Discourse-Aware Rumour Stance Classification in Social Media Using Sequential Classifiers

Arkaitz Zubiaga[a], Elena Kochkina[a,b], Maria Liakata[a,b], Rob Procter[a,b], Michal Lukasik[c], Kalina Bontcheva[c], Trevor Cohn[d], Isabelle Augenstein[e]

[a] *University of Warwick, Coventry, UK*
[b] *Alan Turing Institute, London, UK*
[c] *University of Sheffield, Sheffield, UK*
[d] *University of Melbourne, Melbourne, Australia*
[e] *University of Copenhagen, Copenhagen, Denmark*

Abstract

Rumour stance classification, defined as classifying the stance of specific social media posts into one of supporting, denying, querying or commenting on an earlier post, is becoming of increasing interest to researchers. While most previous work has focused on using individual tweets as classifier inputs, here we report on the performance of sequential classifiers that exploit the discourse features inherent in social media interactions or 'conversational threads'. Testing the effectiveness of four sequential classifiers – Hawkes Processes, Linear-Chain Conditional Random Fields (Linear CRF), Tree-Structured Conditional Random Fields (Tree CRF) and Long Short Term Memory networks (LSTM) – on eight datasets associated with breaking news stories, and looking at different types of local and contextual features, our work sheds new light on the development of accurate stance classifiers. We show that sequential classifiers that exploit the use of discourse properties in social media conversations while using only local features, outperform non-sequential classifiers. Furthermore, we show that LSTM using a reduced set of features can outperform the other sequential classifiers; this performance is consistent across datasets and across types of stances. To conclude, our work also analyses the different features under study, identifying those that best help characterise and distinguish between stances, such as supporting tweets being more likely to be accompanied by evidence than denying tweets. We also

*Corresponding author
Email address: a.zubiaga@warwick.ac.uk (Arkaitz Zubiaga)

set forth a number of directions for future research.

Keywords: stance classification, social media, breaking news, veracity classification

1. Introduction

Social media platforms have established themselves as important sources for learning about the latest developments in breaking news. People increasingly use social media for news consumption [1, 2, 3], while media professionals, such as journalists, increasingly turn to social media for news gathering [4] and for gathering potentially exclusive updates from eyewitnesses [5, 6]. Social media platforms such as Twitter are a fertile and prolific source of breaking news, occasionally even outpacing traditional news media organisations [7]. This has led to the development of multiple data mining applications for mining and discovering events and news from social media [8, 9]. However, the use of social media also comes with the caveat that some of the reports are necessarily rumours at the time of posting, as they have yet to be corroborated and verified [10, 11, 12]. The presence of rumours in social media has hence provoked a growing interest among researchers for devising ways to determine veracity in order to avoid the diffusion of misinformation [13].

Resolving the veracity of social rumours requires the development of a rumour classification system and we described in [14], a candidate architecture for such a system consisting of the following four components: (1) detection, where emerging rumours are identified, (2) tracking, where those rumours are monitored to collect new related tweets, (3) stance classification, where the views expressed by different tweet authors are classified, and (4) veracity classification, where knowledge garnered from the stance classifier is put together to determine the likely veracity of a rumour.

In this work we focus on the development of the third component, a stance classification system, which is crucial to subsequently determining the veracity of the underlying rumour. The stance classification task consists in determining how individual posts in social media observably orientate to the postings of others [15, 16]. For instance, a post replying with "no, that's definitely false" is *denying* the preceding claim, whereas "yes, you're right" is *supporting* it. It has been argued that aggregation of the distinct stances evident in the multiple tweets discussing a rumour could help in determining its likely veracity, providing, for example, the means to flag highly disputed rumours as being potentially false [10].

This approach has been justified by recent research that has suggested that the aggregation of the different stances expressed by users can be used for determining the veracity of a rumour [13, 17].

In this work we examine in depth the use of so-called sequential approaches to the rumour stance classification task. Sequential classifiers are able to utilise the discursive nature of social media [6], learning from how 'conversational threads' evolve for a more accurate classification of the stance of each tweet. The use of sequential classifiers to model the conversational properties inherent in social media threads is still in its infancy. For example, in preliminary work we showed that a sequential classifier modelling the temporal sequence of tweets outperforms standard classifiers [18, 19]. Here we extend this preliminary experimentation in four different directions that enable exploring further the stance classification task using sequential classifiers: (1) we perform a comparison of a range of sequential classifiers, including a Hawkes Process classifier, a Linear CRF, a Tree CRF and an LSTM; (2) departing from the use of only local features in our previous work, we also test the utility of contextual features to model the conversational structure of Twitter threads; (3) we perform a more exhaustive analysis of the results looking into the impact of different datasets and the depth of the replies in the conversations on the classifiers' performance, as well as performing an error analysis; and (4) we perform an analysis of features that gives insight into what characterises the different kinds of stances observed around rumours in social media. To the best of our knowledge, dialogical structures in Twitter have not been studied in detail before for classifying each of the underlying tweets and our work is the first to evaluate it exhaustively for stance classification. Twitter conversational threads are identifiable by the relational features that emerge as users respond to each others' postings, leading to tree-structured interactions. The motivation behind the use of these dialogical structures for determining stance is that users' opinions are expressed and evolve in a discursive manner, and that they are shaped by the interactions with other users.

The work presented here advances research in rumour stance classification by performing an exhaustive analysis of different approaches to this task. In particular, we make the following contributions:

- We perform an analysis of whether and the extent to which use of the sequential structure of conversational threads can improve stance classification in comparison to a classifier that determines a tweet's stance from the tweet in isolation. To do so, we evaluate the effectiveness of a range of sequential classifiers: (1) a state-of-the-art classifier that uses Hawkes Pro-

cesses to model the temporal sequence of tweets [18]; (2) two different variants of Conditional Random Fields (CRF), i.e., a linear-chain CRF and a tree CRF; and (3) a classifier based on Long Short Term Memory (LSTM) networks. We compare the performance of these sequential classifiers with non-sequential baselines, including the non-sequential equivalent of CRF, a Maximum Entropy classifier.

- We perform a detailed analysis of the results broken down by dataset and by depth of tweet in the thread, as well as an error analysis to further understand the performance of the different classifiers. We complete our analysis of results by delving into the features, and exploring whether and the extent to which they help characterise the different types of stances.

Our results show that sequential approaches do perform substantially better in terms of macro-averaged F1 score, proving that exploiting the dialogical structure improves classification performance. Specifically, the LSTM achieves the best performance in terms of macro-averaged F1 scores, with a performance that is largely consistent across different datasets and different types of stances. Our experiments show that LSTM performs especially well when only local features are used, as compared to the rest of the classifiers, which need to exploit contextual features to achieve comparable – yet still inferior – performance scores. Our findings reinforce the importance of leveraging conversational context in stance classification. Our research also sheds light on open research questions that we suggest should be addressed in future work. Our work here complements other components of a rumour classification system that we implemented in the PHEME project, including a rumour detection component [20, 21], as well as a study into the diffusion of and reactions to rumour [22].

2. Related Work

Stance classification is applied in a number of different scenarios and domains, usually aiming to classify stances as one of "in favour" or "against". This task has been studied in political debates [23, 24], in arguments in online fora [25, 26] and in attitudes towards topics of political significance [27, 28, 29]. In work that is closer to our objectives, stance classification has also been used to help determine the veracity of information in micro-posts [16], often referred to as *rumour stance classification* [30, 18, 12, 19]. The idea behind this task is that the aggregation of distinct stances expressed by users in social media can be used to

assist in deciding if a report is actually true or false [13]. This may be particularly useful in the context of rumours emerging during breaking news stories, where reports are released piecemeal and which may be lacking authoritative review; in consequence, using the 'wisdom of the crowd' may provide a viable, alternative approach. The types of stances observed while rumours circulate, however, tend to differ from the original "in favour/against", and different types of stances have been discussed in the literature, as we review next.

Rumour stance classification of tweets was introduced in early work by Qazvinian et al. [16]. The line of research initiated by [16] has progressed substantially with revised definitions of the task and hence the task tackled in this paper differs from this early work in a number of aspects. Qazvinian et al. [16] performed 2-way classification of each tweet as *supporting* or *denying* a long-standing rumour such as disputed beliefs that *Barack Obama is reportedly Muslim*. The authors used tweets observed in the past to train a classifier, which was then applied to new tweets discussing the same rumour. In recent work, rule-based methods have been proposed as a way of improving on Qazvinian et al.'s baseline method; however, rule-based methods are likely to be difficult – if not impossible – to generalise to new, unseen rumours. Hamidian et al. [31] extended that work to analyse the extent to which a model trained from historical tweets could be used for classifying new tweets discussing the same rumour.

The work we present here has three different objectives towards improving stance classification. First, we aim to classify the stance of tweets towards rumours that emerge while breaking news stories unfold; these rumours are unlikely to have been observed before and hence rumours from previously observed events, which are likely to diverge, need to be used for training. As far as we know, only work by Lukasik et al. [30, 32, 18] has tackled stance classification in the context of breaking news stories applied to new rumours. Zeng et al. [33] have also performed stance classification for rumours around breaking news stories, but overlapping rumours were used for training and testing. Augenstein et al. [27, 29] studied stance classification of unseen events in tweets, but ignored the conversational structure. Second, recent research has proposed that a 4-way classification is needed to encompass responses seen in breaking news stories [12, 22]. Moving away from the 2-way classification above, which [12] found to be limited in the context of rumours during breaking news, we adopt this expanded scheme to include tweets that are *supporting*, *denying*, *querying* or *commenting* rumours. This adds more categories to the scheme used in early work, where tweets would only support or deny a rumour, or where a distinction between querying and commenting is not made [27, 28, 29]. Moreover, our approach takes into account the

interaction between users on social media, whether it is about appealing for more information in order to corroborate a rumourous statement (*querying*) or to post a response that does not contribute to the resolution of the rumour's veracity (*commenting*). Finally – and importantly – instead of dealing with tweets as single units in isolation, we exploit the emergent structure of interactions between users on Twitter, building a classifier that learns the dynamics of stance in tree-structured conversational threads by exploiting its underlying interactional features. While these interactional features do not, in the final analysis, map directly onto those of conversation as revealed by Conversation Analysis [34], we argue that there are sufficient relational similarities to justify this approach [35]. The closest work is by Ritter et al. [36] who modelled linear sequences of replies in Twitter conversational threads with Hidden Markov Models for dialogue act tagging, but the tree structure of the thread as a whole was not exploited.

