DELIVERABLE SUBMISSION SHEET

To:	Susan Fraser	*(Project Officer)*
	EUROPEAN COMMISSION Directorate-General Information Society and Media EUFO 1165A L-2920 Luxembourg	

From:

Project acronym:	PHEME	Project number:	611233
Project manager:	Kalina Bontcheva		
Project coordinator	The University of Sheffield (USFD)		

The following deliverable:

Deliverable title:	Algorithms for Detecting Misinformation and Disinformation: Final Version
Deliverable number:	D4.3.2
Deliverable date:	30 November 2016
Partners responsible:	The University of Sheffield (USFD)
Status:	[✓ Public] ☐ Restricted ☐ Confidential

is now complete. [✓] It is available for your inspection.
[✓] Relevant descriptive documents are attached.

The deliverable is:

☐ a document
☐ a Website (URL: ....)
☐ software (...........................)
☐ an event
[✓] other (.....Prototype..........)

Sent to Project Officer: *Susan.Fraser@ec.europa.eu*	Sent to functional mail box: *CNECT-ICT*-611233*@ec.europa.eu*	On date: 02 December 2016

D4.3.2 Algorithms for Detecting Misinformation and Disinformation: Final Version

Ahmet Aker, Michal Lukasik (University of Sheffield),
Arkaitz Zubiaga (University of Warwick)
Kalina Bontcheva (University of Sheffield),
Trevor Cohn (University of Melbourne)

Abstract.
FP7-ICT Collaborative Project ICT-2013-611233 PHEME
Deliverable D4.3.2 (WP4)

This deliverable describes methods for detecting misinformation in social media streams. We document the development and evaluation of three separate components that form this misinformation detection process: rumour detection, rumour stance classification and rumour veracity classification.

Keyword list: rumour detection; rumour classification; veracity classification

Project	PHEME No. 611233
Delivery Date	December 2, 2016
Contractual Date	November 30, 2016
Nature	Prototype
Reviewed By	Ivelina Nikolova, Maria Liakata
Web links	http://www.pheme.eu
Dissemination	PU

PHEME Consortium

This document is part of the PHEME research project (No. 611233), partially funded by the FP7-ICT Programme.

University of Sheffield
Department of Computer Science
Regent Court, 211 Portobello St.
Sheffield S1 4DP
UK
Contact person: Kalina Bontcheva
E-mail: K.Bontcheva@dcs.shef.ac.uk

MODUL University Vienna GMBH
Am Kahlenberg 1
1190 Wien
Austria
Contact person: Arno Scharl
E-mail: scharl@modul.ac.at

ATOS Spain SA
Calle de Albarracin 25
28037 Madrid
Spain
Contact person: Tomás Pariente Lobo
E-mail: tomas.parientelobo@atos.net

iHub Ltd.
NGONG, Road Bishop Magua Building
4th floor
00200 Nairobi
Kenya
Contact person: Rob Baker
E-mail: robbaker@ushahidi.com

The University of Warwick
Kirby Corner Road
University House
CV4 8UW Coventry
United Kingdom
Contact person: Rob Procter
E-mail: Rob.Procter@warwick.ac.uk

Universitaet des Saarlandes
Campus
D-66041 Saarbrücken
Germany
Contact person: Thierry Declerck
E-mail: declerck@dfki.de

Ontotext AD
Polygraphia Office Center fl.4,
47A Tsarigradsko Shosse,
Sofia 1504, Bulgaria
Contact person: Georgi Georgiev
E-mail: georgiev@ontotext.com

King's College London
Strand
WC2R 2LS London
United Kingdom
Contact person: Robert Stewart
E-mail: robert.stewart@kcl.ac.uk

SwissInfo.ch
Giacomettistrasse 3
3000 Bern
Switzerland
Contact person: Peter Schibli
E-mail: Peter.Schibli@swissinfo.ch

Executive Summary

Social media is a rich source of news, with the caveat that it is also rife with rumours. Despite the challenge that rumours posit in mining social media for news, one can take advantage of the numerous reactions of the community of users to make sense of the rumours and try to determine their veracity.

This deliverable focuses on the challenge of detecting and classifying rumours in social media, as well as trying to determine their veracity value. The focus is on Twitter as the vehicle for viral rumour spreading for two reasons. Firstly, the availability of already annotated datasets makes research and experimentation significantly easier. Secondly, and more importantly, Twitter has become the de-facto standard platform for sharing this type of information, especially in crises or emergency situations. Even practitioners such as journalists are continually using this social platform to unwittingly initiate rumours, as happened in a recent case where a BBC reporter tweeted about the Queen being dead.

Work in this deliverable is organised in three rumour analysis tasks, which together form the rumour classification process: rumour detection, rumour stance classification and veracity classification. In the first component, we deal with the challenge of classifying tweet threads into rumourous versus non-rumourous, based on linguistic, social, and spread pattern information. Use of contextual characteristics, such as preceding tweets learnt throughout the event, show very promising results substantially boosting the classifier's performance by over 40%.

Next we report experiments on rumour stance classification. Individual tweets discussing a rumour are classified as supporting, denying, querying or commenting on the rumour. We do in two different settings: treating tweets as an isolate unit, and using conversational features of Twitter to leverage context. While the latter shows substantial improvements over the use of tweets in isolation, it also presents the caveat that it is not straightforward to collect complete conversations from Twitter's API in a streaming scenario, for which we suggest a hybrid approach.

In the third and final component, we discuss the development of a system for rumour veracity classification. Using features inherent in a tweet, and without needing to leverage additional context, we show that our system can achieve accuracy values around 90%.

The three components make use of language- and domain-independent features, which facilitates their application to similar rumour classification tasks in other languages and domains. Two of the component have, in fact, been integrated into the GATE open-source text analytypics platform (http://gate.ac.uk) as reusable open-source plugins.

Software and data used in this deliverable are publicly available from the PHEME

software download web page (https://www.pheme.eu/software-downloads/), in order to promote replicability and take-up.

