Rumour Verification through Recurring Information and an Inner-Attention Mechanism

Ahmet Aker[1,3], Alfred Sliwa[1], Fahim Dalvi[2], Kalina Bontcheva[3]

University of Duisburg-Essen[1], QCRI[2], University of Sheffield[3]

Abstract

Verification of online rumours is becoming an increasingly important task with the prevalence of event discussions on social media platforms. This paper proposes an inner-attention-based neural network model that uses frequent, recurring terms from past rumours to classify a newly emerging rumour as *true*, *false* or *unverified*. Unlike other methods proposed in related work, our model uses the source rumour alone without any additional information, such as user replies to the rumour or additional feature engineering. Our method outperforms the current state-of-the-art methods on benchmark datasets (RumourEval2017) by 3% accuracy and 6% F-1 leading to 60.7% accuracy and 61.6% F-1. We also compare our attention-based method to two similar models which however do not make use of recurrent terms. The attention-based method guided by frequent recurring terms outperforms this baseline on the same dataset, indicating that the recurring terms injected by the attention mechanism have high positive impact on distinguishing between true and false rumours. Furthermore, we perform out-of-domain evaluations and show that our model is indeed highly competitive compared to the baselines on a newly released RumourEval2019 dataset and also achieves the best performance on classifying fake and legitimate news headlines.

Email addresses: a.aker@is.inf.uni-due.de (Ahmet Aker[1,3]), a.sliwa@is.inf.uni-due.de (Alfred Sliwa[1]), dalvifahim@gmail.com (Fahim Dalvi[2]), k.bontcheva@sheffield.ac.uk (Kalina Bontcheva[3])

1. Introduction

Social media platforms now play an important role in the lives of many. They are not only used for social interaction, entertainment and relaxation but also for information seeking and sharing [1]. However, social media platforms are rife with rumours, which following [2] we define as fast-spreading, unverified pieces of information. False rumours can cause major problems not only for everyday users, but also for professionals like journalists and institutions that rely on social information. Thus, in the past few years, automatic rumour verification has become a hot research topic. Most research on rumour analysis has focused on Twitter, as it has established itself as the go-to social platform for real-time news [3] coupled with its comprehensive API. Twitter's unmoderated nature is also the perfect ground for spreading rumours [4].

According to Zubiaga et al. [2], rumour analysis consists of a pipeline of four stages: rumour detection, rumour tracking, stance classification and rumour verification. Our work focuses on the rumour verification stage. The aim of rumour verification is to verify a rumour and label it as *true*, *false* or *unverified*. To tackle this problem, related work has mainly performed feature engineering and investigated different state-of-the-art machine learning algorithms and learning strategies [5, 6, 7, 8, 9, 10, 11, 12, 13]. Unlike the related work, we focus on recurrent patterns in the messages and use these pieces of information to distinguish between the veracity levels.

According to Sen et al. [14] messages during any crisis situation are centered on a finite set of terms related to that specific event. For instance, in the event of an airplane crash, the messages are likely to repeat words related to the crash itself, number of deaths, information related to the pilots, etc. Messages about a terror act will be dominated by words related to the terrorists, number of casualties, location, etc. However, different rumour levels (true, false and unverified) will use different wordings. For instance, false rumours will aim to exaggerate the situation, whereas true rumours would aim to report an incident according to known facts and minimise sentimental overlay.

We argue that these repeating or recurrent pieces of information in the rumour classes are important for the verification task. In this research, we use recurrent words to distinguish between the different veracity classes. To capture the recurrent words we use a simple but effective method of uni-gram counting. Frequent uni-grams are determined from the training data and are used to teach our model where in a Twitter message to focus. Our model is a combination of an LSTM layer with an attention mechanism.

The recurrent words are extracted from the source tweets only, i.e. our approach does not look at replying tweets. Several previous approaches to rumour verification use stance information of replying tweets [12, 13, 15]. The advantage of our approach is that the model can be applied directly on the first message reporting a rumour and does not wait until a certain number of replying tweets appear.

Compared to current state-of-the-art performances on known benchmark data (RumourEval2017), our results show 3% gain in accuracy and 6% in F-1 – so far the best reported figures are 57.1% accuracy and 55.8% F-1 [13]. To also show that the results are due to our use of the recurrent words, we compare our model to a vanilla LSTM as well as soft attention approach where recurrent words are not used. Again, the model with the recurrent words leads to best performance. We also test the model on two out-of-domain datasets. The first out-of-domain dataset is the newly released RumourEval2019 dataset that focuses on natural disasters. The second dataset entails headlines collected from fake and legitimate news. Our results show that our inner attention approach achieves satisfactory results on the RumourEval2019 dataset and reaches the best performance on the headlines dataset.

In the following section, we discuss related work. In Section 3 we describe the task as well as the data used in our experiments. Our method is presented in Section 4. Section 5 describes our experimental settings. Results are given in Section 6 and discussed in Section 7. We conclude our paper with immediate future directions in Section 8.

2. Related Work

The veracity classification task aims to determine whether a given rumour can be confirmed as true, debunked as false, or in some cases marked as yet to be resolved (i.e. unverified). Related work has tackled the problem in a supervised fashion by applying state-of-the-art machine learning algorithms on features extracted from rumour datasets. The pioneering paper of [5] proposed message, user, topic and propagation-based features. In most subsequent studies these features have been used as baselines. Following these feature sets, Kwon et al. [6] and Kwon et al. [16] proposed a new set of feature categories: temporal, structural and linguistic, and showed their importance in verifying rumours. Other than Twitter, the Chinese microblogging platform Sina Weibo also has been analysed for rumours. For this service, Yang et al. [17] proposed client and location-based features and showed that these help to increase prediction accuracy. Further studies modelled features over time [9], investigated other ways of determining the features such as from propagation trees [8, 18] or applied different techniques to model them such as using Hidden Markov Models [7]. Chen et al. [19] treated rumour veracity classification as an anomaly detection problem where false rumours are regarded as anomalies. Several features relating to the content, crowd opinion and post propagation were used. Chang et al. [20] put the emphasis on the characteristics of users who post the rumours to determine the veracity. Tong et al. [21] aimed at curtailing false rumour spread rather than marking tweets as true or false. Motivated by the fact that later corrections are not as effective, the authors argued that the first post seen by a user is influential for their future opinion and thus it is important to show users rumours only once they are confirmed to be true.

Rumour veracity classification has also been studied in the RumourEval shared task at SemEval 2017 [22]. Subtask B consisted in determining if each rumour in the dataset was *true*, *false* or remained *unverified*. Participants viewed the task either as a three-way [12, 23, 24] or two-way [25, 26], single tweet classification task. The winning system [12] added features more specific to the dis-

tribution of stance labels in the tweets replying to the source tweet (percentage of reply tweets classified as either support, deny or query). The authors report an accuracy of 53.6% on the RumourEval dataset. Authors of [13] performed a re-implementation of this winning system and reported a slightly better performance (57.1%). In addition, the authors propose a multi-task-learning model where they learn veracity, stance and rumour detection at the same time and reported again a performance of 57.1% accuracy. Note, in both studies [12, 13] stance from the responding tweets played a major role in achieving the score of 57.1%. The power of stance to verify rumours was also recently reported by Dungs et al. [15]. The survey paper of Zubiaga et al. [2] provides an extensive summary of current work on rumour verification, along with related tasks such as detection of rumours and stance classification of messages involved in rumours.