As we were writing this article, we also organised, in parallel, a shared task on rumour stance classification, RumourEval [37], at the well-known natural language processing competition SemEval 2017. The subtask A consisted in stance classification of individual tweets discussing a rumour within a conversational thread as one of *support*, *deny*, *query*, or *comment*, which specifically addressed the task presented in this paper. Eight participants submitted results to this task, including work by [38] using an LSTM classifier which is being also analysed in this paper. In this shared task, most of the systems viewed this task as a 4-way single tweet classification task, with the exception of the best performing system by [38], as well as the systems by [39] and [40]. The winning system addressed the task as a sequential classification problem, where the stance of each tweet takes into consideration the features and labels of the previous tweets. The system by Singh et al. [40] takes as input pairs of source and reply tweets, whereas Wang et al. [39] addressed class imbalance by decomposing the problem into a two step classification task, first distinguishing between comments and non-comments, to then classify non-comment tweets as one of support, deny or query. Half of the systems employed ensemble classifiers, where classification was obtained through majority voting [39, 41, 42, 43]. In some cases the ensembles were hybrid, consisting both of machine learning classifiers and manually created rules with differential weighting of classifiers for different class labels [39, 41, 43]. Three systems used deep learning, with [38] employing LSTMs for sequential classification, Chen et al. [44] used convolutional neural networks (CNN) for obtaining the representation of each tweet, assigned a probability for a class by a softmax classifier and García Lozano et al. [41] used CNN as one of the classifiers in their hybrid conglomeration. The remaining two systems by Enayet et al. [45] and Singh et al.

[40] used support vector machines with a linear and polynomial kernel respectively. Half of the systems invested in elaborate feature engineering, including cue words and expressions denoting Belief, Knowledge, Doubt and Denial [42] as well as Tweet domain features, including meta-data about users, hashtags and event specific keywords [39, 42, 40, 45]. The systems with the least elaborate features were Chen et al. [44] and García Lozano et al. [41] for CNNs (word embeddings), Srivastava et al. [43] (sparse word vectors as input to logistic regression) and Kochkina et al. [38] (average word vectors, punctuation, similarity between word vectors in current tweet, source tweet and previous tweet, presence of negation, picture, URL). Five out of the eight systems used pre-trained word embeddings, mostly Google News word2vec embeddings[1], whereas [39] used four different types of embeddings. The winning system used a sequential classifier, however the rest of the participants opted for other alternatives.

To the best of our knowledge Twitter conversational thread structure has not been explored in detail in the stance classification problem. Here we extend the experimentation presented in our previous work using Conditional Random Fields for rumour stance classification [19] in a number of directions: (1) we perform a comparison of a broader range of classifiers, including state-of-the-art rumour stance classifiers such as Hawkes Processes introduced by Lukasik et al. [18], as well as a new LSTM classifier, (2) we analyse the utility of a larger set of features, including not only local features as in our previous work, but also contextual features that further model the conversational structure of Twitter threads, (3) we perform a more exhaustive analysis of the results, and (4) we perform an analysis of features that gives insight into what characterises the different kinds of stances observed around rumours in social media.

3. Research Objectives

The main objective of our research is to analyse whether, the extent to which and how the sequential structure of social media conversations can be exploited to improve the classification of the stance expressed by different posts towards the topic under discussion. Each post in a conversation makes its own contribution to the discussion and hence has to be assigned its own stance value. However, posts in a conversation contribute to previous posts, adding up to a discussion attempting to reach a consensus. Our work looks into the exploitation of this evolving nature

[1] https://github.com/mmihaltz/word2vec-GoogleNews-vectors

of social media discussions with the aim of improving the performance of a stance classifier that has to determine the stance of each tweet. We set forth the following six research objectives:

RO 1. *Quantify performance gains of using sequential classifiers compared with the use of non-sequential classifiers.*

Our first research objective aims to analyse how the use of a sequential classifier that models the evolving nature of social media conversations can perform better than standard classifiers that treat each post in isolation. We do this by solely using local features to represent each post, so that the analysis focuses on the benefits of the sequential classifiers.

RO 2. *Quantify the performance gains using contextual features extracted from the conversation.*

With our second research objective we are interested in analysing whether the use of contextual features (i.e. using other tweets surrounding in a conversation to extract the features of a given tweet) are helpful to boost the classification performance. This is particularly interesting in the case of tweets as they are very short, and inclusion of features extracted from surrounding tweets would be especially helpful. The use of contextual features is motivated by the fact that tweets in a discussion are adding to each other, and hence they cannot be treated alone.

RO 3. *Evaluate the consistency of classifiers across different datasets.*

Our aim is to build a stance classifier that will generalise to multiple different datasets comprising data belonging to different events. To achieve this, we evaluate our classifiers on eight different events.

RO 4. *Assess the effect of the depth of a post in its classification performance.*

We want to build a classifier that will be able to classify stances of different posts occurring at different levels of depth in a conversation. A post can be from a source tweet that initiates a conversation, to a nested reply that occurs later in the sequence formed by a conversational thread. The difficulty increases as replies are deeper as there is more preceding conversation to be aggregated for the classification task. We assess the performance over different depths to evaluate this.

RO 5. *Perform an error analysis to assess when and why each classifier performs best.*

We want to look at the errors made by each of the classifiers. This will help us understand when we are doing well and why, as well as in what cases and with which types of labels we need to keep improving.

RO 6. *Perform an analysis of features to understand and characterise stances in social media discussions.*

In our final objective we are interested in performing an exploration of different features under study, which is informative in two different ways. On the one hand, to find out which features are best for a stance classifier and hence improve performance; on the other hand, to help characterise the different types of stances and hence further understand how people respond in social media discussions.

4. Rumour Stance Classification

In what follows we formally define the rumour stance classification task, as well as the datasets we use for our experiments.

4.1. Task Definition

The rumour stance classification task consists in determining the type of orientation that each individual post expresses towards the disputed veracity of a rumour. We define the rumour stance classification task as follows: we have a set of conversational threads, each discussing a rumour, $D = \{C_1, ..., C_n\}$. Each conversational thread C_j has a variably sized set of tweets $|C_j|$ discussing it, with a source tweet (the root of the tree), $t_{j,1}$, that initiates it. The source tweet $t_{j,1}$ can receive replies by a varying number k of tweets $Replies_{t_{j,1}} = \{t_{j,1,1}, ..., t_{j,1,k}\}$, which can in turn receive replies by a varying number k of tweets, e.g., $Replies_{t_{j,1,1}} = \{t_{j,1,1,1}, ..., t_{j,1,1,k}\}$, and so on. An example of a conversational thread is shown in Figure 1.

The task consists in determining the stance of each of the tweets t_j as one of $Y = \{supporting, denying, querying, commenting\}$.

4.2. Dataset

As part of the PHEME project [13], we collected a rumour dataset associated with eight events corresponding to breaking news events [22].[2] Tweets in this dataset include tree-structured conversations, which are initiated by a tweet about a rumour (source tweet) and nested replies that further discuss the rumour circulated by the source tweet (replying tweets). The process of collecting the tree-structured conversations initiated by rumours, i.e. having a rumour discussed in the source tweet, and associated with the breaking news events under study was conducted with the assistance of journalist members of the Pheme project team.

[2]The entire dataset included nine events, but here we describe the eight events with tweets in English, which we use for our classification experiments. The ninth dataset with tweets in German was not considered for this work.

[depth=0] **u1:** These are not timid colours; soldiers back guarding Tomb of Unknown Soldier after today's shooting #StandforCanada –PICTURE– **[support]**

[depth=1] **u2:** @u1 Apparently a hoax. Best to take Tweet down. **[deny]**

[depth=1] **u3:** @u1 This photo was taken this morning, before the shooting. **[deny]**

[depth=1] **u4:** @u1 I don't believe there are soldiers guarding this area right now. **[deny]**

[depth=2] **u5:** @u4 wondered as well. I've reached out to someone who would know just to confirm that. Hopefully get response soon. **[comment]**

[depth=3] **u4:** @u5 ok, thanks. **[comment]**

Figure 1: Example of a tree-structured thread discussing the veracity of a rumour, where the label associated with each tweet is the target of the rumour stance classification task.

Tweets comprising the rumourous tree-structured conversations were then annotated for stance using CrowdFlower[3] as a crowdsourcing platform. The annotation process is further detailed in [46].

The resulting dataset includes 4,519 tweets and the transformations of annotations described above only affect 24 tweets (0.53%), i.e., those where the source tweet denies a rumour, which is rare. The example in Figure 1 shows a rumour thread taken from the dataset along with our inferred annotations, as well as how we establish the depth value of each tweet in the thread.

One important characteristic of the dataset, which affects the rumour stance classification task, is that the distribution of categories is clearly skewed towards *commenting* tweets, which account for over 64% of the tweets. This imbalance varies slightly across the eight events in the dataset (see Table 1). Given that we consider each event as a separate fold that is left out for testing, this varying imbalance makes the task more realistic and challenging. The striking imbalance towards *commenting* tweets is also indicative of the increased difficulty with respect to previous work on stance classification, most of which performed binary classification of tweets as supporting or denying, which account for less than 28% of the tweets in our case representing a real world scenario.