Contents

Chapter 1

Introduction

From a business and government point of view there is an increasing need to interpret and act upon information from large-volume media, such as Twitter, Facebook, and newswire. However, knowledge gathered from online sources and social media comes with the major caveat that it cannot always be trusted. Below we show some of the many examples why this might be a serious problem. This serves as motivation for our work, which is on distinguishing misinformation from accurate reports in social media posts.

1.1 Motivation

Ratkiewicz et al. (2011) show a particular type of abuse called political astroturf. Political astroturf are campaigns targeted at misleading the public, that a particular piece of information is spreading spontaneously, when in fact it is coordinated by a single person or organisation. As the authors posit, multiple political elections have been influenced by such means, such as Barrack Obama's 2008 presidential campaign or Howard Dean's failed 2004 presidential candidacy. The possible reason for this, according to the authors, is that catchiness and repeatably are the drivers of information diffusion, rather than truthfulness.

Mendoza et al. (2010a) study the disaster event which is an earthquake in Chile in 2010. Authors point at the fact that false rumours were spreading at this time and claim that it contributed to the general chaos in the region, given the absence of first hand information from professionally curated and moderated sources. As an example, false rumours spread that a volcano had become active and there was a tsunami warning in Valparaiso. As in these cases, the distinction of verified news and false rumours is key in many situations.

A few very general examples of what may be achieved using disinformation are given by Karlova and Fisher (2012). *"People can use disinformation to harness influence over others (e.g., insinuating knowledge of personal information). Governments can use dis-*

information to exercise control over a populace. Businesses can use disinformation to maintain or repair their own reputation or to damage the reputation of a competitor". Being able to detect misinformation and disinformation, may present *"opportunities for meaningful engagement, public awareness and education, and commercial information service provision".*

The overall aim is to investigate the problem of veracity estimation of social media content. The information type to be analysed in the project at hand are rumours, formally defined in next chapter. We can distinguish two types of veracity. The first of them is subjective, that is what is the collective judgment of veracity from an individual's or a group's perspective. The second type is the ground truth, which refers to what is the actual truth value behind a story. Both of these phenomena are very useful. For example, in marketing the collective judgments are very useful, whereas in journalism the ground truth might be more useful. We aim to analyse both of them. Later, we refer to the connection between them.

Here we aim to determine the veracity of social media content in both its meanings, i.e. subjective and ground truth. Analysis may reveal interesting characteristics of information from social media, in particular we may be able to identify the features that reflect the different types of veracity. Prediction would be of great value to society, as false information could be filtered out.

Moreover, our aim is to create a system that would automatically assess veracity of a rumour thread as a whole (which consists of multiple sources and conversations) and the veracity of a single source. Our second, more specific aim, is to model rumour dynamics, by exploring temporal patterns.

1.2 Research questions

In more detail, this deliverable addresses the following research questions:

- **Rumour detection:** Can rumourous post threads/clusters be distinguished reliably from non-rumourous ones using a classifier?
- **Rumour stance classification:** For each post classified as rumourous, can we determine whether it confirms, denies, questions or simply makes a comment on the particular rumour?
- **Rumour veracity classification:** How accurately can rumour veracity be estimated, based on the post-level belief classification information from the previous step?

1.3 Relevance to PHEME

The PHEME project aims to detect and study the emergence and propagation of rumours in social media. The detection of mis- and disinformation, in particular, is an essential challenge which is being addressed here. This deliverable comprises results arising from research in Task T4.3 on methods for detecting misinformation.

1.3.1 Relevance to Project Objectives

Developing methods for the detection of rumours (or phemes) as they emerge in social media, is the goal of the PHEME project. The first step of that process is to separate rumorous from non-rumorous conversations, followed by post-level belief classification and veracity estimation. The ability to model rumour dynamics is also a key project objective.

1.3.2 Relation to Other Workpackages

WP4 builds on the linguistic tools arising from WP2, which are used to derive features for the machine learning models described in this deliverable. It also builds on work in Task 3.3. on identifying stories and conversations through clustering. This forms the input to the rumour vs non-rumour detection experiments reported here, and the subsequent post-level and veracity classification work.

The data used in experiments here has been gathered and annotated as part of WP8 and WP2 at present, with subsequent experiments planned on WP7 data.

The results from the methods presented here will be visualised in the PHEME dashboard (WP5), while the methods themselves are being integrated within the PHEME integrated platform for near real-time processing.

1.4 Outline of the Deliverable

The rest of this deliverable is organised as follows. Next, in chapter 2 we describe the development, testing and evaluation of the three different components that form our misinformation detection system. These three components are organised into separate sections, describing first our rumour detection component in Section 2.1, then our stance classification components, for isolate tweets in Section 2.2 and for conversations in Section 2.3, and finally the veracity classification component in Section 2.4. The deliverable follows with a discussion of our achievements and limitation of the components in Chapter 3.

Chapter 2

Detection of mis- and dis-information

Tweets are used to spread rumours which can be true or false. In terms of false the information conveyed is either a type of misinformation or disinformation. These information types have been defined in the earlier deliverable D4-3-1 as such that misinformation is information that is inaccurate but where the author makes an honest mistake, and disinformation is information that is stated deliberately wrong.

It is desirable to have an automated process that automatically flags rumours, i.e. determines that the content a tweet constitutes a rumour. Classifying tweets as rumours is also referred to as rumour retrieval or *rumour detection*. Once a tweet is flagged as a rumour the next step is to determine the stance of different tweets towards the same rumour, with the aim of aggregating the stances of different authors. This task is referred to as belief classification or *rumour stance classification*. Finally, it is desired to verify whether a rumour is true or false which is referred to as credibility or *veracity classification*. In this chapter we define the whole pipeline that includes rumour detection (Section 2.1), stance classification (for isolate tweets in Section 2.2 and for conversational threads in 2.3) and veracity classification (Section 2.4).