Unlike the related work we do not perform explicit feature engineering, nor do we rely on the wisdom from the crowd, e.g. stance information. Our approach works only with the source tweet and does not look at replying messages. The advantage of our method over the previous studies, especially those which rely on replying tweets and are forced to wait until a certain number of replies appear, is that it can work without any delays and therefore be applied in real time – whenever a source tweet appears, our approach can judge it in terms of veracity. In our approach, we make use of recurrent words extracted from source tweets and use them in an attention mechanism to steer the model direction to areas which are distinctive in different rumour levels. Note, attention mechanism has been applied by Chen et al. [27] to perform early rumour detection. However, unlike these authors, we apply inner-attention whereas the authors applied soft attention on top of the LSTM layers.[1] The idea of recurrent words or patterns have been also applied to perform rumour detection [5, 28]. For instance, Zhao et al. [28] use manually constructed patterns like <*is (that / this / it) true*> to

[1] We implemented a soft attention method as baseline to compare it with inner-attention approach.

detect tweets which enquiry the truth of information conveyed by some rumour spreading tweets. The patterns are used as features in traditional machine learning settings. We extract our patterns automatically from the training data and inject them through attention directly to our model. In addition, we use them to verify rumours instead of, as done by Zhao et al. [28], detecting or tracking rumours.

3. Task and Data

3.1. Task: Rumour Verification

Rumour Verification refers to the task of determining a rumour source post being either *true*, *false* or *unverified* and thus is treated as a 3-label classification task [2]. The label *unverified* is used when there is not enough information to verify the message. The input to the task is a source tweet reporting a rumour. Optionally, the source tweet may have replying tweets. Based on the source tweet and the optional replying tweets the task is to label the rumour with a class label.

3.2. Data

We use two rumour datasets for the rumour verification task. Both datasets include conversation threads in Twitter about several events. Each conversation thread is represented in a tree structure with the source tweet as a root node and several branches of replying tweets. A reply chain is a sequence of tweets from root node to a leaf node.

3.2.1. RumourEval2017

Our first data resource is the RumourEval2017 dataset [29] which is derived from the PHEME dataset [30]. The RumourEval2017 dataset contains 325 Twitter conversation threads discussing rumours with respect to eight different events like Germanwings Air Crash, Charlie Hebdo, Ottowa Shootings, etc. All the events are man made. Each thread in the dataset is annotated as *true*, *false* or *unverified*. Also each reply to a source tweet is annotated with one of the

Split Set	# Rumours	# True	# False	# Unverified
Train	272	127	50	95
Dev	25	10	12	3
Test	28	8	12	8
Total	325	145	74	106

Table 1: Distribution of RumourEval2017 dataset.

labels: *supporting*, *denying*, *questioning* and *commenting*. An example conversation structure is shown in Figure 1. As it was used for the RumourEval2017 challenge it is split into training, development and held-out test set. Table 1 shows the class distribution for the three sets.

Figure 1: RumourEval2017: Example conversation structure.

Note the split of the dataset into training, development and testing sets is performed randomly. Also note that the rumours in the dataset are independent of each other so that there is no timely development between the rumours. This is also the case for the RumourEval2019 dataset (see next Section).

Split Set	# Rumours	# True	# False	# Unverified
Test	56	22	30	4

Table 2: Distribution of RumourEval2019 dataset.

3.2.2. RumourEval2019

Our second data contains rumours about natural disasters such as hurricanes, tornados, floods, etc. and thus has a completely different domain focus as the RumourEval2017 data. The structure of the data is similar to the RumourEval2017 and contains conversations with a source tweet that starts a rumour and replies reacting to the source tweet. Also similar to the RumourEval2017 data, the veracity levels are *true*, *false* and *unverified* and have been also annotated for stance (*supporting*, *denying*, *questioning* and *commenting*). This data was adopted by the RumourEval2019 challenge as testing data. In this work we refer to this data as the RumourEval2019 dataset. Statistical details about it are shown in Table 2.

3.2.3. Headlines

In addition to the rumour datasets described in the previous sections, we also used news headlines to validate our model. More precisely we have news headlines extracted from manually determined fake news articles and legitimate articles. Here the idea is to validate our model whether it is able to distinguish legitimate headlines which follow rather a reporting style from fake news headlines that use catchy and exaggerating terms.

We use the data repository FakeNewsSet[2] provided by [31] for the purpose of fake news headline instances. For the legitimate news we took random 100 articles from the Guardian and the independent and used their headlines as legitimate instances. The domains of the articles – fake and non-fake – are from the political and economic areas. In total we have 100 headlines from the fake articles and 100 headlines from the true news. We label the fake news

[2]https://github.com/KaiDMML/FakeNewsNet

Veracity Class	Example Headlines
False	WHOA! NEW DISTURBING VIDEO Shows HILLARY'S Campaign Likely FAKED Her Audience At NC Rally
True	New car sales plunge 20 % in September

Table 3: Example fake and non-fake headlines.

related headlines as *false* and the headlines from the legitimate articles with *true*. Example of true and false news headlines are shown in Table 3.

4. Method

4.1. Core Idea

An event causes many social media users to form groups around related issues and hashtags [32]. These groups use language that is narrow and is centered on a finite set of terms related to the event [14], i.e. there are terms that repeat or recur among the messages. We argue that these recurring terms carry useful information to distinguish between rumour classes. In Table 4 we list some frequent uni-grams for different rumour classes. In the *true* rumour case we can see that the terms indicate some fact reporting behaviour such as "live, died, freed, identified", etc. However, unlike *true* rumours the *false* ones have a different style. They tend to contain words commonly used to exaggerate like "horrifying, terrorist", etc. This indicates that they rely on emotionally coloured language aiming to fuel fear in readers. Interestingly, false rumours even contain terms like "rumours, false, fake", which are used to attack other pieces of information reporting about the event and claim that they are all false. This has been also reported by earlier work such as Starbird [33] who reports that false messages accuse true ones of reporting fake information. For the *unverified* messages the language is vague and thus difficult to classify as one of the previous classes – probably this is also why the messages in this class were marked as unverified. However, as in the true rumours the terms in the unverified messages seem to follow a reporting behaviour such as "alleged,

Veracity Class	Example Terms
False	rumours, terrorist, #muslim, internet, horrifying, false, cold, fake
True	live, died, freed, free, follow, identified
Unverified	alleged, disappeared, military, give

Table 4: Example recurring terms extracted for each veracity class from RumourEval 2017 dataset.

disappeared" but the message is not as clear as in the case of true rumour. Our aim is to capture these recurring terms and use them to guide our classification model. We capture the recurring terms using the frequency distribution of unigrams observed in the training data.

4.2. Inner-Attention LSTM

In our work, we tackle the veracity classification using a inner-attention based LSTM neural network [34]. Our model resembles the ones used in earlier work for question answering [35] and topic specific argument retrieval [36]. Figure 2 shows the architecture of our inner-attention based LSTM network.

This neural network is able to learn the importance weighting of source tweet words with respect to a given set of recurring terms. Both the source words as well as the recurring words are represented using word embeddings (Word2Vec [37] or Glove [38] trained on the Twitter dataset[3]). For each source word, an individual weighting factor is learned denoting the importance of that word with respect to the recurring terms. The following formula is used to obtain this weighting factor α_i for each word embedding x_i:

$$\alpha_i = \sigma(u^T W_s x_i) \tag{1}$$

where u is the averaged recurring terms embedding, and W_s are the parameters of the embedding mechanism. Each word embedding x_i is then multiplied by its respective relevance weight factor α_i using Formula 2.