5. Classifiers

In this section we describe the different classifiers that we used for our experiments. Our focus is on sequential classifiers, especially looking at classifiers that

[3]https://www.crowdflower.com/

Event	Supporting	Denying	Querying	Commenting	Total
charliehebdo	239 (22.0%)	58 (5.0%)	53 (4.0%)	721 (67.0%)	1,071
ebola-essien	6 (17.0%)	6 (17.0%)	1 (2.0%)	21 (61.0%)	34
ferguson	176 (16.0%)	91 (8.0%)	99 (9.0%)	718 (66.0%)	1,084
germanwings-crash	69 (24.0%)	11 (3.0%)	28 (9.0%)	173 (61.0%)	281
ottawashooting	161 (20.0%)	76 (9.0%)	63 (8.0%)	477 (61.0%)	777
prince-toronto	21 (20.0%)	7 (6.0%)	11 (10.0%)	64 (62.0%)	103
putinmissing	18 (29.0%)	6 (9.0%)	5 (8.0%)	33 (53.0%)	62
sydneysiege	220 (19.0%)	89 (8.0%)	98 (8.0%)	700 (63.0%)	1,107
Total	910 (20.1%)	344 (7.6%)	358 (7.9%)	2,907 (64.3%)	4,519

Table 1: Distribution of categories for the eight events in the dataset.

exploit the discursive nature of social media, which is the case for Conditional Random Fields in two different settings – i.e. Linear CRF and tree CRF – as well as that of a Long Short-Term Memory (LSTM) in a linear setting – Branch LSTM. We also experiment with a sequential classifier based on Hawkes Processes that instead exploits the temporal sequence of tweets and has been shown to achieve state-of-the-art performance [18]. After describing these three types of classifiers, we outline a set of baseline classifiers.

5.1. Hawkes Processes

One approach for modelling arrival of tweets around rumours is based on point processes, a probabilistic framework where tweet occurrence likelihood is modelled using an intensity function over time. Intuitively, higher values of intensity function denote higher likelihood of tweet occurrence. For example, Lukasik et al. modelled tweet occurrences over time with a log-Gaussian Cox Process, a point process which models its intensity function as an exponentiated sample of a Gaussian Process [47, 48, 49]. In related work, tweet arrivals were modelled with a Hawkes Process and a resulting model was applied for stance classification of tweets around rumours [18]. In this subsection we describe the sequence classification algorithm based on Hawkes Processes.

Intensity Function. The intensity function in a Hawkes Process is expressed as a summation of base intensity and the intensities corresponding to influences of previous tweets,

$$\lambda_{y,m}(t) = \mu_y + \sum_{t_\ell < t} \mathbb{I}(m_\ell = m) \alpha_{y_\ell, y} \kappa(t - t_\ell), \tag{1}$$

where the first term represents the constant base intensity of generating label y. The second term represents the influence from the previous tweets. The influence from each tweet is modelled with an exponential kernel function $\kappa(t - t_\ell) = \omega \exp(-\omega(t - t_\ell))$. The matrix α of size $|Y| \times |Y|$ encodes how pairs of labels corresponding to tweets influence one another, e.g. how a *querying* label influences a *rejecting* label.

Likelihood function. The parameters governing the intensity function are learnt by maximising the likelihood of generating the tweets:

$$L(\boldsymbol{t}, \boldsymbol{y}, \boldsymbol{m}, \boldsymbol{W}) = \prod_{n=1}^{N} p(\boldsymbol{W}_n | y_n) \times \Big[\prod_{n=1}^{N} \lambda_{y_n, m_n}(t_n) \Big] \times p(E_T), \tag{2}$$

where the likelihood of generating text given the label is modelled as a multinomial distribution conditioned on the label (parametrised by matrix β). The second term provides the likelihood of occurrence of tweets at times $t_1, \ldots, t_n$ and the third term provides the likelihood that no tweets happen in the interval $[0, T]$ except at times $t_1, \ldots, t_n$. We estimate the parameters of the model by maximising the log-likelihood. As in [18], Laplacian smoothing is applied to the estimated language parameter β.

In one approach to μ and α optimisation (*Hawkes Process with Approximated Likelihood*, or *HP Approx.* [18]) a closed form updates for μ and α are obtained using an approximation of the log-likelihood of the data. In a different approach (*Hawkes Process with Exact Likelihood*, or *HP Grad.* [18]) parameters are found using joint gradient based optimisation over μ and α, using derivatives of log-likelihood[4]. L-BFGS approach is employed for gradient search. Parameters are initialised with those found by the *HP Approx.* method. Moreover, following previous work we fix the decay parameter ω to 0.1.

We predict the most likely sequence of labels, thus maximising the likelihood of occurrence of the tweets from Equation (2), or the approximated likelihood in case of *HP Approx.* Similarly as in [18], we follow a greedy approach, where we choose the most likely label for each consecutive tweet.

[4]For both implementations we used the 'seqhawkes' Python package: `https://github.com/mlukasik/seqhawkes`

5.2. Conditional Random Fields (CRF): Linear CRF and Tree CRF

We use CRF as a structured classifier to model sequences observed in Twitter conversations. With CRF, we can model the conversation as a graph that will be treated as a sequence of stances, which also enables us to assess the utility of harnessing the conversational structure for stance classification. Different to traditionally used classifiers for this task, which choose a label for each input unit (e.g. a tweet), CRF also consider the neighbours of each unit, learning the probabilities of transitions of label pairs to be followed by each other. The input for CRF is a graph $G = (V, E)$, where in our case each of the vertices V is a tweet, and the edges E are relations of tweets replying to each other. Hence, having a data sequence X as input, CRF outputs a sequence of labels Y [50], where the output of each element y_i will not only depend on its features, but also on the probabilities of other labels surrounding it. The generalisable conditional distribution of CRF is shown in Equation 3 [51].

$$p(y|x) = \frac{1}{Z(x)} \prod_{a=1}^{A} \Psi_a(y_a, x_a) \tag{3}$$

where Z(x) is the normalisation constant, and Ψ_a is the set of factors in the graph G.

We use CRFs in two different settings.[5] First, we use a linear-chain CRF (Linear CRF) to model each branch as a sequence to be input to the classifier. We also use Tree-Structured CRFs (Tree CRF) or General CRFs to model the whole, tree-structured conversation as the sequence input to the classifier. So in the first case the sequence unit is a branch and our input is a collection of branches and in the second case our sequence unit is an entire conversation, and our input is a collection of trees. An example of the distinction of dealing with branches or trees is shown in Figure 2. With this distinction we also want to experiment whether it is worthwhile building the whole tree as a more complex graph, given that users replying in one branch might not have necessarily seen and be conditioned by tweets in other branches. However, we believe that the tendency of types of replies observed in a branch might also be indicative of the distribution of types of replies in other branches, and hence useful to boost the performance of the classifier when using the tree as a whole. An important caveat of modelling a tree in branches is also that there is a need to repeat parts of the tree across branches, e.g., the source

[5]We use the PyStruct to implement both variants of CRF [52].

tweet will repeatedly occur as the first tweet in every branch extracted from a tree.[6]

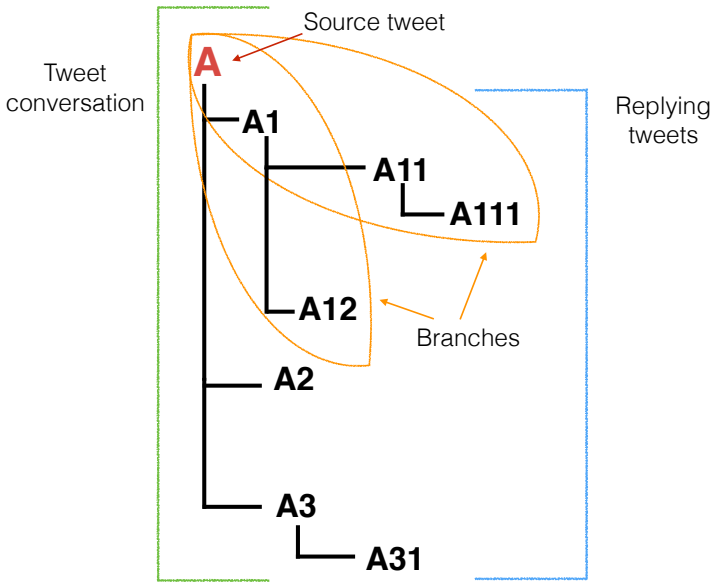

Figure 2: Example of a tree-structured conversation, with two overlapping branches highlighted.

To account for the imbalance of classes in our datasets, we perform cost-sensitive learning by assigning weighted probabilities to each of the classes, these probabilities being the inverse of the number of occurrences observed in the training data for a class.

5.3. Branch LSTM

Another model that works with structured input is a neural network with Long Short-Term Memory (LSTM) units [53]. LSTMs are able to model discrete time series and possess a 'memory' property of the previous time steps, therefore we propose a *branch-LSTM* model that utilises them to process branches of tweets.

Figure 3 illustrates how the input of the time step of the LSTM layer is a vector that is an average of word vectors from each tweet and how the information propagates between time steps.

[6]Despite this also leading to having tweets repeated across branches in the test set and hence producing an output repeatedly for the same tweet with Linear CRF, this output does is consistent and there is no need to aggregate different outputs.