2.1 Rumour detection

The rumour detection task as that in which, given a timeline of tweets, the system has to determine which of the tweets are reporting rumours, and hence are spreading information that is yet to be verified. The identification of rumours is the initial step whose output feeds the subsequent stage of the pipeline, the stance classification task. Formally, the task takes an evolving timeline of tweets $TL = \{t_1, ..., t_{|TL|}\}$ as input, and the classifier has to determine whether each of these tweets, t_i, is a rumour or a non-rumour by assigning a label from $Y = \{R, NR\}$.

Hence, we formulate the task as a binary classification problem, whose performance is evaluated by computing the precision, recall and F1 scores for the target category, i.e., rumours.

In what follows in Section 2.1.1 we summarise previous work on rumour detection. Next, we describe the rumour detection approaches we study in Section 2.1.2. In Section 2.1.3 we present and analyse the results.

2.1.1 Related Work

To the best of our knowledge, the only published work that has tackled the detection of new rumours is that by Zhao et al. (2015). Their approach builds on the assumption that rumours will provoke tweets from skeptical users who question or enquire about their veracity; the fact that a piece of information has a number of querying tweets associated with it would then imply that the information is rumourous. The authors created a manually curated list of five regular expressions (e.g., "is (that | this | it) true"), which are used to identify querying tweets. These querying tweets are then clustered by similarity, each cluster being ultimately deemed a candidate rumour. It was not viable for the authors to evaluate by recall, but their best approach achieved 52% and 28% precision for two datasets. While this work builds on a sensible hypothesis and presents a clever approach to tackling the rumour detection task, we foresee three potential limitations: (1) being based on manually curated regular expressions the approach may not generalise well, (2) the hypothesis might not always apply and hence lead to low recall as, for example, certain rumours reported by reputable media are not always questioned by the general public Zubiaga et al. (2016), and (3) it takes no account of the context that precedes the rumour, which can give additional insights into what is going on and how a piece of information can be rumourous in that context (e.g., the rumour that *a gunman is on the loose*, when the police has not confirmed it yet, is easier to be deemed a rumour if we put it into the context of the preceding events, such as additional reports that the identity of the gunman is unknown and the reasons that motivated the shooting have not been found out). As a more flexible rumour detection approach in the context of breaking news, we introduce a context-aware rumour detection system that uses a sequential classifier to examine the reporting dynamics during breaking news to determine if a new piece of information constitutes a rumour.

Other work in rumour detection Qazvinian et al. (2011); Hamidian and Diab (2015b, 2016) has been limited to finding rumours known *a priori*. A classifier is fed with a set of predefined rumours (e.g., *Obama is muslim*), which then classifies new tweets as being related to one of the known rumours or not (e.g., *I think Obama is not muslim* would be about the rumour, while *Obama was talking to a group of Muslims* wouldn't). An approach like this can be useful for long-standing rumours, where one wants to identify relevant tweets to track the rumours that have already been identified; one may also refer to this task as *rumour tracking* rather than *rumour detection*. Our goal here is instead to detect new rumours.

2.1.2 Method

We use Conditional Random Fields (CRF) as a sequential classifier that enables aggregation of tweets as a chain of reports. We use a Maximum Entropy classifier as the non-sequential equivalent of CRF to test the validity of the hypothesis, and also use additional baseline classifiers for further comparison. Moreover, we also reproduce a baseline based on the approach introduced by Zhao et al. (2015) to compare the performance of our approach with that of a state-of-the-art approach.

Conditional Random Fields (CRF). We use CRF as a structured classifier to model sequences of tweets as observed in the timelines of Twitter breaking news. With CRF, we can model the timeline as a linear chain or graph that will be treated as a sequence of rumours and non-rumours. In contrast to classifiers traditionally used 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, i.e., a link between a tweet and its preceding tweet in the event. Hence, having a data sequence X as input, CRF outputs a sequence of labels Y Lafferty et al. (2001), 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 2.1 Sutton and McCallum (2011)[1].

$$p(y|x) = \frac{1}{Z(x)} \prod_{a=1}^{A} \Psi_a(y_a, x_a) \tag{2.1}$$

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

Therefore, in our specific case of rumour detection, CRF will exploit the sequence of rumours and non-rumours leading up to the current tweet to determine if it is a rumour or not. It is important to note that with CRF the sequence of rumours and non-rumours preceding the tweet being classified will be based on the predictions of the classifier itself, and will not use any ground truth annotations. Errors in early tweets in the sequence may then augment errors in subsequent tweets. For each tweet to be classified, we solely feed the preceding tweets to the classifier to simulate a realistic scenario where subsequent tweets are not yet posted and early decisions need to be made on each tweet.

Maximum Entropy classifier (MaxEnt). As the non-sequential equivalent 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 sequence and hence the preceding tweets. This enables us to compare directly the extent to which treating the tweets posted during an event as a sequence instead of having each tweet as a separate unit can boost the performance of the classifier.

[1]We use the PyStruct to implement Conditional Random Fields Müller and Behnke (2014).

Enquiry-based approach by Zhao et al. (2015): As a state-of-the-art baseline for rumour detection, and the only approach that so far has tackled rumour detection in social media, we reproduce the approach by Zhao et al., which uses regular expressions to look for enquiry posts. We use the set of replies responding to each tweet to look for enquiry posts. Following the approach described by the authors, we consider that a tweet is a rumour if at least one of the replying tweets matches with one of the regular expressions that the authors curated. The list of regular expressions defined by the authors is shown in Table 2.1.

Pattern Regular Expression	Type
is (that | this | it) true	Verification
wh[a]*t[?!][?1]*	Verification
(real? | really ? | unconfirmed)	Verification
(rumour | debunk)	Correction
(that | this | it) is not true	Correction

Table 2.1: List of regular expressions utilised by Zhao et al., which we reimplemented to reproduce their approach as a baseline. Regular expressions for both enquires and corrections are combined, and a tweet that matches any of them will be deemed an enquiry tweet.

Additional baselines. We also compare three more non-sequential classifiers[2]: Naive Bayes (NB), Support Vector Machines (SVM), and Random Forests (RF).