[3]https://nlp.stanford.edu/projects/glove/

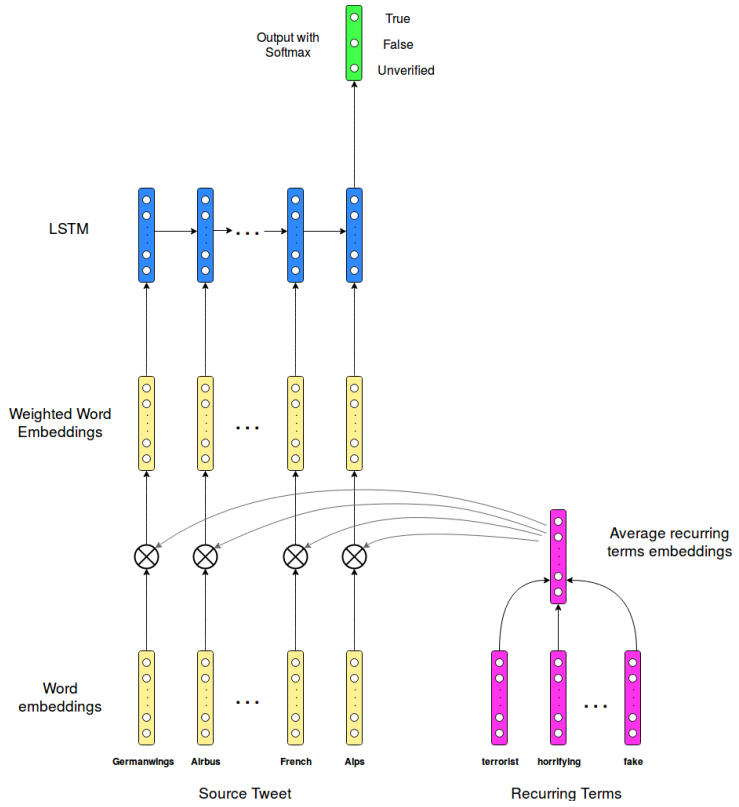

Figure 2: Inner-Attention LSTM network.

$$\bar{x_i} = \alpha_i x_i \tag{2}$$

In this way, non-relevant source words are filtered out or weighted less. This leads to weighted source tweet word embeddings, on which an Long Short Term Memory (LSTM) [39] layer is applied in the next step. The LSTM reads each weighted tweet word at a time and produces an output when the end of the tweet is reached (last word). The output depends on the last word but also on what the LSTM has read till the last word. This history of information starts with the first word and is built up with every new input. However, instead of letting every new word contribute to the historical knowledge LSTM unit applies, similar to humans, some kind of information focus. It uses only those

words in the historical knowledge which are also worth to consider. Words that are not worth to consider are not used in construction of the history knowledge. Once the tweet ends LSTM constructs the output. In our model this output is passed to a dense layer (feed forward neural network) which is followed by softmax. Note the LSTM units also used in the baseline systems work on the same principle as described here.

4.3. Real Time Usage

As noted earlier, in our method we tackle the rumour verification with the source tweet alone, i.e. the tweet that is the origin of the rumour. This means that reply chains with respect to the source tweet are not required. This makes our approach applicable for any source tweet with or without replies. Furthermore, since we do not use replying tweets, we also do not require knowledge such as stance information from them. This makes our approach independent from any additional pipelines which might introduce further noise for the subsequent parts. Due to these facts – focus only on the source tweet and no use of additional pipelines – we argue that our approach can be applied in real time situations and on any short message reporting a rumour.

Note, what we refer with real time usage is that since our approach works only on the source tweet that it can be applied once such a source tweet about an event such as the German Aircraft crash is posted. The source would be carrying a rumour that requires verification. In this current study our aim is to investigate the contribution of recurring terms for this task. Also as noted in Section 3.2.1 the rumours in both RumourEval datasets are independent of each other. This setting makes our approach suitable for distinguishing between rumours that carry exaggeration and false information and those that aim to report proper pieces of information.

5. Experiments

5.1. Baselines

*5.1.1. NileTMRG**

The best system on the RumourEval2017 shared task achieved 53.6 % accuracy on the test dataset [12]. The winning system relied on a traditional feature engineering approach by proposing a linear SVC with a concatenated feature set consisting of a Bag of Word (BoW) source tweet representation, the existence of URL and hashtags as well as the percentage of stance information from the replying tweets (querying, denying or supporting). A re-implementation of the NileTMRG was performed by Kochkina et al. [13]. The new model NileTMRG* achieves an accuracy score of 57.1% on the same dataset.

5.1.2. Branch LSTM

The branch LSTM (branchLSTM) is a sequential approach which consists of several LSTM layers that are connected to several feed-forward ReLU layers. The prediction layer consists of a softmax layer and outputs the probabilities for the class labels [30]. Kochkina et al. [13] adopted this model for the rumour verification task and applied on the RumourEval2017 dataset.

5.1.3. Multi-Task Learning

Kochkina et al. [13] use a multi-task-learning model in order to predict veracity status of rumours. The model jointly learns to predict for the tasks of rumour detection, stance classification and rumour veracity classification. The joint learning of all the tasks leads to the best performing setting. Using this setting the authors report 57.1 % accuracy on the RumourEval2017 test dataset.

5.1.4. Vanilla LSTM

In addition to the baselines described above we also use a vanilla LSTM without the inner-attention mechanism. This is to evaluate whether recurring terms applied through the attention mechanism have an impact on the results

or not. Apart from omitting the inner-attention part, the remaining parts of the model shown in Figure 2 are kept for the vanilla LSTM baseline.

5.1.5. LSTM with Soft Attention

In the last years attention mechanism methods have been applied on the rumour detection task to highlight terms that are helpful to distinguish between rumour/non-rumour representations [27]. To compare our proposed Inner-Attention LSTM we use such an attention mechanism, also referred to as soft attention mechanism [34], as additional baseline. This approach makes use of LSTM working with an attention model. However, unlike our inner attention model, this one does not require recurring term as input to draw attention on the indicative words in the source tweet. LSTM model with soft attention applies attention on top of the LSTM layer in order to determine the important parts of the source tweet after it passes the LSTM layer (see Figure 3). The outputs of the LSTM layer (at each timestamp) are weighted and aggregated. The weights are learned through the attention mechanism. The result after the aggregation is a context vector c:

$$c = \sum_{t=1}^{T} \alpha_t h_t \tag{3}$$

where T is the total number of time steps in the hidden state sequence h and α_t is an individual weight factor at each time step t. Each weighting factor α_t is derived by Formula 4.

$$\alpha_t = \sigma(\tanh(W_s h_t)) \tag{4}$$

α_t is determined by a learning function that is composed of its hidden unit h_t and a trainable weight vector W_s. Softmax is applied to transform the weighting factors into a probability distribution.