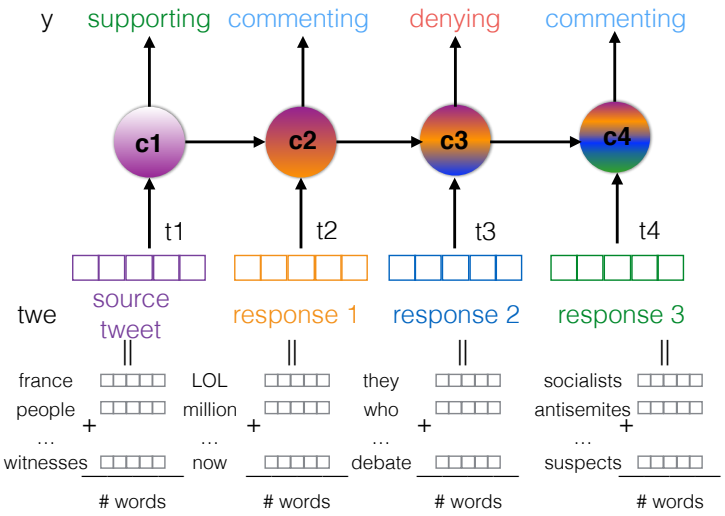

Figure 3: Illustration of the input/output structure of the LSTM-branch model

The full model consists of several LSTM layers that are connected to several feed-forward ReLU layers and a softmax layer to obtain predicted probabilities of a tweet belonging to certain class. As a means for weight regularisation we utilise *dropout* and *l2-norm*. We use categorical cross-entropy as the loss function. The model is trained using mini-batches and the Adam optimisation algorithm [54].[7]

The number of layers, number of units in each layer, regularisation strength, mini-batch size and learning rate are determined using the Tree of Parzen Estimators (TPE) algorithm [57][8] on the development set.[9]

The *branch-LSTM* takes as input tweets represented as the average of its word vectors. We also experimented with obtaining tweet representations through per-word nested LSTM layers, however, this approach did not result in significantly better results than the average of word vectors.

Extracting branches from a tree-structured conversation presents the caveat that some tweets are repeated across branches after this conversion. We solve this issue by applying a mask to the loss function to not take repeated tweets into account.

[7]For implementation of all models we used Python libraries Theano [55] and Lasagne [56].
[8]We use the implementation in the hyperopt package [58].
[9]For this setting, we use the 'Ottawa shooting' event for development.

5.4. Summary of Sequential Classifiers

All of the classifiers described above make use of the sequential nature of Twitter conversational threads. These classifiers take a sequence of tweets as input, where the relations between tweets are formed by replies. If C replies to B, and B replies to A, it will lead to a sequence "A → B → C". Sequential classifiers will use the predictions on preceding tweets to determine the possible label for each tweet. For instance, the classification for B will depend on the prediction that has been previously made for A, and the probabilities of different labels for B will vary for the classifier depending on what has been predicted for A.

Among the four classifiers described above, the one that differs in how the sequence is treated is the Tree CRF. This classifier builds a tree-structured graph with the sequential relationships composed by replying tweets. The rest of the classifiers, Hawkes Processes, Linear CRF and LSTM, will break the entire conversational tree into linear branches, and the input to the classifiers will be linear sequences. The use of a graph with the Tree CRF has the advantage of building a single structure, while the rest of the classifiers building linear sequences inevitably need to repeat tweets across different linear sequences. All the linear sequences will repeatedly start with the source tweet, while some of the subsequent tweets may also be repeated. The use of linear sequences has however the advantages of simplifying the model being used, and one may also hypothesise that inclusion of the entire tree made of different branches into the same graph may not be suitable when they may all be discussing issues that differ to some extent from one another. Figure 2 shows an example of a conversation tree, how the entire tree would make a graph, as well as how we break it down into smaller branches or linear sequences.

5.5. Baseline Classifiers

Maximum Entropy classifier (MaxEnt). As the non-sequential counterpart of CRF, we use a Maximum Entropy (or logistic regression) classifier, which is also a conditional classifier but which will operate at the tweet level, ignoring the conversational structure. This enables us to directly compare the extent to which treating conversations as sequences instead of having each tweet as a separate unit can boost the performance of the CRF classifiers. We perform cost-sensitive learning by assigning weighted probabilities to each class as the inverse of the number of occurrences in the training data.

Additional baselines. We also compare two more non-sequential classifiers[10]: Support Vector Machines (SVM), and Random Forests (RF).

5.6. Experiment Settings and Evaluation Measures

We experiment in an 8-fold cross-validation setting. Given that we have 8 different events in our dataset, we create 8 different folds, each having the data linked to an event. In our cross-validation setting, we run the classifier 8 times, on each occasion having a different fold for testing, with the other 7 for training. In this way, each fold is tested once, and the aggregation of all folds enables experimentation on all events. For each of the events in the test set, the experiments consist in classifying the stance of each individual tweet. With this, we simulate a realistic scenario where we need to use knowledge from past events to train a model that will be used to classify tweets in new events.

Given that the classes are clearly imbalanced in our case, evaluation based on accuracy arguably cannot suffice to capture competitive performance beyond the majority class. To account for the imbalance of the categories, we report the macro-averaged F1 scores, which measures the overall performance assigning the same weight to each category. We aggregate the macro-averaged F1 scores to get the final performance score of a classifier. We also use the McNemar test [59] throughout the analysis of results to further compare the performance of some classifiers.

It is also worth noting that all the sequential classifiers only make use of preceding tweets in the conversation to classify a tweet, and hence no later tweets are used. That is the case of a sequence t_1, t_2, t_3 of tweets, each responding to the preceding tweet. The sequential classifier attempting to classify t_2 would incorporate t_1 in the sequence, but t_3 would not be considered.

6. Features

While focusing on the study of sequential classifiers for discursive stance classification, we perform our experiments with three different types of features: local features, contextual features and Hawkes features. First, local features enable us to evaluate the performance of sequential classifiers in a comparable setting to non-sequential classifiers where features are extracted solely from the current tweet; this makes it a fairer comparison where we can quantify the extent to which

[10] We use their implementation in the scikit-learn Python package, using the *class_weight="balanced"* parameter to perform cost-sensitive learning.

mining sequences can boost performance. In a subsequent step, we also incorporate contextual features, i.e. features from other tweets in a conversation, which enables us to further boost performance of the sequential classifiers. Finally, and to enable comparison with the Hawkes process classifier, we describe the Hawkes features.

Table 2 shows the list of features used, both local and contextual, each of which can be categorised into several subtypes of features, as well as the Hawkes features. For more details on these features, please see Appendix A.

Local features	
Lexicon	Word embeddings POS tags Negation Swear words
Content formatting	Tweet length Word count
Punctuation	Question mark Exclamation mark
Tweet formatting	URL attached
Contextual features	
Relational	Word2Vec similarity wrt source tweet Word2Vec similarity wrt preceding tweet Word2Vec similarity wrt thread
Structural	Is leaf Is source tweet Is source user
Social	Has favourites Has retweets Persistence Time difference
Hawkes features	
Hawkes features	Bag of words Timestamp

Table 2: List of features.

7. Experimental Results

7.1. Evaluating Sequential Classifiers (RO 1)

First, we evaluate the performance of the classifiers by using only local features. As noted above, this enables us to perform a fairer comparison of the different classifiers by using features that can be obtained solely from each tweet in isolation; likewise, it enables us to assess whether and the extent to which the use of a sequential classifier to exploit the discursive structure of conversational threads can be of help to boost performance of the stance classifier while using the same set of features as non-sequential classifiers.

Therefore, in this section we make use of the local features described in Section Appendix A.1. Additionally, we also use the Hawkes features described in Section Appendix A.3 for comparison with the Hawkes processes. For the set of local features, we show the results for three different scenarios: (1) using each subgroup of features alone, (2) in a leave-one-out setting where one of the subgroups is not used, and (3) using all of the subgroups combined.

Table 3 shows the results for the different classifiers using the combinations of local features as well as Hawkes features. We make the following observations from these results:

- LSTM consistently performs very well with different features.
- Confirming our main hypothesis and objective, sequential classifiers do show an overall superior performance to the non-sequential classifiers. While the two CRF alternatives perform very well, the LSTM classifier is slightly superior (the differences between CRF and LSTM results are statistically significant at $p < 0.05$, except for the LF1 features). Moreover, the CRF classifiers outperform their non-sequential counterpart MaxEnt, which achieves an overall lower performance (all the differences between CRF and MaxEnt results being statistically significant at $p < 0.05$).
- The LSTM classifier is, in fact, superior to the Tree CRF classifier (all statistically significant except LF1). While the Tree CRF needs to make use of the entire tree for the classification, the LSTM classifier only uses branches, reducing the amount of data and complexity that needs to be processed in each sequence.
- Among the local features, combinations of subgroups of features lead to clear improvements with respect to single subgroups without combinations.

- Even though the combination of all local features achieves good performance, there are alternative leave-one-out combinations that perform better. The feature combination leading to the best macro-F1 score is that combining lexicon, content formatting and punctuation (i.e. LF123, achieving a score of 0.449).

Summarising, our initial results show that exploiting the sequential properties of conversational threads, while still using only local features to enable comparison, leads to superior performance with respect to the classification of each tweet in isolation by non-sequential classifiers. Moreover, we observe that the local features combining lexicon, content formatting and punctuation lead to the most accurate results. In the next section we further explore the use of contextual features in combination with local features to boost performance of sequential classifiers; to represent the local features, we rely on the best approach from this section (i.e. LF123).

	Macro-F1									
	HF	LF1	LF2	LF3	LF4	LF123	LF124	LF134	LF234	LF1234
SVM	0.336	0.356	0.231	0.258	0.313	0.403	0.365	0.403	0.420	0.408
Random Forest	0.325	0.308	0.276	0.267	**0.437***	0.322	0.310	0.351	0.357	0.329
MaxEnt	0.338	0.363	0.272	0.263	0.428	0.415	0.363	0.421	0.427	0.422
Hawkes-approx	0.309	–	–	–	–	–	–	–	–	–
Hawkes-grad	0.307	–	–	–	–	–	–	–	–	–
Linear CRF	**0.362***	0.357	0.268	0.318	0.317	0.413	0.365	0.403	0.425	0.412
Tree CRF	0.350	**0.375***	0.285	0.221	0.217	0.433	0.385	**0.413**	**0.436***	0.433
LSTM	0.318	0.362	**0.318***	**0.407***	0.419	**0.449***	**0.395***	0.412	0.429	**0.437***

Table 3: Macro-F1 performance results using local features. HF: Hawkes features. LF: local features, where numbers indicate subgroups of features as follows, 1: Lexicon, 2: Content formatting, 3: Punctuation, 4: Tweet formatting. An '*' indicates that the differences between the best performing classifier and the second best classifier for that feature set are statistically significant at $p < 0.05$.