We perform the experiments in a 5-fold cross-validation setting, having in each case four of the events for training, and the remainder event for testing. This enables us to simulate a realistic scenario where an event is completely unknown to the classifier and it has to identify rumours from the knowledge garnered from events in the training set. For evaluation purposes, we aggregate the output of all five runs as the micro-averaged evaluation across runs.

We use two types of features with the classifiers: content-based features and social features. We test them separately as well as combined. The features that fall in each of these categories are as follows:

Content-based Features. We use seven different features extracted from the content of the tweets:

- **Word Vectors:** to create vectors representing the words in each tweet, we build word vector representations using Word2Vec Mikolov et al. (2013). We train a different Word2Vec model with 300 dimensions for each of the five folds, training the model in each case from the collection of tweets pertaining to the four events in the training set, so that the event (and the vocabulary) in the test set is unknown. As a result, we get five different Word2Vec models, each used in a separate fold.

[2]We use their implementation in the scikit-learn Python package for Maximum Entropy, Naive Bayes, Support Vector Machines and Random Forests.

- **Part-of-speech Tags:** we build a vector of part-of-speech (POS) tags with each feature in the vector representing the number of occurrences of a certain POS tag in the tweet. We use Twitie Bontcheva et al. (2013) to parse the tweets for POS tags, an information extraction package that is part of GATE Cunningham et al. (2011).
- **Capital Ratio:** the ratio of capital letters among all alphabetic characters in the tweet. Use of capitalisation tends to represent emphasis, among others.
- **Word Count:** the number of words in the tweet, counted as the number of space-separated tokens.
- **Use of Question Mark:** a binary feature representing if the tweet has at least a question mark in it. Question marks may be indicative of uncertainty.
- **Use of Exclamation Mark:** a binary feature representing if the tweet has at least an exclamation mark in it. Exclamation marks may be indicative of emphasis or surprise.
- **Use of Period:** a binary feature representing if the tweet has at least a period in it. Punctuation may be indicative of good writing and hence potentially of slow reporting.

Social Features. We use five social features, all of which can be inferred from the metadata associated with the author of the tweet, and which is embedded as part of a tweet object retrieved from the Twitter API. We define a set of social features that are indicative of a user's experience and reputation:

- **Tweet Count:** we infer this feature from the number of tweets that a user has posted on Twitter. As numbers can vary substantially across users, we normalise them by rounding up the 10-base logarithm of the tweet count: $\lceil \log_{10}(\text{statusescount}) \rceil$.
- **Listed Count:** this feature is computed by normalising the number of lists a user belongs to, i.e., the number of times other users decided to add them to a list: $\lceil \log_{10}(\text{listedcount}) \rceil$.
- **Follow Ratio:** in this feature we look at the reputation of a user as reflected by their number of followers. However, the number of followers might occasionally be rigged, e.g., by users who simply follow many others to attract more followers. To control for this effect, we define the follow ratio as the logarithmically scaled ratio of followers over followees: $\lfloor \log_{10}\ (\#\text{followers}/\#\text{following}) \rceil$.
- **Age:** we compute the age of a user as the rounded number of years that the user has spent on Twitter, i.e., from the day the account was set up to the day of the current tweet.
- **Verified:** a binary feature representing if the user has been verified by Twitter or not. Verified users are those whose identity Twitter has validated, and tend to be reputable people.

2.1.3 Results

Table 2.2 shows the results for different classifiers using either or both of the content-based and social features, as well as the results for the state-of-the-art classifier by Zhao et al. (2015). Performance results of the classifiers using content-based features suggests a remarkable improvement for CRF over the rest of the classifiers, implying that CRF benefits from the use of the sequence of tweets preceding each tweet as context to enrich the input to the classifier. This is especially true when we look at precision, where CRF performs substantially better than the rest. Only the Naive Bayes classifier performs better in terms of recall, however, it performs poorly in terms of precision. As a result, CRF balances precision and recall in a clearly better way, outperforming all the other classifiers in terms of the F1 score.

Results are not as clear when we look at those using social features. CRF still performs best in terms of precision, but performance drops if we look at the recall. In fact, most of the classifiers perform better than CRF in terms of recall, with SVM as the best performing classifier. Combining both precision and recall in an F1 score shows that SVM is the classifier that best exploits social features. However, performance results using social features are significantly worse than those using content-based features, which suggests that social features alone are not sufficient.

When both content-based features and social features are combined as an input to the classifier, we see that the results resemble that of the use of content-based features alone. CRF outperforms all the rest in terms of precision, while Naive Bayes is good only in terms of recall. As a result, the aggregation of features also leads to CRF being the best classifier in terms of F1 score. In fact, CRF leads to an improvement of 39.9% over the second best classifier in terms of F1, Naive Bayes. If we compare the results of CRF with the use of content-based features alone or combining both types of features, we notice that the improvement comes especially for recall, which is balanced out with a slight drop of precision. As a result, we get an F1 score that is slightly better when using both features together. In fact, all F1 scores for combined features are superior to their counterparts using content-based features alone, among which CRF performs best.

Comparison with respect to the enquiry-based baseline approach introduced by Zhao et al. buttresses our conjecture that a manually curated list of regular expressions may lead to low recall, which is as low as 0.065 in this case. This approach gets a relatively good precision score, which beats all of our baselines, although it performs substantially worse than CRF. However, 59% of false positives as can be inferred from the precision of 0.41 indicates that the regular expressions also match non-rumours. One could also opt to expand the list of regular expressions and/or adapt them to our specific scenario and events; however, this may involve substantial manual work and would not guarantee generalisable performance.