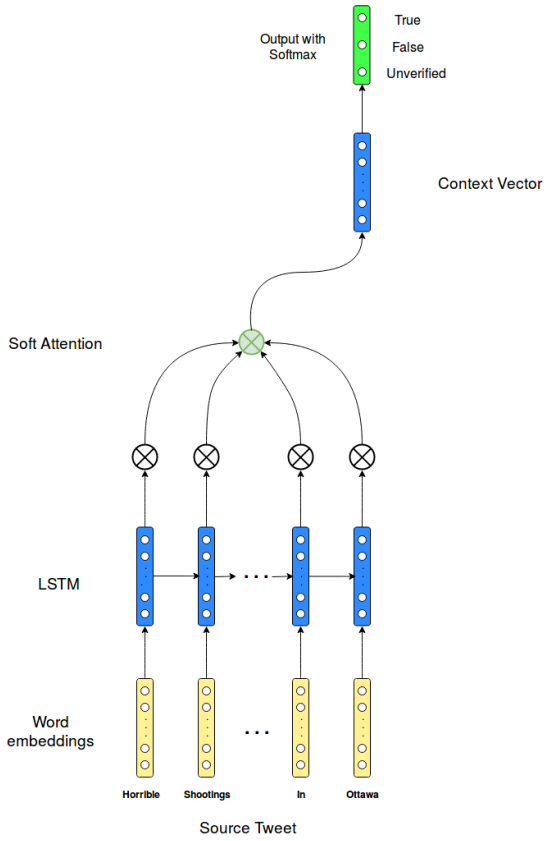

Figure 3: LSTM network with Soft Attention

5.2. Experimental Set-up

5.2.1. Experiments on RumourEval2017 and RumourEval2019

For the rumour veracity classification task we conduct two experiments based on different data set-ups. We also conduct an experiment where we apply existing models on the news headlines. In total we have three experiments. As noted above rumours (source tweets) (Section 3.2.1) in our datasets carry some pieces of information put forward about an event and thus require verification. Each such piece of information is independent from the others and thus the rumours about an event do not have timely ordering. This is also the case for the headlines. Therefore in our experiments we treat each rumour/headline independent

from others and assess its veracity only based on the recurring information.

First Experiment. :

The first experiment uses the same train/dev/test split as used in the RumourEval2017 Task (cf. Table 1). This means we train our models on the train set, tune the parameters using the dev set and test the tuned model on the test set.

Second Experiment:. The second experiment makes use of the RumourEval2019 dataset. In this second experiment we have two different settings.

Setting 1:. In the first setting we keep the training from RumourEval2017 dataset as it is, however, merge the dev and test sets together and use this merged dataset for parameter tuning. The entire RumourEval2019 dataset is used as testing data. This tests the performance of a domain independent model (RumourEval2017) on a foreign domain (RumourEval2019).

Setting 2:. In the second setting, we extend the previously training data with additional RumourEval2019 data (20% – we will release the splits for the community to re-produce our experiments). The merged dev and test sets from RumourEval2017 are still used for tuning the model parameters. This tests the contribution of in-domain data on out-domain trained models. The test happens then on the remaining 80% of the RumourEval2019 data.

Third Experiment:. In this experiment, we replicate the settings as done in experiment 2 but replace the testing data with the news headlines. This means we have again two settings:

Setting 1:. Same as in experiment 2 with setting 1 but the testing data is the entire news headline dataset. The RumourEval2019 dataset is not used.

Setting 2:. Again same as in experiment 2 with setting 2 but the injected additional data comes from the headline dataset. Also the testing is purely performed with the news headlines.

In all settings the results are macro-averaged. Figure 4 shows the distribution of train/dev/test sets of all experiments and settings.

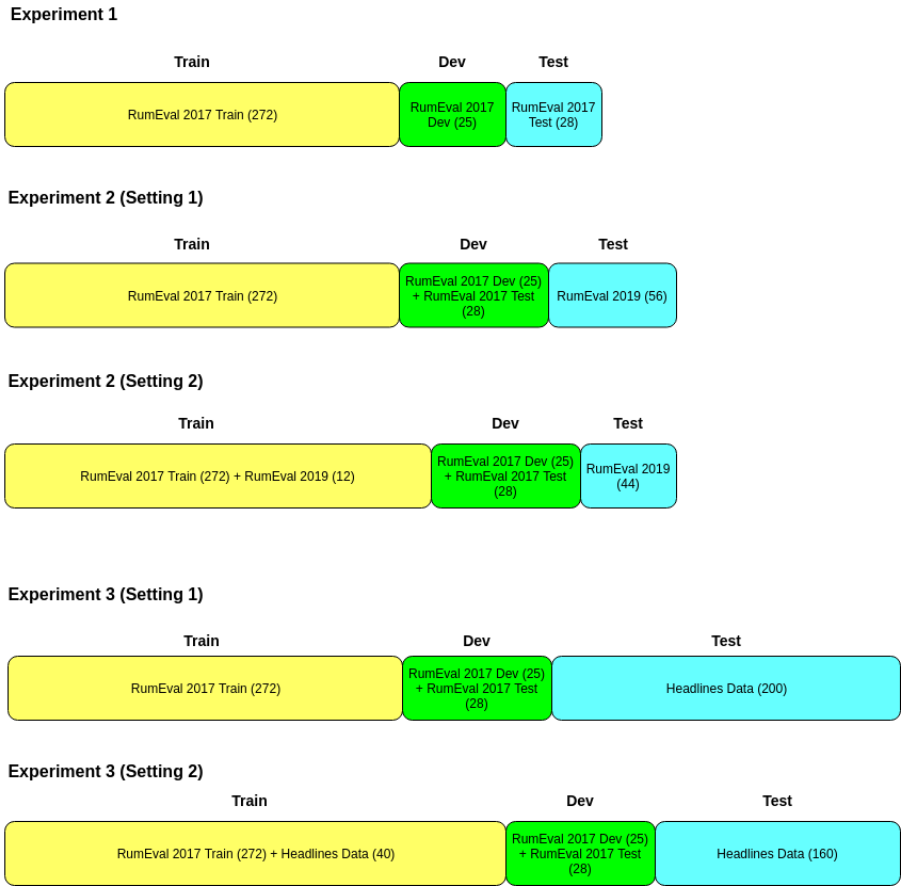

Figure 4: Train/Dev/Test Distribution Sets of all experiments and settings.

5.3. Hyperparameters

For the baselines reported by related work (NileTMRG*, branch LSTM and multi-task learning approaches) we use the models as reported. In the following we describe the hyperparameter tuning performed on the three LSTMs implemented in this work.

5.3.1. Inner-Attention LSTM

Hyperparameter tuning is used for finding the best model configuration. We trained several models based on different LSTM sizes, dropout rates, batch sizes, learning rates and epochs. Our best model is trained for 11 epochs with ADAM optimization algorithm, an LSTM size of 10, a learning rate of 0.01, a dropout rate of 0.1 and a batch size of 16.

We also investigated both GloVe and Word2Vec embeddings. The 25 dimensional GloVe word embeddings performed better than Word2Vec.

5.3.2. Vanilla LSTM

The best Vanilla LSTM model is trained on 10 epochs with ADAM optimization algorithm, an LSTM size of 64, a learning rate of 0.01, a dropout rate of 0.1 and a batch size of 16.

5.3.3. Soft Attention LSTM

Our second attention-based model gains best performance results by training for 11 epochs with ADAM optimization algorithm, an LSTM size of 16, a learning rate of 0.01, a dropout rate of 0.1 and a batch size of 16. In terms of hyperparameters this model does not differ much from our inner-attention attention-based model.

5.4. Usage of Recurrent Terms

Our recurrent terms are extracted based on two conditions: (1) We use distinct terms for each class, i.e. we make sure terms occurring frequently in one veracity class never occur in the other veracity classes. (2) A term must appear in at least 2 events. The second condition aims to generalize the term extraction by avoiding terms from one event dominating the entire set of recurring terms which are provided as input to the LSTM model.