7.2. Exploring Contextual Features (RO 2)

The experiments in the previous section show that sequential classifiers that model discourse, especially the LSTM classifier, can provide substantial improvements over non-sequential classifiers that classify each tweet in isolation, in both cases using only local features to represent each tweet. To complement this, we now explore the inclusion of contextual features described in Section Appendix

A.2 for the stance classification. We perform experiments with four different groups of features in this case, including local features and the three subgroups of contextual features, namely relational features, structural features and social features. As in the previous section, we show results for the use of each subgroup of features alone, in a leave-one-out setting, and using all subgroups of features together.

Table 4 shows the results for the classifiers incorporating contextual features along with local features. We make the following observations from these results:

- The use of contextual features leads to substantial improvements for non-sequential classifiers, getting closer to and even in some cases outperforming some of the sequential classifiers.
- Sequential classifiers, however, do not benefit much from using contextual features. It is important to note that sequential classifiers are taking the surrounding context into consideration when they aggregate sequences in the classification process. This shows that the inclusion of contextual features is not needed for sequential classifiers, given that they are implicitly including context through the use of sequences.
- In fact, for the LSTM, which is still the best-performing classifier, it is better to only rely on local features, as the rest of the features do not lead to any improvements. Again, the LSTM is able to handle context on its own, and therefore inclusion of contextual features is redundant and may be harmful.
- Addition of contextual features leads to substantial improvements for the non-sequential classifiers, achieving similar macro-averaged scores in some cases (e.g. MaxEnt / All vs LSTM / LF). This reinforces the importance of incorporating context in the classification process, which leads to improvements for the non-sequential classifier when contextual features are added, but especially in the case of sequential classifiers that can natively handle context.

Summarising, we observe that the addition of contextual features is clearly useful for non-sequential classifiers, which do not consider context natively. For the sequential classifiers, which natively consider context in the classification process, the inclusion of contextual features is not helpful and is even harmful in most cases, potentially owing to the contextual information being used twice. Still, sequential classifiers, and especially LSTM, are the best classifiers to achieve optimal results, which also avoid the need for computing contextual features.

Macro-F1									
	LF	R	ST	SO	LF+R+ST	LF+R+SO	LF+ST+SO	R+ST+SO	All
SVM	0.403	**0.335***	**0.318**	0.260	0.429	0.347	0.388	0.295	0.375
Random Forest	0.322	0.325	0.269	0.328	0.356	0.358	0.376	**0.343***	0.364
MaxEnt	0.415	0.333	**0.318**	0.310	0.434	**0.447**	0.447	0.318	**0.449**
Linear CRF	0.413	0.318	**0.318**	**0.334***	0.424	0.431	0.431	0.342	0.437
Tree CRF	0.433	0.322	0.317	0.312	0.425	0.429	0.430	0.232	0.433
LSTM	**0.449***	0.318	**0.318**	0.315	**0.445***	0.436	**0.448**	0.314	0.437

Table 4: Macro-F1 performance results incorporating contextual features. LF: local features, R: relational features, ST: structural features, SO: social features. An '*' indicates that the differences between the best performing classifier and the second best classifier for that feature set are statistically significant.

7.3. Analysis of the Best-Performing Classifiers

Despite the clear superiority of LSTM with the sole use of local features, we now further examine the results of the best-performing classifiers to understand when they perform well. We compare the performance of the following five classifiers in this section: (1) LSTM with only local features, (2) Tree CRF with all the features, (3) Linear CRF with all the features, (4) MaxEnt with all the features, and (5) SVM using local features, relational and structural features. Note that while for LSTM we only need local features, for the rest of the classifiers we need to rely on all or almost all of the features. For these best-performing combinations of classifiers and features, we perform additional analyses by event and by tweet depth, and perform an analysis of errors.

7.3.1. Evaluation by Event (RO 3)

The analysis of the best-performing classifiers, broken down by event, is shown in Table 5. These results suggest that there is not a single classifier that performs best in all cases; this is most likely due to the diversity of events. However, we see that the LSTM is the classifier that outperforms the rest in the greater number of cases; this is true for three out of the eight cases (the difference with respect to the second best classifier being always statistically significant). Moreover, sequential classifiers perform best in the majority of the cases, with only three cases where a non-sequential classifier performs best. Most importantly, these results suggest that sequential classifiers outperform non-sequential classifiers across the different events under study, with LSTM standing out as a classifier that performs best in numerous cases using only local features.

Macro-F1								
	CH	Ebola	Ferg.	GW crash	Ottawa	Prince	Putin	Sydney
SVM	0.399	0.380	0.382	0.427	**0.492**	0.491	0.509	0.427
MaxEnt	0.446	0.425	**0.418**	0.475	0.468	**0.514**	0.381	0.443
Linear CRF	0.443	0.619	0.380	0.470	0.412	0.512	**0.528**	**0.454**
Tree CRF	0.457	0.557	0.356	0.523	0.441	0.505	0.491	0.426
LSTM	**0.465**	**0.657**	0.373	**0.543**	0.475	0.379	0.457	0.446

Table 5: Macro-F1 results for the best-performing classifiers, broken down by event.

7.3.2. Evaluation by Tweet Depth (RO 4)

The analysis of the best-performing classifiers, broken down by depth of tweets, is shown in Table 6. Note that the depth of the tweet reflects, as shown in Figure 1, the number of steps from the source tweet to the current tweet. We show results for all the depths from 0 to 4, as well as for the subsequent depths aggregated as 5+.

Again, we see that there is not a single classifier that performs best for all depths. We see, however, that sequential classifiers (Linear CRF, Tree CRF and LSTM) outperform non-sequential classifiers (SVM and MaxEnt) consistently. However, the best sequential classifier varies. While LSTM is the best-performing classifier overall when we look at macro-averaged F1 scores, as shown in Section 7.2, surprisingly it does not achieve the highest macro-averaged F1 scores at any depth. It does, however, perform well for each depth compared to the rest of the classifiers, generally being close to the best classifier in that case. Its consistently good performance across different depths makes it the best overall classifier, despite only using local features.

Tweets by depth						
	0	1	2	3	4	5+
Counts	297	2,602	553	313	195	595
Macro-F1						
	0	1	2	3	4	5+
SVM	0.272	0.368	0.298	0.314	0.331	0.274
MaxEnt	0.238	0.385	0.286	0.279	**0.369**	**0.290**
Linear CRF	**0.286**	0.394	**0.306**	0.282	0.271	0.266
Tree CRF	0.278	**0.404**	0.280	**0.331**	0.230	0.237
LSTM	0.271	0.381	0.298	0.274	0.307	0.286

Table 6: Macro-F1 results for the best-performing classifiers, broken down by tweet depth.

7.3.3. Error Analysis (RO 5)

To analyse the errors that the different classifiers are making, we look at the confusion matrices in Table 7. If we look at the correct guesses, highlighted in bold in the diagonals, we see that the LSTM clearly performs best for three of the categories, namely *support*, *deny* and *query*, and it is just slightly behind the other classifiers for the majority class, *comment*. Besides LSTM's overall superior performance as we observed above, this also confirms that the LSTM is doing better than the rest of the classifiers in dealing with the imbalance inherent in our datasets. For instance, the *Deny* category proves especially challenging for being less common than the rest (only 7.6% of instances in our datasets); the LSTM still achieves the highest performance for this category, which, however, only achieves 0.212 in accuracy and may benefit from having more training instances.

We also notice that a large number of instances are misclassified as *comments*, due to this being the prevailing category and hence having a much larger number of training instances. One could think of balancing the training instances to reduce the prevalence of *comments* in the training set, however, this is not straightforward for sequential classifiers as one needs to then break sequences, losing not only some instances of *comments*, but also connections between instances of other categories that belong to those sequences. Other solutions, such as labelling more data or using more sophisticated features to distinguish different categories, might be needed to deal with this issue; given that the scope of this paper is to assess whether and the extent to which sequential classifiers can be of help in stance classification, further tackling this imbalance is left for future work.

7.4. Feature Analysis (RO 6)

To complete the analysis of our experiments, we now look at the different features we used in our study and perform an analysis to understand how distinctive the different features are for the four categories in the stance classification problem. We visualise the different distributions of features for the four categories in beanplots [60]. We show the visualisations pertaining to 16 of the features under study in Figure 4. This analysis leads us to some interesting observations towards characterising the different types of stances:

- As one might expect, *querying tweets* are more likely to have question marks.

SVM				
	Support	Deny	Query	Comment
Support	**0.657**	0.041	0.018	0.283
Deny	0.185	**0.129**	0.107	0.579
Query	0.083	0.081	**0.343**	0.494
Comment	0.150	0.075	0.053	**0.723**
MaxEnt				
	Support	Deny	Query	Comment
Support	**0.794**	0.044	0.003	0.159
Deny	0.156	**0.130**	0.079	0.634
Query	0.088	0.066	**0.366**	0.480
Comment	0.152	0.074	0.048	**0.726**
Linear CRF				
	Support	Deny	Query	Comment
Support	**0.603**	0.048	0.013	0.335
Deny	0.219	**0.140**	0.050	0.591
Query	0.071	0.095	**0.357**	0.476
Comment	0.139	0.072	0.062	**0.726**
Tree CRF				
	Support	Deny	Query	Comment
Support	**0.552**	0.066	0.019	0.363
Deny	0.145	**0.169**	0.081	0.605
Query	0.077	0.081	**0.401**	0.441
Comment	0.128	0.074	0.068	**0.730**
LSTM				
	Support	Deny	Query	Comment
Support	**0.825**	0.046	0.003	0.127
Deny	0.225	**0.212**	0.125	0.438
Query	0.090	0.087	**0.432**	0.390
Comment	0.144	0.076	0.057	**0.723**

Table 7: Confusion matrices for the best-performing classifiers.