Content			
Classifier	P	R	F1
SVM	0.355	0.445	0.395
Random Forest	0.271	0.087	0.131
Naive Bayes	0.309	**0.723**	0.433
Maximum Entropy	0.329	0.425	0.371
CRF	**0.683**	0.545	**0.606**
Social			
Classifier	P	R	F1
SVM	0.337	**0.524**	**0.410**
Random Forest	0.343	0.433	0.382
Naive Bayes	0.294	0.010	0.020
Maximum Entropy	0.336	0.476	0.394
CRF	**0.462**	0.268	0.339
Content + Social			
Classifier	P	R	F1
SVM	0.337	0.483	0.397
Random Forest	0.275	0.099	0.145
Naive Bayes	0.310	**0.723**	0.434
Maximum Entropy	0.338	0.442	0.383
CRF	**0.667**	0.556	**0.607**
State-of-the-art Baseline			
Classifier	P	R	F1
Zhao et al. (2015)	0.410	0.065	0.113

Table 2.2: Results for rumour detection.

2.2 Rumour stance classification

Stance classification is well studied in online debates where the aim is to classify the user entries by “for” and “against”. Studies in these respect define stance as an overall position held by a person towards an object, idea or position Somasundaran and Wiebe (2009); Walker et al. (2012). Unlike stance classification in online debates the aim of rumour stance classification is to classify the user contributions by “supporting”, “questioning”, “denying” and “commenting”.

In our earlier work described in Deliverable D6.2.1 Derczynski et al. (2016) we investigated techniques to classify rumours as “supporting”, “questioning” or “denying”. In this section we expand our earlier work by considering the forth category namely “commenting”. We also enhance our feature set with additional features in order to boost the classification accuracy. Furthermore, we introduce several classifiers such as Support Vector Machines (SVM), J48 Tree, Bayes, Random Forest and Instance Based Learning to the task and show that they can lead to better classification accuracy compared to Gaussian Process classification used in D6.2.1.

In the following we briefly review related work on rumour stance classification. In Section 2.2.2 we outline our classification approach and present and discuss the classification experiments in Section 2.2.3. We packaged the developed approach as a GATE plugin. Section 2.2.4 gives details about the plugin.

2.2.1 Related work

Mendoza et al. (2010b) manually looked at rumours with established veracity levels to understand the stance of Twitter users take with respect to true and false rumours, i.e. with rumours which were proven true or false. They looked at seven rumours which were later proven true and seven rumours that were false. They manually labeled the tweets with the stance categories “affirms” (supports), “denies” and “questions”. They found out that over 95% of true tweets have been classified with the “affirms”, around 4% with “questions” and only 0.4% with the “denies” category. On the other hand 38% of the false tweets were put under the “denies” category and 17% under the “questions” category. Both of these figures show that false tweets are either denied or questioned by others. However, nevertheless there are still 45% of tweets that were affirmed by other tweets.

The study of Mendoza et al. (2010b) is manual and did not involve any development of tools for automated classification. The first study that tackles the stance classification automatically is reported by Qazvinian et al. (2011). In addition to stance classification the authors also perform automatic detection of relevance of tweets to rumours. In both tasks they use manually generated Twitter data set containing 10K tweets to guide a supervised machine learning approach. The authors use different features categorized by “content”, “network” and “Twitter specific memes”. The content category contains unigrams, bi-grams and their POS tags as features. In the network category the authors look at the retweet (RT) as a feature. Finally, the Twitter specific memes entail hashtag and URLs' content features. As machine learning approach Bayesian Classifiers are used. In the rumour detection tasks a mean average precision of 96.5% is reported. For this task the authors also note that the best performing features are the content based ones. In the rumour stance classification task, the tweets are classified as supporting, denying, questioning or neutral. In terms of results similar observations are reported. When all features are used an accuracy of 93.5% and 94.4% precision and 90.6% recall are achieved. Again the best performing features are those in the content category.

Similar to Qazvinian et al. (2011), Hamidian and Diab (2015a) perform rumour detection and rumour type classification by applying supervised machine learning using the same data set. However, instead of Bayesian classifiers the authors use J48 decision tree implemented within the Weka platform Hall et al. (2009). The features from Qazvinian et al. (2011) are adopted and extended with time related information and hashtags. In addition to the feature categories introduced above Hamidian and Diab introduce another feature category namely "pragmatic". The pragmatic features include named entity, event, sentiment and emoticons. The aim with the pragmatic features is to detect the stance within the tweet. The evaluation of the performance is casted as either 1-step problem containing a 6 class classification task (not rumour, 4 classes of stance and not determined by the annotator) or 2-step problem containing first a 3 class classification task (not rumour or rumour, not determined) and then 4 class classification task (stance classification). Better performances are achieved using the 2-step approach leading to 82.9% F-1 measure compared to 74% with the 1-step approach. The authors also report that the best performing features were the content based features and the worst performing ones the network and Twitter specific features. In their recent paper, Hamidian and Diab (2016) introduce the Tweet Latent Vector (TLV) approach that is obtained by applying the Semantic Textual Similarity model proposed by Guo and Diab (2012). The authors compare the TLV approach to their own earlier system as well as to original features of Qazvinian et al. (2011) and show that the TLV approach outperforms both baselines.

Liu et al. (2015) follow the resulting investigations about stances in rumours made by Mendoza et al. (2010b) and use stance as additional feature to those reported by related work to tackle the veracity classification problem. On the stance classification the authors adopt the approach of Qazvinian et al. (2011) and compare it with a rule-based method briefly outlined by the authors. They claim that their rule-based approach performed better than the one adopted from related work and thus use the rule-based stance classification as additional component on the veracity problem (see Section 2.4.1). The experiments were performed on the data set reported by Qazvinian et al. (2011). Unfortunately the authors do not provide detailed analysis about the performance of the stance classification.

More recently, Zeng et al. (2016) enriches the feature sets investigated by earlier studies by features determined through the Linguistic Inquiry and Word Count (LIWC) dictionaries Tausczik and Pennebaker (2010). They investigate supervised approach using Logistic Regression, naïve Bayes and Random Forest classification. The authors use their own manually annotated data to classify them by stance. However, unlike previous studies Zeng et al. consider only two classes: affirm and deny. Best results are reported with Random Forest leading to 87% precision, 96.9% recall, 91.7% F1-measure and 88.4% accuracy.