We tried several strategies to use the recurring terms in our inner-attention LSTM model. In out first strategy, we extracted from each rumour class up to 20 frequent terms and used them all in our model. In the second strategy,

we included only terms from the *true* and *unverified* classes and omitted terms from the *false* class. Finally we used terms only from the *false* class. Note that the last two settings aim to distinguish the *false* class from the other two classes. We achieve the best results when we guide our LSTM model with the *false* terms only. The results reported in this work are all based on recurring terms extracted from the *false* rumours. These terms are extracted from the RumourEval2017 dataset.[4]

6. Results

Table 5 shows the evaluation results of our proposed method compared to the baseline approaches. We first report results on the RumourEval2017 dataset (experiment 1).

First we see that all systems outperform the majority vote baseline which simply outputs the class that has the majority class in the training data. The baselines NileTMRG* and multi-task learning achieve the same accuracy score of 57.1% and differ in F-1 scores (53.9% vs. 55.8% F-1) . Both systems are currently known state-of-the-art systems on the RumourEval2017 dataset. The branchLSTM performs slightly worse and achieves only 50% accuracy and 49.1% F-1. A similar performance is achieved with the soft attention model leading to 50% accuracy and 49.6% F-1. The Vanilla LSTM model performs better than the branchLSTM and the soft attention models and achieves 53.7% accuracy and 52.8% F-1. On the other hand, the proposed inner-attention based LSTM model outperforms the state-of-the-art models by 3% in the accuracy metric and almost by 6% in F-1.[5] This shows that injecting the model with recurrent words indeed helps the model to do better distinction between the classes.

[4]Note terms can also be prepared manually and injected to the system. This is especially useful if the manual terms present a better distinction between certain classes of a given problem.

[5]These are results of the model after parameter tuning. We also experimented with models, this is also the case for all subsequent experiments, where no parameter tuning was performed. Rresults of this experiment are shown in Appendix A.

	Majority Vote	**Nile TMRG***	**Multi Task Learning**	**branch LSTM**	**Vanilla LSTM**	**Soft Attention LSTM**	**Inner-Attention LSTM**
Macro F-1	0.429	0.539	0.558	0.491	0.528	0.496	**0.616**
Accuracy	0.2	0.571	0.571	0.5	0.537	0.5	**0.607**

Table 5: Results from experiment 1. Comparison of performance of several baselines with Inner-Attention LSTM model tested on RumourEval2017 dataset.

	Multi Task Learning	**Vanilla LSTM**	**Soft Attention LSTM**	**Inner-Attention LSTM**
Macro F-1	0.285	0.333	**0.413**	0.399
Accuracy	0.304	0.482	0.464	**0.554**

Table 6: Results from experiment 2, setting 1. Comparison of models implemented in this work and strong baseline from related work (according to F-1).

The results of the second experiment (experiment 2), setting 1, are shown in Table 6. Note that in this experiment, we evaluate the performance of an existing model on out-of-domain data. For this experiment, we use the models implemented in this work – the proposed inner-attention model, the re-implementation of the soft attention model and finally the vanilla LSTM model – and compare it with the best performing baseline from related work (according to F-1), which is the multi-task learning model.

From the results, we see that there is a performance drop by all the systems due to the out-of-domain application. The biggest drop happens for the multi-task learning system and the least drop is seen for the soft-attention LSTM approach. In case of performance the proposed inner-attention approach is superior to all other models in terms of accuracy. It is able to classify the

	Multi Task Learning	**Vanilla LSTM**	**Soft Attention LSTM**	**Inner-Attention LSTM**
Macro F-1	0.477	0.396	**0.633**	0.478
Accuracy	0.489	0.636	0.636	**0.705**

Table 7: Results from experiment 2, setting 2. Comparison of models implemented in this work and strong baseline from related work (according to F-1)

instances with 55.4% accuracy. In terms of F-1 although it has a huge drop on this metric it is only outperformed by the soft attention approach and achieves better results compared to the other two baselines.

We also tested the scenario of recover by injecting some additional data from the RumourEval2019 testing data to the training data. This is the experiment 2 with setting 2. The results for this experiment are shown in Table 7.

From the results in Table 7, we can see that all systems recover and get a boost in performance after seeing 20% of the new data during training. The trend in terms of performance ranking continues among the systems. In terms of accuracy, the inner-attention approach is superior to all other systems. However, according to F-1, the soft attention method shines and outperforms all other systems by a large margin.

7. Discussion

For results shown in Table 5:. Based on the experiment 1 we noted that our inner-attention approach performs substantially better than the current reported state-of-the-art systems such as NileTMRG* and the multi-task learning approaches. This shows that the recurrent terms indeed help to guide the model in terms of better classification. In addition, it should be noted that these systems use additional information such as stance from the replying tweets. Thus

they are forced to wait until some number of replies happen for the source tweet. Our system does not have this limitation and thus have also another advantage that it can be used whenever there is a source tweet. For instance, in the absence of stance information, the multi-task learning system's performance drops to 37% in F-1 and 41% in accuracy [13]. Also systems similar to NileTMRG* like e.g. [23] that focus on feature engineering – but unlike NileTMRG* omit crowd wisdom such as stance information – achieve an accuracy of only 46.4% on the RumourEval2017 shared task. The proposed approach outperforms also the vanilla LSTM and the soft attention based LSTM approaches with great differences in the scores.

As described in Section 5.4, the inner-attention model uses terms that are selected by conditioning e.g. that they occur in one class (false) but do not appear in the other class (true). So there is in advance a clear distinction in the e.g. false-vs-true rumour terms and the results show that exactly this clear distinction helps the model to perform best. The other systems try to learn this distinction on their own, but given the small training data size the distinction stays blurry and they perform poorly.

However, we also injected stance information in our proposed inner attention model to investigate any positive impact the stance would have in the task. The resulting architecture is shown in Figure 5. In this model we simply aggregate the stance information from the replying tweets (e.g. total number of supporting stances) and merge them with the output of the LSTM before inputting to the softmax layer. This late fusion of stance to the LSTM output aims to emphasize on the importance of stance features and should help to use the stance information effectively.

The late fusion architecture shown in Figure 5 leads to 56% in macro F-1 and 57.1 accuracy. This is less than what our original model alone achieves. A reason for this, as also shown by [15], is the way how stance information is passed to the system. Dungs et al. [15] show that stance orientation changes over time and thus is important to capture this development to better capture the wisdom from the crowd and employ it on the verification problem. Our simple aggregation