- Interestingly, *supporting tweets* tend to have a higher similarity with respect to the source tweet, indicating that the similarity based on word embeddings can be a good feature to identify those tweets.

- *Supporting tweets* are more likely to come from the user who posted the source tweet.

- *Supporting tweets* are more likely to include links, which is likely indicative of tweets pointing to evidence that supports their position.

- Looking at the delay in time of different types of tweets (i.e., the *time dif-*

ference feature), we see that *supporting*, *denying* and *querying tweets* are more likely to be observed only in the early stages of a rumour, while later tweets tend to be mostly comments. In fact, these suggests that discussion around the veracity of a rumour occurs especially in the period just after it is posted, whereas the conversation then evolves towards comments that do not discuss the veracity of the rumour in question.

- *Denying tweets* are more likely to use negating words. However, negations are also used in other kinds of tweets to a lesser extent, which also makes it more complicated for the classifiers to identify denying tweets. In addition to the low presence of denying tweets in the datasets, the use of negations also in other kinds of responses makes it more challenging to classify them. A way to overcome this may be to use more sophisticated approaches to identify negations that are rebutting the rumour initiated in the source tweet, while getting rid of the rest of the negations.

- When we look at the extent to which users persist in their participation in a conversational thread (i.e., the *persistence* feature), we see that users tend to participate more when they are posting *supporting tweets*, showing that users especially insistent when they support a rumour. However, we observe a difference that is not highly remarkable in this particular case.

The rest of the features do not show a clear tendency that helps visually distinguish characteristics of the different types of responses. While some features like swear words or exclamation marks may seem indicative of how they orient to somebody else's earlier post, there is no clear difference in reality in our datasets. The same is true for social features like retweets or favourites, where one may expect, for instance, that denying tweets may attract more retweets than comments, as people may want to let others know about rebuttals; the distributions of retweets and favourites are, however, very similar for the different categories.

One possible concern from this analysis is that there are very few features that characterise *commenting tweets*. In fact, the only feature that we have identified as being clearly distinct for *comments* is the *time difference*, given that they are more likely to appear later in the conversations. This may well help classify those late *comments*, however, early comments will be more difficult to be classified based on that feature. Finding additional features to distinguish *comments* from the rest of the tweets may be of help for improving the overall classification.

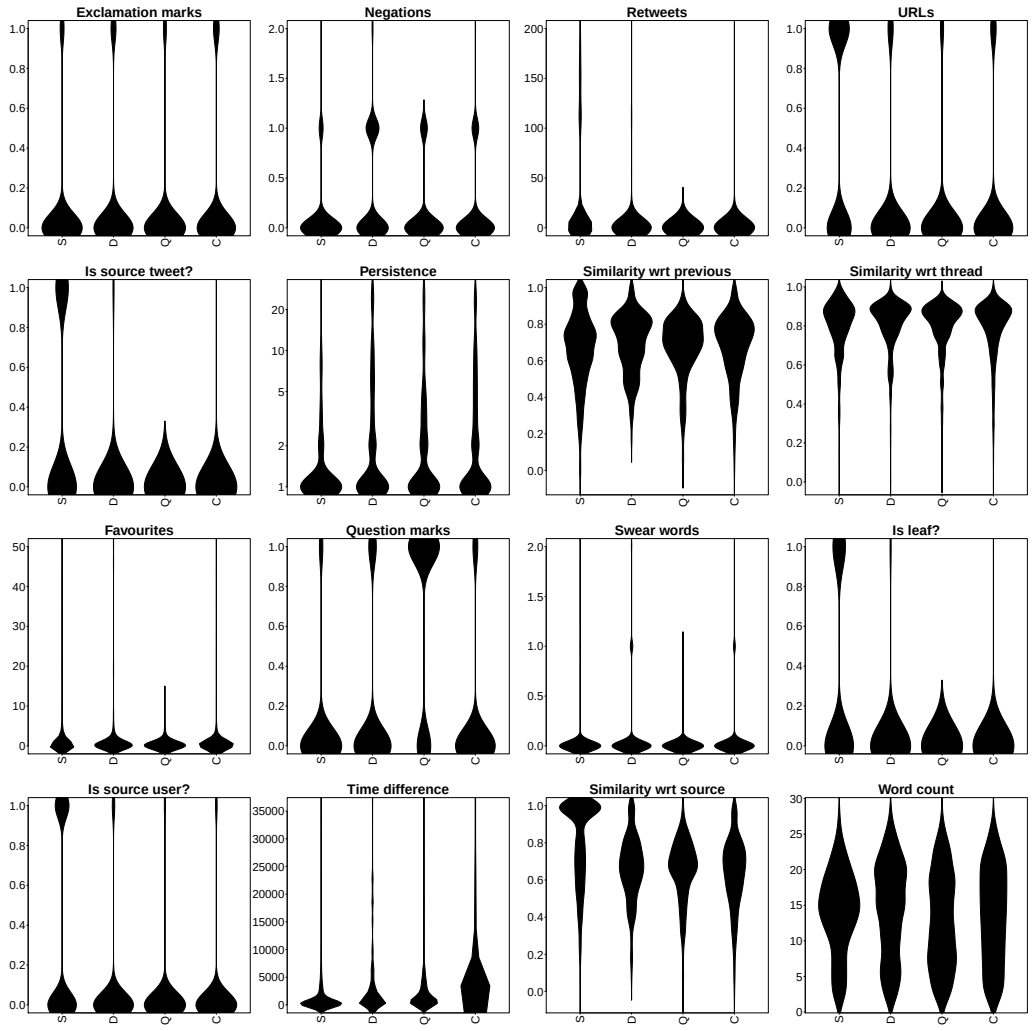

Figure 4: Distributions of feature values across the four categories: Support, Deny, Query and Comment.

8. Conclusions and Future Work

While discourse and sequential structure of social media conversations have been barely explored in previous work, our work has performed an analysis on the use of different sequential classifiers for the rumour stance classification task. Our work makes three core contributions to existing work on rumour stance classification: (1) we focus on the stance of tweets towards rumours that emerge while breaking news stories unfold; (2) we broaden the stance types considered in previous work to encompass all types of responses observed during breaking news, performing a 4-way classification task; and (3) instead of dealing with tweets as single units in isolation, we exploit the emergent structure of interactions between users on Twitter. In this task, a classifier has to determine if each tweet is supporting, denying, querying or commenting on a rumour's truth value. We mine the sequential structure of Twitter conversational threads in the form of users' replies to one another, extending existing approaches that treat each tweet as a separate unit. We have used four different sequential classifiers: (1) a Hawkes Process classifier that exploits temporal sequences, which showed state-of-the-art performance [18]; (2) a linear-chain CRF modelling tree-structured conversations broken down into branches; (3) a tree CRF modelling them as a graph that includes the whole tree; and (4) an LSTM classifier that also models the conversational threads as branches. These classifiers have been compared with a range of baseline classifiers, including the non-sequential equivalent Maximum Entropy classifier, on eight Twitter datasets associated with breaking news.

While previous stance detection work had mostly limited classifiers to looking at tweets as single units, we have shown that exploiting the discursive characteristics of interactions on Twitter, by considering probabilities of transitions within tree-structured conversational threads, can lead to substantial improvements. Among the sequential classifiers, our results show that the LSTM classifier using a more limited set of features performs the best, thanks to its ability to natively handle context, as well as only relying on branches instead of the whole tree, which reduces the amount of data and complexity that needs to be processed in each sequence. The LSTM has been shown to perform consistently well across datasets, as well as across different types of stances. Besides the comparison of classifiers, our analysis also looks at the distributions of the different features under study as well as how well they characterise the different types of stances. This enables us both to find out which features are the most useful, as well as to suggest improvements needed in future work for improving stance classifiers.

To the best of our knowledge, this is the first attempt at aggregating the conver-

sational structure of Twitter threads to produce classifications at the tweet level. Besides the utility of mining sequences from conversational threads for stance classification, we believe that our results will, in turn, encourage the study of sequential classifiers applied to other natural language processing and data mining tasks where the output for each tweet can benefit from the structure of the entire conversation, e.g., sentiment analysis [61, 62, 63, 64, 65, 66], tweet geolocation [67, 68], language identification [69, 70], event detection [71] and analysis of public perceptions on news [72, 73] and other issues [74, 75].

Our plans for future work include further developing the set of features that characterise the most challenging and least-frequent stances, i.e., denying tweets and querying tweets. These need to be investigated as part of a more detailed and interdisciplinary, thematic analysis of threads [6, 76, 77]. We also plan to develop an LSTM classifier that mines the entire conversation as a single tree. Our approach assumes that rumours have been already identified or input by a human, hence a final and ambitious aim for future work is the integration with our rumour detection system [20], whose output would be fed to the stance classification system. The output of our stance classification will also be integrated with a veracity classification system, where the aggregation of stances observed around a rumour will be exploited to determine the likely veracity of the rumour.

Acknowledgments

This work has been supported by the PHEME FP7 project (grant No. 611233), the EPSRC Career Acceleration Fellowship EP/I004327/1, Elsevier through the UCL Big Data Institute, and The Alan Turing Institute under the EPSRC grant EP/N510129/1.