2.2.2 Method

Our rumour stance classification is performed using Support Vector Machines (SVMs) with the RBF kernel Buhmann (2003), the J48 Tree, Random Forest and naïve Bayes and Instance Based classifier. The SVM parameter settings we used in our experiments are: *-S 0 -K 2 -D 3 -G 0.1 -R 0.0 -N 0.5 -M 40.0 -C 1.5 -E 0.001 -P 0.1*. We applied Weka's integrated grid search to learn to optimal values for Gamma (G) and regularization parameter (C). The J48 Tree is run with *-U* (unpruned tree) flag set. For the Random Forest we use 100 trees (*-I 100*). Finally we run the naïve Bayes as well as the Instance Based classifier with default Weka settings.

The classifiers rely on the following features extracted from each tweet:

- **BOW:** Bag of words have been used in D6.2.1 and we continue using them in the current setting. In short for this feature we first create a dictionary from all tweets in the corpus. Next each tweet is assigned the words in the dictionary as features. For words occurring in the tweet the feature values are set to the number of times they occur in the tweet. For all other words "0" is used.

- **Brown Cluster:** Similar to the BOW feature the brown clustering is further used from our previous approach. Brown clusters are obtained from a bigger tweet corpus that entails assignments of words to brown cluster ids. We used 1000 clusters, i.e. there are 1000 cluster ids. All 1000 ids are used as features however, only ids that cover words in the tweet are assigned a feature value "1". All other cluster id feature values are set to "0".

- **POS tag:** The BOW feature captures the actual words and is domain dependent. To create a feature that is not domain dependent we added Part of Speech (POS) tags as additional feature. Similar to the BOW feature we created a dictionary of POS tags from the entire corpus (excluding the medical data) and used this dictionary to label each tweet with it – binary, i.e. whether a POS tag is present.[3] However, instead of using just single POS tag we created sequences containing bi-gram, tri-gram and 4-gram POS tags. Feature values are the frequencies of POS tag sequences occurring in the tweet.

[3]We also experimented with frequencies of POS tags, i.e. counting how many times a particular POS tag occurs in the tweet. The counts then have been normalized using mean and standard deviation. However, the frequency based POS feature negatively affected the classification accuracy so that we omitted it from the feature set.

- **Sentiment:** This is another domain independent feature. Sentiment analysis reveals the sentimental polarity of the tweet such as whether it is positive or negative. We used the Stanford sentimentSocher et al. (2013) tool to create this feature. The tool returns a range from 0 to 4 with 0 indicating "very negative" and 4 "very positive". First, we used this as a categorical feature but turning it to a numeric feature gave us better performance. Thus each tweet is assigned a sentiment feature whose value varies from 0 to 4.

- **NE:** Named entity (NE) is also domain independent. We check each tweet whether it contains *Person, Organization, Date, Location* and *Money* tags and for each tag in case of presence we add "1" otherwise "0".

- **Reply:** This feature is a binary feature and assigns "1" if the tweet is a reply to a previous one or not. The reply information is extracted from the tweet metadata. Again this feature is domain independent.

- **Emoticon:** We created a dictionary of emoticons using Wikipedia[4]. In Wikipedia those emoticons are grouped by categories. We use the categories as the feature. If any emoticon from a category occurs in the tweet we assign for that category feature the value "1" otherwise "0". Again similar to the previous features this feature is domain independent.

- **URL:** This is again domain independent. We assign the tweet "1" if it contains any URL otherwise "0".

- **Mood:** Mood detection analyses a textual content using different view points or angles. We use the tool described by Celli et al. (2016) to perform the mood detection. This tools looks from 5 different angles to each tweet: amused, disappointed, indignant, satisfied and worried. For each of this angles it returns a value from -1 to +1. We use the different angles as the mood features and the returned values as the feature value.

- **Originality score**: Is the count of tweets the user has produced, i.e. the "statuses count" in the Twitter API.

- **isUserVerified(0-1)**: Whether the user is verified or not.

- **NumberOfFollowers**: Number of followers the user have.

- **Role score**: Is the ratio between the number of followers and followees (i.e. NumberOfFollowers/NumberOfFollowees).

- **Engagement score**: Is the number of tweets divided by the number of days the user has been active (number of days since the user account creation till today).

[4]https://en.wikipedia.org/wiki/List_of_emoticons

- **Favourites score**: The "favourites count" divided by the number of days the user has been active.
- **HasGeoEnabled(0-1)**: Whether the user has enabled geo-location or not.
- **HasDescription(0-1)**: Whether the user has description or not.
- **LenghtOfDescription in words**: The number of words in the user description.
- **averageNegation**: We determine using the Stanford parser Chen and Manning (2014) the dependency parse tree of the tweet, count the number of negation relation ("neg") that appears between two terms and divide this by the number of total relations.
- **hasNegation(0-1)**: Whether the tweet has negation relationship or not.
- **hasSlangOrCurseWord(0-1)**: A dictionary of key words[5] are used to determine the presence of slang or curse words in the tweet.
- **hasGoogleBadWord(0-1)**: Same above but the dictionary of slang words is obtained from Google.[6]
- **hasAcronyms(0-1)**: The tweet is checked for presence of acronyms using a acronym dictionary.[7]
- **averageWordLength**: Average length of words (sum of word character counts divided by number of words in each tweet).
- **surpriseScore**: We collected a list of surprise words such as "amazed", "surprised", etc. We use this list to compute a cumulative vector using word2Vec Mikolov et al. (2013) – for each word in the list we obtain its word2Vec representation, add them together and finally divide the resulting vector by the number of words to obtain the cumulative vector. Similarly a cumulative vector is computed for the words in the tweet – excluding acronyms, named entities and URLs. We use cosine to compute the angle between those two cumulative vectors to determine the surprise score. Our word embeddings comprise the vectors published by Baroni et al. Baroni et al. (2014).
- **doubtScore**: Similar to the *surpriseScore*. The difference is that we use a list of doubt words such as "doubt", "uncertain", etc.
- **noDoubtScore**: In this feature we proceed as in *doubtScore* but use instead words which stand for certainty such as "surely", "sure", "certain", etc.
- **hasQuestionMark(0-1)**: Whether the tweet has "?" or not.