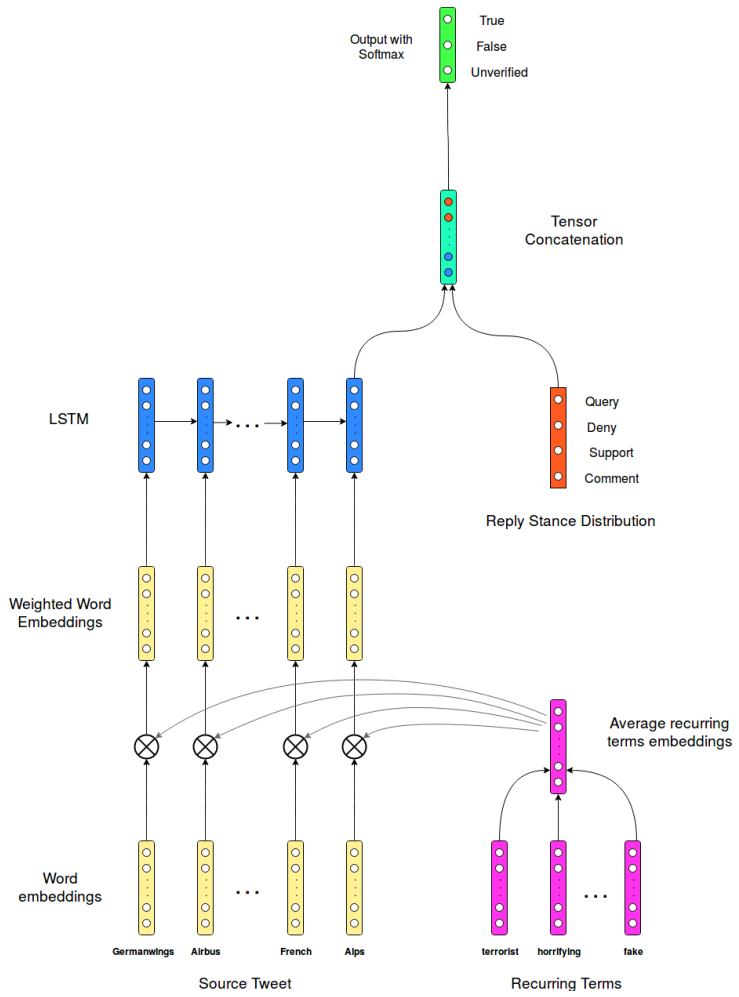

Figure 5: Inner-Attention LSTM network with Replies Stance Distribution.

model does not have this capturing characteristics and simply collapses the different stance information given at different time stamps together and employs the aggregated information on the verification problem. This leads to confusion of the system and affects the results negatively. In our future studies we plan to investigate alternative architectures that capture also stance information well.

Finally, as discussed in Section 4.1 the *false* rumour recurring terms tend to use an exaggerating and the *true* rumours a reporting language (cf. Table 4).

In Table 8 we show an example of a false rumour predicted as such. The table shows where the model focuses most (the higher the value the more important is the term for the model for performing the right class prediction). It is clearly recognizable that the words "Muslim", "big", "fat" and "mistake" are highly weighted by the model. Most of them also follow an exaggeration pattern and thus the model decides the false label for this particular example. In Table 9 we also show an opposite example with a true rumour that is also correctly classified by our system. The model does not pay almost no attention on stop words, the number and the month name "september". It pays attention on the other terms which follow unlike in the false rumour example rather a reporting style. The model makes use of this difference in the style of reporting and classifies this rumour as true.

Merkel 0	admits 0.9	bringing 0.01	Muslim 0.99	Refugees 0.4	was 0.2	a 0.01	big 0.99	fat 0.99	mistake 0.99

Table 8: Attention Weighting of a correctly classified false rumour post example.

New 0.81	car 0.99	sales 0.5	plunge 0.2	20 0.01	% 0.01	in 0.01	September 0

Table 9: Attention Weighting of a correctly classified true rumour post example.

The *unverified* rumours have vague terms but they tend to look similar to the true rumour terms. This means that the false rumours are lexically different from the other two classes. To understand whether this phenomenon is also reflected in the model behaviour, we computed the confusion matrix (Figure 6) of the model trained on experimental 1 setting (experiment 1).

From the matrix we can see that the model is able to make the correct pre-

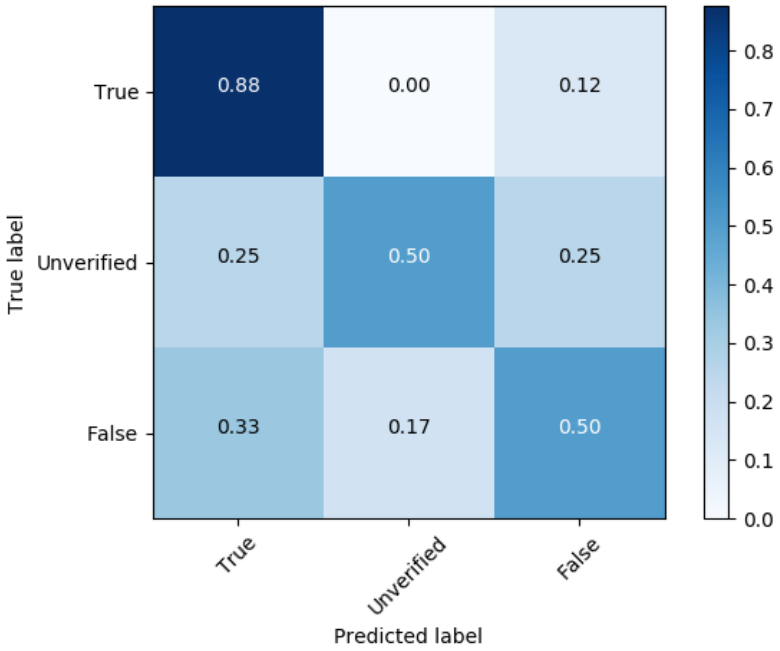

Figure 6: Confusion matrix of the inner-attention based LSTM model on the RumourEval test dataset.

diction for 88% of the time for the true class and confuses them only in 12% of the cases with the false class. This shows that the recurring terms helped to guide the model to perform the class distinction during the classification. It learned that true instances do follow different behaviour than false instances. However, interestingly the false class is confused with the true and the unverified classes. The model miss-classifies 50% of the false messages to the true and unverified categories. We think this is the case when the false rumour starts using the reporting style as in those two classes and omits indicators of exaggeration. This is e.g. shown in Table 10 with a false rumour that is classified by our model as unverified. The model focuses mostly on the terms like "murder", "every" and "life" which seem to try exaggerate a phenomenon. However, they are also put into context with reporting style words like "declare". Because of

this the model confuses the false rumour with the unverified class. For real use scenario this means that the proposed system is good in detecting obvious false rumours i.e., rumours that follow no reporting style at all but contain exaggerations, and struggles to capture false rumours following reporting style. Thus in real use scenario we think that the proposed system could work as a pre-filter to detect such obvious false rumours. To also detect false rumours that follow reporting style the system requires to look deeper on other aspects such as user meta information, communication network but also at the replies. We aim to enrich the proposed approach with such aspects in our future work.

Pro-Lifers 0	declare 0.01	ejaculation 0.01	is 0.01	murder 0.99	every 0.99	sperm 0.01	cell 0.01	is 0.01	a 0.01	life 0.99

Table 10: Attention Weighting of a wrongly classified false rumour post example as *unverified*.

Finally, we have also mis-classification in the unverified category (confusion with the true class). As discussed in Section 5.4 the unverified messages are rather vague and we think this is reflected in the results.

For results shown in Table 6 and 7:. In the second experiment (setting 1 and setting 2) we performed an out-of-domain evaluation. The results showed that overall there is a performance drop in all systems when the model is trained using the RumourEval2017 dataset and tested on the RumourEval2019 testing data. This drops show that all systems are somewhat domain dependent and thus have difficulties to perform decisions on an unseen domain. The most severe case was observed with the multi-task learning approach and least with the soft-attention LSTM. Overall the best performing systems were the inner-attention approach according to accuracy and the soft attention LSTM according to F-1. Later in the setting 2 we injected (see Table 7) some out-of-domain data to the training set and could observe recoveries in all systems. The highest recovery was observed in the soft attention LSTM and less in the other models.

