-Fai Wong. 2017. Detect rumors in microblog posts using propagation structure via kernel learning. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 708–717.

Jing Ma, Wei Gao, and Kam-Fai Wong. 2018. Rumor detection on twitter with tree-structured recursive neural networks. Association for Computational Linguistics.

Saif Mohammad, Svetlana Kiritchenko, Parinaz Sobhani, Xiaodan Zhu, and Colin Cherry. 2016. Semeval-2016 task 6: Detecting stance in tweets. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 31–41.

Van-Hoang Nguyen, Kazunari Sugiyama, Preslav Nakov, and Min-Yen Kan. 2020. Fang: Leveraging social context for fake news detection using graph representation. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 1165–1174.

Bhavtosh Rath, Wei Gao, Jing Ma, and Jaideep Srivastava. 2017. From retweet to believability: Utilizing trust to identify rumor spreaders on twitter. In *Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2017*, pages 179–186. ACM.

Kai Shu, Amy Sliva, Suhang Wang, Jiliang Tang, and Huan Liu. 2017. Fake news detection on social media: A data mining perspective. *ACM SIGKDD Explorations Newsletter*, 19(1):22–36.

Kai Shu, Suhang Wang, and Huan Liu. 2018. Understanding user profiles on social media for fake news detection. In *2018 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)*, pages 430–435. IEEE.

Kate Starbird, Jim Maddock, Mania Orand, Peg Achterman, and Robert M Mason. 2014. Rumors, false flags, and digital vigilantes: Misinformation on twitter after the 2013 boston marathon bombing. *IConference 2014 Proceedings*.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. Fever: A large-scale dataset for fact extraction and verification. 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 809–819.

Lin Tian, Xiuzhen Zhang, Yan Wang, and Huan Liu. 2020. Early detection of rumours on Twitter via stance transfer learning. In *European Conference on Information Retrieval*, pages 575–588. Springer.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903*.

Sarah Vieweg, Amanda L Hughes, Kate Starbird, and Leysia Palen. 2010. Microblogging during two natural hazards events: what Twitter may contribute to situational awareness. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pages 1079–1088.

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

Wanjun Zhong, Jingjing Xu, Duyu Tang, Zenan Xu, Nan Duan, Ming Zhou, Jiahai Wang, and Jian Yin. 2020. Reasoning over semantic-level graph for fact checking. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6170–6180.

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.

A Implementation Details

We implement our models in PyTorch using the HuggingFace library[13] and their pretrained BERT[14] and Chinese-BERT[15]. Graph neural networks are implemented with the Geometric[16] package.

For the comment tree, we set maximum token length=40 and dropout rate $= [0.5, 0.6]$ for GAT and 0.2 for BERT embeddings. Learning rate is tuned in the range between $[1e^{-5}, 5e^{-5}]$ for BERT and $[1e^{-4}, 5e^{-4}]$ for GAT based on the development set. For comment chain, the learning rate for two-tier transformer (comment chain) is tuned in between $[2e^{-5}, 5e^{-5}]$ with maximum token length as 40. For user tree, we set the dimension of each node hidden features as 256. All models use the Adam optimiser (Kingma and Ba, 2014), and our experiments are run using $4\times$A100 GPU with 40GB Memory.

[13] https://github.com/huggingface

[14] https://huggingface.co/bert-base-cased

[15] https://huggingface.co/bert-base-chinese

[16] https://pytorch-geometric.readthedocs.io/en/latest/.