[5]www.noswearing.com/dictionary
[6]http://fffff.at/googles-official-list-of-bad-words
[7]www.netlingo.com/category/acronyms.php

Dataset	Rumours	Supporting	Denying	Questioning	Commenting
Ottawa shooting	58	161	76	64	481
Ferguson riots	46	192	83	94	685
Germanwings crash	68	177	12	28	169
Charlie Hebdo	74	236	56	51	710
Sydney siege	71	89	4	99	713

Table 2.3: Counts of tweets with supporting, denying or questioning labels in each event collection from the PHEME rumour data set.

- **hasExclamationMark(0-1)**: Whether the tweet has “!” or not.
- **hasDotDotDot(0-1)**: Whether the tweet is has “...” or not.
- **numberOfQuestionMark**: Number of time “?” occurs in the tweet.
- **NumberOfExclamationMark**: Number of times “!” occurs in the tweet.
- **numberOfDotDotDot**: Number of time “...” occurs in the tweet.
- **Binary regular expressions applied on each tweet**: .*(rumor?—debunk?).*, .*is (that—this—it) true.*, etc. In total there are 10 features covering regular expressions.

Each tweet feature described above is extracted along with its class label. These are used by the classifiers either for learning purposes when they are run in training mode or for prediction if they run in testing mode. The training and testing steps are outlined in the next Section.

2.2.3 Experimental setup and results

In our experiments we continue working with the same data set as in D6.2.1. A short summary of the dataset is given in Table 2.3. In summary the data consist of Tweets from 5 different events: Ottawa shooting, Ferguson riots, Germanwings crash, Charlie Hebdo and finally Sydney siege. Each data set has a different number of rumours where each rumour contains tweets marked with annotations such as “supporting”, “questioning”, “denying” and “commenting”.

Following a similar strategy for training-testing to that we described in Lukasik et al. (2015), we cast the rumour stance classification task into its specific categories as a supervised problem. In Lukasik et al. (2015) the idea was to follow two scenarios for training and testing: (1) training and testing are performed on out-domain data and (2) introducing small proportion of in-domain data in the training phase. In (1) the classifier is trained on all rumours except the one that is used for testing. In (2) the training data is enriched with first 10, 20, 30, 40 and 50 tweets from the rumour set that builds the testing data. In this deliverable we perform (1) as described above however, make a slight change on

classifier	**in domain tweets (in %)**	**Ottawa shooting**	**Ferguson riots**	**Charlie Hebdo**	**German wings**	**Sydney siege**	**macro mean (Acc)**
SVM	0	64.72	65.89	66.52	64.46	61.36	64.59
Random Forest	0	68.76	66.71	75.03	71.52	68.31	70.07
IBk	0	68.55	66.12	77.1	76.5	70.82	71.82
Bayes	0	71.81	70.42	76.7	73.38	73.42	73.14
J48	0	72.4	71.26	80.41	80.57	74.56	**75.84**

Table 2.4: Different classifier performances on setting (1). IBk is the Instance Based Learning classifier. Results are for the stance classification.

in domain tweets (in %)	**Ottawa shooting**	**ferguson riots**	**Charlie Hebdo**	**Germanwings**	**Sydney siege**	**macro mean (Acc)**
0	72.4	71.26	80.41	80.57	74.56	75.84
10	73.53	72.16	80.14	79.25	73.25	75.66
20	74.6	73.29	80.47	80.29	74.57	76.64
30	76.65	72.99	79.96	80.37	74.35	76.86
40	76.31	73.61	80.63	81.24	75.69	77.5
50	80.95	74.67	81.5	81.03	76.36	78.9
60	81.95	78.36	82.04	81.53	76.64	80.1

Table 2.5: Classification results in accuracy obtained using the J48 Tree. Results are for the stance classification.

the (2) set-up. Instead of specific counts we use percentage, i.e. 10% 20%, etc. We think percentage is a better option because some rumours might have smaller number of tweets than what would be required to transfer from the testing data to the training one. We use setting (1) to determine the best performing classifier. The results are shown in Table 2.4.

From the results in Table 2.4 we can see that the least performing classifier is the SVM one and the best the J48. We think SVM does not perform well because our training data is imbalanced in terms of class instances. As shown in Table 2.3 there are far more commenting instances than the other 3 classes. The J48 Tree is not very much affected by this as it can handle imbalanced data. In the following we use J48 to report detailed results in both evaluation settings (1) and (2). The results for the best performing classifier – J48 Tree – are shown in Table 2.5.

The results shown in Table 2.5 are obtained with the combination of features described above and using J48 Tree as the classifier. The results are reported in classification accuracy and are macro averaged over the different events. The row with 0% represents the set-up scenario (1) discussed above. From the results we can see, as noted in our earlier studies Lukasik et al. (2015) and report (D6-2-1), inclusion of in-domain data in the train-

Method	**in domain tweets**	**Ottawa shooting**	**ferguson riots**	**Charlie Hebdo**	**German w.**	**Sydney siege**	**macro mean (Acc)**
GPICM	10	62.95	59.56	76.79	74.14	65.37	67.76
GPICM	20	63.75	55.49	76.45	74.14	61.77	66.32
GPICM	30	62.95	57.68	76.79	74.14	62.6	66.83
GPICM	40	68.92	59.87	78.5	70.69	63.16	68.23
GPICM	50	69.72	58.31	77.13	74.14	63.99	**68.66**

Table 2.6: Accuracy for each of the PHEME datasets using Gaussian Process as the stance classification method. Note that the count of in-domain tweets are raw numbers, not percentages.