These behaviours are connected with how the models are focused on the task. The multi-task learning as well as the vanilla LSTM approaches have no attention mechanisms integrated. They have only the LSTM units to adopt themselves to the task – learn what terms are useful to distinguish between the classes. However, this can only happen with a lot of data which is unfortunately not available. This is why these models do not have much chance to adopt themselves to the task.

The soft attention model has also an LSTM in its core but it has the advantage that it uses an attention part on top of the LSTM which helps the entire model to focus on terms that are indicative for the different classes. It is an extra help for the model to do something more quickly. The recovery results from Table 6 to 7 show this phenomenon.

In case of the inner-attention, the selection of the recurrent terms is very important. This is the extra bit that helps the model to quickly focus itself to the right classes. However, as described in 5.4 our terms are taken from the false rumours and also only from the training data. In experiment 2 with setting 1 all the false rumours come from the RumourEval2017 data and more than 90% in case of setting 2 (4 false rumours from RumourEval2019 have been injected to the training data). This means the recurrent terms still represent the RumourEval2017 domain and how false rumours are expressed in man made disasters. However, unlike the man made disasters the natural catastrophes follow rather a different language usage and have little in common with the false terms. For instance, recurrent terms extracted from the RumourEval2019 entire dataset contain in the false category are terms like *wiping, illegal, incredible, never* (see Table 11). These are certainly different from those extracted from the RumourEval2017 dataset (see Table 4) and thus the model has little information from the attention part to be guided. This is why we think that the inner attention approach struggles with out-of-domain data.

The weakness is maybe a strength on something else:. In Section 4.1 we discussed that the true/unverified recurrent terms follow a reporting and that the

Veracity Class	Example Terms
False	WTF, wiping, illegal, incredible, never, flooded, infidels, THUGS
True	weather, islands, hurricanes, guard
Unverified	nuclear, ocean, drop, helping

Table 11: Example recurring terms extracted for each veracity class from RumourEval2019 dataset.

	Vanilla LSTM	**Soft Attention LSTM**	**Inner-Attention LSTM**
Macro F-1	0.295	0.323	**0.539**
Accuracy	0.485	0.545	**0.805**

Table 12: Some models trained on RumourEval2017 dataset and tested on headlines data (Experiment 3, setting 1).

terms extracted from the false message follow an exaggerating style. From the results shown in Table 5, we have seen that our inner attention model has learned to use them well to distinguish between the different classes. A logical question which arises with this is whether such a model is also able to distinguish between news headlines of legitimate news (which follow rather reporting style) and fake news headlines where the language is rather exaggerated. To analyse this we performed the experiment 3 with setting 1 and 2. For these experiments the headlines dataset described in Section 3 is used.

According to experiment 3 setting 1 we first apply the RumourEval2017 trained model on the entire headlines dataset – this means the entire headlines dataset is treated as testset. In the second case (setting 2) we inject 20% from the headlines data to the RumourEval2017 training set and test the resulting model on the remaining 80% headlines. We use the vanilla, soft attention and the inner-attention LSTM approaches. Note, these are the only models which do not use additional information such as stance and thus are directly applicable to the headlines use case. The other systems require also stance information from replies which do not exist in the headlines case.

	Vanilla LSTM	**Soft Attention LSTM**	**Inner-Attention LSTM**
Macro F-1	0.49	0.344	**0.811**
Accuracy	0.734	0.581	**0.813**

Table 13: Some models trained on RumourEval2017 dataset with 20 % reserved headlines data and tested on remaining headlines (Experiment 3, setting 2).

From the results shown in Table 12, we see that the inner-attention LSTM approach outperforms the other two baselines significantly ($p < 0.001$).[6] It clearly learned how to distinguish between reporting style (true messages) and exaggerated language (false messages). It achieves an accuracy of 80.5% and 53.4% F-1. Furthermore, it learns to adopt itself more on the task after seeing 20% of the headlines (results shown in Table 13). In this case its F-1 performance gets boosted to 81.1% and gains slightly further on accuracy. In the two other models the difference between reporting and exaggeration is not captured and thus they fail to perform well on this second out-of-domain data.

8. Conclusion and Outlook

We have proposed a rumour veracity system that solely works on the content of source rumour post without requiring reply messages and additional crowd wisdom such as stance information. By leveraging the fact that false rumours are reported with different recurring terms in contrast to true as well as unverified rumours, we use the recurring false terms and inject them in our inner-attention LSTM in order to guide it in the distinction between the rumour classes. We compare the results of the proposed approach to vanilla LSTM and soft attention model where no attention on these (false) indicators is performed along with three baselines which reported state-of-the-art results on known bench mark

[6] We performed an independent 2-samples t-test with Bonferroni correction.

data. In the RumourEval2017 experiment our model outperforms the state-of-the-art classifiers leading to 60.7% accuracy and 61.6% F-1. We also showed that simple aggregation of the stance information from the replies and merging it with the inner-attention network does not lead to better performance but rather confuses the system slightly. Additionally we conducted an experiment using the RumourEval2019 as test set and showed that our model is competitive compared to the baselines, achieves even best results in terms of accuracy but fails to take the lead in terms of F-1. Finally we showed that our proposed model shines in the headline experiments and clearly outperforms baselines with large margin.

In the future we aim to investigate alternative architectures that capture both recurring patterns and stance information. However, we also plan to investigate user information as well as communication network as additional features in our system. Next, we aim to also apply our model to different problems such as hate speech and help seeking/offering detection in disaster situations. In both cases, users reporting in social media will tend to use recurring terms. We think that capturing and using these terms will help to tackle these tasks.

Acknowledgements

This work was partially supported by the European Union under grant agreements No. 654024 SoBigData, No. 825297 WeVerify and the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) - GRK 2167, Research Training Group "User-Centred Social Media".

Appendix A

Table 14 shows the performance of our inner attention model with and without parameter tuning.

	Inner-Attention LSTM	**Inner-Attention LSTM w/o parameter tuning**
Experiment 1	0.607/0.616	0.534/0.536
Experiment 2 (Setting 1)	0.399/0.554	0.373/0.536
Experiment 2 (Setting 2)	0.478/0.705	0.467/0.659
Experiment 3 (Setting 1)	0.539/0.805	0.505/0.76
Experiment 3 (Setting 2)	0.811/0.813	0.778/0.781

Table 14: Performance comparison of inner-attention LSTM model w/ and w/o parameter tuning. The first number refers to Macro F1-score while the second number represents accuracy.

References

[1] A. Whiting, D. Williams, Why people use social media: a uses and gratifications approach, Qualitative Market Research: An International Journal 16 (4) (2013) 362–369.

[2] A. Zubiaga, A. Aker, K. Bontcheva, M. Liakata, R. Procter, Detection and resolution of rumours in social media: A survey, ACM Computing Surveys (CSUR) 51 (2) (2018) 32.

[3] M. Hu, S. Liu, F. Wei, Y. Wu, J. Stasko, K.-L. Ma, Breaking news on twitter, in: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, ACM, 2012, pp. 2751–2754.

[4] V. Qazvinian, E. Rosengren, D. R. Radev, Q. Mei, Rumor has it: Identifying misinformation in microblogs, in: Proceedings of EMNLP, 2011, pp. 1589–1599.

[5] C. Castillo, M. Mendoza, B. Poblete, Information credibility on twitter, in: Proceedings of the 20th international conference on World wide web, ACM, 2011, pp. 675–684.