References

[1] A. Hermida, F. Fletcher, D. Korell, D. Logan, Share, like, recommend: Decoding the social media news consumer, Journalism Studies 13 (5-6) (2012) 815–824.

[2] A. Mitchell, J. Gottfried, K. Matsa, Millennials and political news: Social media – the local tv for the next generation?, Tech. rep., Pew Research Center (2015).

[3] A. Zubiaga, D. Spina, R. Martinez, V. Fresno, Real-time classification of twitter trends, Journal of the Association for Information Science and Technology 66 (3) (2015) 462–473.

[4] A. Zubiaga, H. Ji, K. Knight, Curating and contextualizing twitter stories to assist with social newsgathering, in: Proceedings of the 2013 international conference on Intelligent User Interfaces, IUI, ACM, 2013, pp. 213–224.

[5] N. Diakopoulos, M. De Choudhury, M. Naaman, Finding and assessing social media information sources in the context of journalism, in: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI, ACM, 2012, pp. 2451–2460.

[6] P. Tolmie, R. Procter, M. Rouncefield, M. Liakata, A. Zubiaga, Microblog analysis as a programme of work, ACM Transactions on Social Computing (To Appear).

[7] H. Kwak, C. Lee, H. Park, S. Moon, What is twitter, a social network or a news media?, in: Proceedings of the 19th International Conference on World Wide Web, WWW, ACM, 2010, pp. 591–600.

[8] X. Dong, D. Mavroeidis, F. Calabrese, P. Frossard, Multiscale event detection in social media, Data Mining and Knowledge Discovery 29 (5) (2015) 1374–1405.

[9] G. Stilo, P. Velardi, Efficient temporal mining of micro-blog texts and its application to event discovery, Data Mining and Knowledge Discovery 30 (2) (2016) 372–402.

[10] M. Mendoza, B. Poblete, C. Castillo, Twitter under crisis: can we trust what we rt?, in: Proceedings of the first workshop on social media analytics, ACM, 2010, pp. 71–79.

[11] R. Procter, J. Crump, S. Karstedt, A. Voss, M. Cantijoch, Reading the riots: What were the police doing on twitter?, Policing and society 23 (4) (2013) 413–436.

[12] R. Procter, F. Vis, A. Voss, Reading the riots on twitter: methodological innovation for the analysis of big data, International journal of social research methodology 16 (3) (2013) 197–214.

[13] L. Derczynski, K. Bontcheva, M. Lukasik, T. Declerck, A. Scharl, G. Georgiev, P. Osenova, T. P. Lobo, A. Kolliakou, R. Stewart, et al., Pheme: Computing veracity – the fourth challenge of big social data, in: EU Project Networking Session at the European Semantic Web Conference, ESWC, 2015.

[14] A. Zubiaga, A. Aker, K. Bontcheva, M. Liakata, R. Procter, Detection and resolution of rumours in social media: A survey, ACM Computing Surveys.

[15] M. A. Walker, P. Anand, R. Abbott, R. Grant, Stance classification using dialogic properties of persuasion, in: Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT, Association for Computational Linguistics, 2012, pp. 592–596.

[16] V. Qazvinian, E. Rosengren, D. R. Radev, Q. Mei, Rumor has it: Identifying misinformation in microblogs, in: Proceedings of the Conference on Empirical Methods in Natural Language Processing, EMNLP, 2011, pp. 1589–1599.

[17] 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, CIKM, ACM, New York, NY, USA, 2015, pp. 1867–1870.

[18] M. Lukasik, P. K. 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 the Annual Meeting of the Association for Computational Linguistics, ACL, 2016, pp. 393–398.

[19] A. Zubiaga, E. Kochkina, M. Liakata, R. Procter, M. Lukasik, Stance classification in rumours as a sequential task exploiting the tree structure of social media conversations, in: Proceedings of International Conference on Computational Linguistics, COLING, 2016, pp. 2438–2448.

[20] A. Zubiaga, M. Liakata, R. Procter, Learning reporting dynamics during breaking news for rumour detection in social media, arXiv preprint arXiv:1610.07363.

[21] A. Zubiaga, M. Liakata, R. Procter, Exploiting context for rumour detection in social media, in: International Conference on Social Informatics, Springer, 2017, pp. 109–123.

[22] A. Zubiaga, M. Liakata, R. Procter, G. Wong Sak Hoi, P. Tolmie, Analysing how people orient to and spread rumours in social media by looking at conversational threads, PLoS ONE 11 (3) (2016) 1–29. doi:10.1371/journal.pone.0150989.

[23] K. S. Hasan, V. Ng, Extra-linguistic constraints on stance recognition in ideological debates., in: Proceedings of the Annual Meeting of the Association for Computational Linguistics, ACL, 2013, pp. 816–821.

[24] M. A. Walker, P. Anand, R. Abbott, J. E. F. Tree, C. Martell, J. King, That is your evidence?: Classifying stance in online political debate, Decision Support Systems 53 (4) (2012) 719–729.

[25] K. S. Hasan, V. Ng, Stance classification of ideological debates: Data, models, features, and constraints., in: Proceedings of the International Joint Conference on Natural Language Processing, IJCNLP, 2013, pp. 1348–1356.

[26] D. Sridhar, L. Getoor, M. Walker, Collective stance classification of posts in online debate forums, in: Proceedings of the Joint Workshop on Social Dynamics and Personal Attributes in Social Media, 2014, pp. 109–117.

[27] I. Augenstein, T. Rocktäschel, A. Vlachos, K. Bontcheva, Stance Detection with Bidirectional Conditional Encoding, in: Proceedings of the Conference on Empirical Methods for Natural Language Processing, EMNLP, 2016, pp. 876–885.

[28] S. M. Mohammad, S. Kiritchenko, P. Sobhani, X. Zhu, C. Cherry, Semeval-2016 task 6: Detecting stance in tweets, in: Proceedings of the International Workshop on Semantic Evaluation, SemEval, 2016, pp. 31–41.

[29] I. Augenstein, A. Vlachos, K. Bontcheva, USFD: Any-Target Stance Detection on Twitter with Autoencoders, in: Proceedings of the International Workshop on Semantic Evaluation, San Diego, California, 2016, pp. 389–393.

[30] M. Lukasik, T. Cohn, K. Bontcheva, Classifying tweet level judgements of rumours in social media, in: Proceedings of the Conference on Empirical Methods in Natural Language Processing, EMNLP, 2015, pp. 2590–2595.

[31] S. Hamidian, M. T. Diab, Rumor identification and belief investigation on twitter, in: Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT, 2016, pp. 3–8.

[32] M. Lukasik, K. Bontcheva, T. Cohn, A. Zubiaga, M. Liakata, R. Procter, Using Gaussian processes for rumour stance classification in social media, arXiv preprint arXiv:1609.01962.

[33] L. Zeng, K. Starbird, E. S. Spiro, #unconfirmed: Classifying rumor stance in crisis-related social media messages, in: Proceedings of the Tenth International AAAI Conference on Web and Social Media, ICWSM, 2016, pp. 747–750.

[34] H. Sacks, E. A. Schegloff, G. Jefferson, A simplest systematics for the organization of turn-taking for conversation, Language (1974) 696–735.

[35] P. Tolmie, R. Procter, M. Rouncefield, M. Liakata, A. Zubiaga, D. Randall, Supporting the use of user generated content in journalistic practice, in: Proceedings of the ACM Conference on Human Factors and Computing Systems, CHI, 2017, pp. 3632–3644.

[36] A. Ritter, C. Cherry, B. Dolan, Unsupervised modeling of twitter conversations, in: Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics, NAACL-HLT, Association for Computational Linguistics, 2010, pp. 172–180.

[37] 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, pp. 69–76.

[38] E. Kochkina, M. Liakata, I. Augenstein, Turing at SemEval-2017 Task 8: Sequential Approach to Rumour Stance Classification with Branch-LSTM, in: Proceedings of SemEval, ACL, 2017, pp. 475–480.

[39] F. Wang, M. Lan, Y. Wu, ECNU at SemEval-2017 Task 8: Rumour Evaluation Using Effective Features and Supervised Ensemble Models, in: Proceedings of SemEval, ACL, 2017, pp. 491–496.

[40] 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, ACL, 2017, pp. 497–501.

[41] M. García Lozano, H. Lilja, E. Tjörnhammar, M. Maja Karasalo, Mama Edha at SemEval-2017 Task 8: Stance Classification with CNN and Rules, in: Proceedings of SemEval, ACL, 2017, pp. 481–485.

[42] H. Bahuleyan, O. Vechtomova, UWaterloo at SemEval-2017 Task 8: Detecting Stance towards Rumours with Topic Independent Features, in: Proceedings of SemEval, ACL, 2017, pp. 461–464.

[43] 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.

[44] 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, pp. 465–469.

[45] 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, pp. 470–474.

[46] A. Zubiaga, M. Liakata, R. Procter, K. Bontcheva, P. Tolmie, Crowdsourcing the annotation of rumourous conversations in social media, in: Proceedings of the 24th International Conference on World Wide Web Companion, WWW, International World Wide Web Conferences Steering Committee, 2015, pp. 347–353.

[47] M. Lukasik, T. Cohn, K. Bontcheva, Point process modelling of rumour dynamics in social media, in: Proceedings of the Annual Meeting of the Association for Computational Linguistics, ACL, 2015, pp. 518–523.

[48] M. Lukasik, P. K. Srijith, T. Cohn, K. Bontcheva, Modeling tweet arrival times using log-gaussian cox processes, in: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP, 2015, pp. 250–255.

[49] M. Lukasik, T. Cohn, Convolution kernels for discriminative learning from streaming text, in: Proceedings of the Thirtieth AAAI Conference, 2016, pp. 2757–2763.

[50] J. Lafferty, A. McCallum, F. Pereira, Conditional random fields: Probabilistic models for segmenting and labeling sequence data, in: Proceedings of the Eighteenth International Conference on Machine Learning, ICML, Vol. 1, 2001, pp. 282–289.