[6] S. Kwon, M. Cha, K. Jung, W. Chen, Y. Wang, Prominent features of rumor propagation in online social media, in: 2013 IEEE 13th International Conference on Data Mining, IEEE, 2013, pp. 1103–1108.

[7] S. Vosoughi, Automatic detection and verification of rumors on twitter, Ph.D. thesis (2015).

[8] K. Wu, S. Yang, K. Q. Zhu, False rumors detection on sina weibo by propagation structures, in: 2015 IEEE 31st International Conference on Data Engineering, IEEE, 2015, pp. 651–662.

[9] J. Ma, W. Gao, Z. Wei, Y. Lu, K.-F. Wong, 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, ACM, 2015, pp. 1751–1754.

[10] M. Lukasik, P. Srijith, D. Vu, K. Bontcheva, A. Zubiaga, T. Cohn, Hawkes processes for continuous time sequence classification: an application to rumour stance classification in twitter, in: Proceedings of 54th Annual Meeting of the Association for Computational Linguistics, Association for Computational Linguistics, 2016, pp. 393–398.

[11] X. Liu, A. Nourbakhsh, Q. Li, R. Fang, S. Shah, Real-time rumor debunking on twitter, in: Proceedings of the 24th ACM International on Conference on Information and Knowledge Management, ACM, 2015, pp. 1867–1870.

[12] O. Enayet, S. R. El-Beltagy, NileTMRG at SemEval-2017 Task 8: Determining Rumour and Veracity Support for Rumours on Twitter, in: Proceedings of SemEval, ACL, 2017.

[13] E. Kochkina, M. Liakata, A. Zubiaga, All-in-one: Multi-task learning for rumour verification, in: Proceedings of the 27th International Conference on Computational Linguistics, 2018, pp. 3360–3370.

[14] A. Sen, K. Rudra, S. Ghosh, Extracting situational awareness from microblogs during disaster events, in: Communication Systems and Networks (COMSNETS), 2015 7th International Conference on, IEEE, 2015, pp. 1–6.

[15] S. Dungs, A. Aker, N. Fuhr, K. Bontcheva, Can rumour stance alone predict veracity?, in: Proceedings of the 27th International Conference on Computational Linguistics, 2018, pp. 3360–3370.

[16] S. Kwon, M. Cha, K. Jung, Rumor detection over varying time windows, PLOS ONE 12 (1) (2017) e0168344.

[17] F. Yang, Y. Liu, X. Yu, M. Yang, Automatic detection of rumor on sina weibo, in: Proceedings of the ACM SIGKDD Workshop on Mining Data Semantics, ACM, 2012, p. 13.

[18] S. Wang, T. Terano, Detecting rumor patterns in streaming social media, in: Big Data (Big Data), 2015 IEEE International Conference on, IEEE, 2015, pp. 2709–2715.

[19] W. Chen, C. K. Yeo, C. T. Lau, B. S. Lee, Behavior deviation: An anomaly detection view of rumor preemption, in: Information Technology, Electronics and Mobile Communication Conference (IEMCON), 2016 IEEE 7th Annual, IEEE, 2016, pp. 1–7.

[20] C. Chang, Y. Zhang, C. Szabo, Q. Z. Sheng, Extreme user and political rumor detection on twitter, in: Advanced Data Mining and Applications: 12th International Conference, ADMA 2016, Gold Coast, QLD, Australia, December 12-15, 2016, Proceedings, Springer, 2016, pp. 751–763.

[21] G. Tong, W. Wu, L. Guo, D. Li, C. Liu, B. Liu, D.-Z. Du, An efficient randomized algorithm for rumor blocking in online social networks, arXiv:1701.02368.

[22] L. Derczynski, K. Bontcheva, M. Liakata, R. Procter, G. Wong Sak Hoi, A. Zubiaga, SemEval-2017 Task 8: RumourEval: Determining rumour veracity and support for rumours, in: Proceedings of SemEval, ACL, 2017.

[23] F. Wang, M. Lan, Y. Wu, Ecnu at semeval-2017 task 8: Rumour evaluation using effective features and supervised ensemble models, in: Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017), 2017, pp. 491–496.

[24] V. Singh, S. Narayan, M. S. Akhtar, A. Ekbal, P. Bhattacharya, IITP at SemEval-2017 Task 8: A Supervised Approach for Rumour Evaluation, in: Proceedings of SemEval, 2017.

[25] Y.-C. Chen, Z.-Y. Liu, H.-Y. Kao, IKM at SemEval-2017 Task 8: Convolutional Neural Networks for Stance Detection and Rumor Verification, in: Proceedings of SemEval, ACL, 2017.

[26] A. Srivastava, R. Rehm, J. Moreno Schneider, DFKI-DKT at SemEval-2017 Task 8: Rumour Detection and Classification using Cascading Heuristics, in: Proceedings of SemEval, ACL, 2017, pp. 486–490.

[27] T. Chen, X. Li, H. Yin, J. Zhang, Call attention to rumors: Deep attention based recurrent neural networks for early rumor detection, in: Pacific-Asia Conference on Knowledge Discovery and Data Mining, Springer, 2018, pp. 40–52.

[28] Z. Zhao, P. Resnick, Q. Mei, Enquiring minds: Early detection of rumors in social media from enquiry posts, in: Proceedings of the 24th International Conference on World Wide Web, International World Wide Web Conferences Steering Committee, 2015, pp. 1395–1405.

[29] L. Derczynski, K. Bontcheva, M. Liakata, R. Procter, G. W. S. Hoi, A. Zubiaga, Semeval-2017 task 8: Rumoureval: Determining rumour veracity and support for rumours, arXiv preprint arXiv:1704.05972.

[30] A. Zubiaga, E. Kochkina, M. Liakata, R. Procter, M. Lukasik, K. Bontcheva, T. Cohn, I. Augenstein, Discourse-aware rumour stance classification in social media using sequential classifiers, Information Processing & Management 54 (2) (2018) 273–290.

[31] K. Shu, D. Mahudeswaran, S. Wang, D. Lee, H. Liu, Fakenewsnet: A data repository with news content, social context and dynamic information for studying fake news on social media, arXiv preprint arXiv:1809.01286.

[32] J. Gerhards, M. S. Schäfer, Is the internet a better public sphere? comparing old and new media in the usa and germany, New media & society 12 (1) (2010) 143–160.

[33] K. Starbird, Examining the alternative media ecosystem through the production of alternative narratives of mass shooting events on twitter, in: Eleventh International AAAI Conference on Web and Social Media, 2017.

[34] D. Bahdanau, K. Cho, Y. Bengio, Neural machine translation by jointly learning to align and translate, arXiv preprint arXiv:1409.0473.

[35] B. Wang, K. Liu, J. Zhao, Inner attention based recurrent neural networks for answer selection, in: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Vol. 1, 2016, pp. 1288–1297.

[36] C. Stab, T. Miller, I. Gurevych, Cross-topic argument mining from heterogeneous sources using attention-based neural networks, arXiv preprint arXiv:1802.05758.

[37] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, J. Dean, Distributed representations of words and phrases and their compositionality, in: Advances in neural information processing systems, 2013, pp. 3111–3119.

[38] J. Pennington, R. Socher, C. D. Manning, Glove: Global vectors for word representation., in: EMNLP, Vol. 14, 2014, pp. 1532–1543.

[39] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural computation 9 (8) (1997) 1735–1780.