[51] C. Sutton, A. McCallum, An introduction to conditional random fields, Machine Learning 4 (4) (2011) 267–373.

[52] A. C. Müller, S. Behnke, Pystruct: learning structured prediction in python, The Journal of Machine Learning Research 15 (1) (2014) 2055–2060.

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

[54] D. Kingma, J. Ba, Adam: A method for stochastic optimization, in: International Conference for Learning Representations, 2015, pp. 1–15.

[55] F. Bastien, P. Lamblin, R. Pascanu, J. Bergstra, I. Goodfellow, A. Bergeron, N. Bouchard, D. Warde-Farley, Y. Bengio, Theano: new features and speed improvements, in: Workshop on Deep Learning and Unsupervised Feature Learning, 2012, pp. 1–10.

[56] S. Dieleman, J. Schlüter, C. Raffel, E. Olson, S. K. Sønderby, D. Nouri, D. Maturana, M. Thoma, E. Battenberg, J. Kelly, J. D. Fauw, M. Heilman, D. M. de Almeida, B. McFee, H. Weideman, G. Takács, P. de Rivaz, J. Crall, G. Sanders, K. Rasul, C. Liu, G. French, J. Degrave, Lasagne: First release. (Aug. 2015). doi:http://doi.org/10.5281/zenodo.27878.

[57] J. S. Bergstra, R. Bardenet, Y. Bengio, B. Kégl, Algorithms for hyper-parameter optimization, in: Advances in Neural Information Processing Systems, 2011, pp. 2546–2554.

[58] J. Bergstra, D. Yamins, D. D. Cox, Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures, in: Proceedings of the International Conference on Machine Learning, ICML, Vol. 28, 2013, pp. 115–123.

[59] Q. McNemar, Note on the sampling error of the difference between correlated proportions or percentages, Psychometrika 12 (2) (1947) 153–157.

[60] P. Kampstra, Beanplot: A boxplot alternative for visual comparison of distributions, Journal of statistical software 28 (1) (2008) 1–9.

[61] E. Kouloumpis, T. Wilson, J. D. Moore, Twitter sentiment analysis: The good the bad and the omg!, in: Proceedings of the International Conference on Weblogs and Social Media, ICWSM, 2011, pp. 538–541.

[62] M. Tsytsarau, T. Palpanas, Survey on mining subjective data on the web, Data Mining and Knowledge Discovery 24 (3) (2012) 478–514.

[63] H. Saif, Y. He, M. Fernandez, H. Alani, Contextual semantics for sentiment analysis of twitter, Information Processing & Management 52 (1) (2016) 5–19.

[64] Z. Liu, B. J. Jansen, Identifying and predicting the desire to help in social question and answering, Information Processing & Management 53 (2016) 490–504.

[65] D. Vilares, M. A. Alonso, C. Gómez-Rodríguez, Supervised sentiment analysis in multilingual environments, Information Processing & Management 53 (3) (2017) 595–607.

[66] A. C. Pandey, D. S. Rajpoot, M. Saraswat, Twitter sentiment analysis using hybrid cuckoo search method, Information Processing & Management 53 (4) (2017) 764–779.

[67] B. Han, P. Cook, T. Baldwin, Text-based twitter user geolocation prediction, Journal of Artificial Intelligence Research 49 (2014) 451–500.

[68] A. Zubiaga, A. Voss, R. Procter, M. Liakata, B. Wang, A. Tsakalidis, Towards real-time, country-level location classification of worldwide tweets, IEEE Transactions on Knowledge and Data Engineering 29 (9) (2017) 2053–2066.

[69] S. Bergsma, P. McNamee, M. Bagdouri, C. Fink, T. Wilson, Language identification for creating language-specific twitter collections, in: Proceedings of the second workshop on language in social media, Association for Computational Linguistics, 2012, pp. 65–74.

[70] A. Zubiaga, I. San Vicente, P. Gamallo, J. R. Pichel, I. Alegria, N. Aranberri, A. Ezeiza, V. Fresno, Tweetlid: a benchmark for tweet language identification, Language Resources and Evaluation 50 (4) (2016) 729–766.

[71] P. Srijith, M. Hepple, K. Bontcheva, D. Preotiuc-Pietro, Sub-story detection in Twitter with hierarchical Dirichlet processes, Information Processing & Management 53 (2017) 989–1003.

[72] J. Reis, F. Benevenuto, P. O. de Melo, R. Prates, H. Kwak, J. An, Breaking the news: First impressions matter on online news, in: Proceedings of the International Conference on Weblogs and Social Media, ICWSM, 2015, pp. 357–366.

[73] J. An, M. Cha, P. Gummadi, J. Crowcroft, Media landscape in twitter: A world of new conventions and political diversity, in: Proceedings of the International Conference on Weblogs and Social Media, ICWSM, The AAAI Press, 2011, pp. 18–25.

[74] A. Pak, P. Paroubek, Twitter as a corpus for sentiment analysis and opinion mining, in: Proceedings of the Language Resources and Evaluation Conference, LREC, 2010, pp. 1320–1326.

[75] J. Bian, K. Yoshigoe, A. Hicks, J. Yuan, Z. He, M. Xie, Y. Guo, M. Prosperi, R. Salloum, F. Modave, Mining Twitter to assess the public perception of the "Internet of Things", PloS one 11 (7) (2016) e0158450.

[76] W. Housley, H. Webb, A. Edwards, R. Procter, M. Jirotka, Digitizing sacks? approaching social media as data, Qualitative Research.

[77] W. Housley, H. Webb, A. Edwards, R. Procter, M. Jirotka, Membership categorisation and antagonistic twitter formulations, Discourse & Communication.

[78] 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.

[79] K. Bontcheva, L. Derczynski, A. Funk, M. A. Greenwood, D. Maynard, N. Aswani, TwitIE: An open-source information extraction pipeline for microblog text, in: Proceedings of the International Conference on Recent Advances in Natural Language Processing, RANLP, Association for Computational Linguistics, 2013, pp. 83–90.

Appendix A. Features

Appendix A.1. Local Features

Local features are extracted from each of the tweets in isolation, and therefore it is not necessary to look at other features in a thread to generate them. We use four types of features to represent the tweets locally.

Local feature type #1: Lexicon.

- *Word Embeddings:* we use Word2Vec [78] to represent the textual content of each tweet. First, we trained a separate Word2Vec model for each of the eight folds, each having the seven events in the training set as input data, so that the event (and the vocabulary) in the test set is unknown. We use large datasets associated with the seven events in the training set, including all the tweets we collected for those events. Finally, we represent each tweet as a vector with 300 dimensions averaging vector representations of the words in the tweet using Word2Vec.

- *Part of speech (POS) tags:* we parse the tweets to extract the part-of-speech (POS) tags using Twitie [79]. Once the tweets are parsed, we represent each tweet with a vector that counts the number of occurrences of each type of POS tag. The final vector therefore has as many features as different types of POS tags we observe in the dataset.

- *Use of negation:* this is a feature determining the number of negation words found in a tweet. The existence of negation words in a tweet is determined by looking at the presence of the following words: not, no, nobody, nothing, none, never, neither, nor, nowhere, hardly, scarcely, barely, don't, isn't, wasn't, shouldn't, wouldn't, couldn't, doesn't.

- *Use of swear words:* this is a feature determining the number of 'bad' words present in a tweet. We use a list of 458 bad words[11].

Local feature type #2: Content formatting.

- *Tweet length:* the length of the tweet in number of characters.

[11] http://urbanoalvarez.es/blog/2008/04/04/bad-words-list/

- *Word count:* the number of words in the tweet, counted as the number of space-separated tokens.

Local feature type #3: Punctuation.

- *Use of question mark:* binary feature indicating the presence or not of at least one question mark in the tweet.
- *Use of exclamation mark:* binary feature indicating the presence or not of at least one exclamation mark in the tweet.

Local feature type #4: Tweet formatting.

- *Attachment of URL:* binary feature, capturing the presence or not of at least one URL in the tweet.

Appendix A.2. Contextual Features

Contextual feature type #1: Relational features.

- *Word2Vec similarity wrt source tweet:* we compute the cosine similarity between the word vector representation of the current tweet and the word vector representation of the source tweet. This feature intends to capture the semantic relationship between the current tweet and the source tweet and therefore help inferring the type of response.
- *Word2Vec similarity wrt preceding tweet:* likewise, we compute the similarity between the current tweet and the preceding tweet, the one that is directly responding to.
- *Word2Vec similarity wrt thread:* we compute another similarity score between the current tweet and the rest of the tweets in the thread excluding the tweets from the same author as that in the current tweet.

Contextual feature type #2: Structural features.

- *Is leaf:* binary feature indicating if the current tweet is a leaf, i.e. the last tweet in a branch of the tree, with no more replies following.

- *Is source tweet:* binary feature determining if the tweet is a source tweet or is instead replying to someone else. Note that this feature can also be extracted from the tweet itself, checking if the tweet content begins with a Twitter user handle or not.
- *Is source user:* binary feature indicating if the current tweet is posted by the same author as that in the source tweet.

Contextual feature type #3: Social features.

- *Has favourites:* feature indicating the number of times a tweet has been favourited.
- *Has retweets:* feature indicating the number of times a tweet has been retweeted.
- *Persistence:* this feature is the count of the total number of tweets posted in the thread by the author in the current tweet. High numbers of tweets in a thread indicate that the author participates more.
- *Time difference:* this is the time elapsed, in seconds, from when the source tweet was posted to the time the current tweet was posted.

Appendix A.3. Hawkes Features

- *Bag of words:* a vector where each token in the dataset represents a feature, where each feature is assigned a number pertaining its count of occurrences in the tweet.
- *Timestamp:* The UNIX time in which the tweet was posted.