in domain tweets (in %)	**Ottawa shooting**	**Ferguson riots**	**Charlie Hebdo**	**Germanwings**	**Sydney siege**	**macro mean (F1)**
0	69.38	67.95	79.11	77.81	72.63	73.37
10	68.1	66.98	77.72	74.42	69.87	71.42
20	69.55	67.79	78.27	76.94	70.8	72.67
30	72.06	67.84	77.44	78.28	70.61	73.25
40	71.77	68.66	77.88	78.08	72.34	73.74
50	77.97	70.93	78.28	77.4	73.63	75.64
60	79.17	73.32	78.43	79.73	73.66	76.86

Table 2.7: J48 Tree stance classification results in weighted F1 over the 4 classes.

ing improves the results significantly. Furthermore, we can see the more in-domain data is included the better is the overall performance (except from 0% to 10%). We also would like to highlight that overall the results obtained through these features as well as the application of J48 lead to better performance than compared to our previous setting reported in D6-2-1. In Table 2.20 of D6-2-1 we reported 68.66% accuracy with 50 tweets included from the testing set to the training pool and using Gaussian process as the classifier. The full results of our previous experiments are shown in Table 2.6. In some cases accuracy can be biased if there is an unbalanced number of class instances. Because of this reason we also computed F1 – the harmonic mean between precision and recall. The results for this are shown in Table 2.7.

2.2.4 GATE plugin

We packaged the best performing classifier into GATE in the form of a GATE-plugin. The plugin requires some pre-processing such as loading a tweet to GATE, tokenisation, POS-tagging, named entity recognition, sentiment detection, etc. It also requires the GATE learning framework to perform the classification. We have created an application pipeline with all the required steps and resources that can be loaded into GATE.

[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 2.1: 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.

2.3 Conversational rumour stance classification

The previous section tackles the rumour stance classification task by looking at individual tweets. As a complementary objective, we also looked at the ability to classify tweets that are part of a conversation. In a conversation initiated by a rumour, where users respond to one another discussing the veracity of the rumour, one can also leveraged the conversational structure to make the most of the context garnered from the discursive nature of tweets. In this section we present experimentation that makes use of conversational structure, comparing its performance with respect to classifiers that ignore the structure.

2.3.1 Problem definition

Within this task we propose leveraging conversation structure as one of the main features that characterise social media Tolmie et al. (2015). So the task becomes one of classifying each tweet in a conversational thread, in the context of the thread. Twitter conversations consist of replies to each other, together forming a tree structure, as shown in the example in Figure 2.1. Replies can be nested in each other, so that the depth of the tree can vary. Hence, in the stance classification task applied to Twitter conversations we have rumours containing a variably sized set of conversations $R_i = \{C_1, ..., C_{|R_i|}\}$. Each of these conversations, C_j, has a varying number of tweets in it. By definition, a conversation has 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 $Replies_{t_{j,1}} = \{t_{j,1,1}, ..., t_{j,1,k}\}$, each of which can in turn receive replies by a varying number k of tweets, e.g., $Replies_{t_{j,1,1}} = \{t_{j,1,1,1}, ..., t_{j,1,1,k}\}$. Thus, we encode the tweet index as a sequence of ids of consecutive children of a preceding node, while traversing the conversation structure.

2.3.2 Experiment Design

In this section we describe the classifiers, features and evaluation measures we used in our experiments.

Classifiers

Conditional Random Fields (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. In contrast 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 Lafferty et al. (2001), 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 2.2 Sutton and McCallum (2011).

$$p(y|x) = \frac{1}{Z(x)} \prod_{a=1}^{A} \Psi_a(y_a, x_a) \tag{2.2}$$

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.[8] First, we use a linear-chain CRFs (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. 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 tweet will repeatedly occur as the first tweet in every branch extracted from a tree.[9]

[8] We use the PyStruct to implement both variants of CRF Müller and Behnke (2014).

[9] 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 is consistent and there is no need to

Maximum Entropy classifier (MaxEnt). As the non-sequential equivalent of CRF, we use a Maximum Entropy (or logistic regression) classifier, which is also a conditional classifier but which operate at the tweet level, ignoring the conversational strucsture. This enables us to compare directly the extent to which treating conversations as sequences instead of having each tweet as a separate unit can boost the performance of the classifier.

Additional baselines. We also compare four more non-sequential classifiers[10]: Naive Bayes (NB), Support Vector Machines (SVM), Random Forests (RF), and Majority (i.e., a dummy classifier always labelling the most frequent class).

We experiment in an 8-fold cross-validation setting. Seven events are used for training and the remainder event is used for testing. 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. For evaluation purposes, we aggregate the output of all eight runs as the micro-averaged evaluation across runs.

Features

We use four types of features to represent the tweets. Note that all of them are local features extracted from the tweet itself and independent of the rest of the conversation, hence enabling us to focus our comparison on how using the sequential structure can impact on the results.

Feature type #1: Lexicon.

- *Word Embeddings:* a vector with 300 dimensions averaging vector representations of the words in the tweet using Word2Vec Mikolov et al. (2013). The Word2Vec model for each of the eight folds is trained from the collection of tweets pertaining to the seven events in the training set, so that the event (and the vocabulary) in the test set is unknown.
- *Part of speech (POS) tags:* a vector where each feature represents the number of occurrences of a type of POS tag in the tweet. The vector is then composed of the numbers of occurrences of different POS tags in the tweet, parsed using Twitie Bontcheva et al. (2013).
- *Use of negation:* binary feature determining if a tweet has a negation word or not. We use a list of negation words, including: 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:* binary feature determining if 'bad' words are present in a tweet. We use a list of 458 bad words[11].

Feature type #2: Content formatting.

aggregate different outputs.

[10]We use their implementation in the scikit-learn Python package

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

- *Tweet length:* the length of the tweet in number of characters.
- *Capital ratio:* the ratio of capital letters among all alphabetic characters in the tweet.
- *Word count:* the number of words in the tweet, counted as the number of space-separated tokens.

Feature type #3: Punctuation.

- *Use of question mark:* binary feature for the presence or not of question marks in the tweet.
- *Use of exclamation mark:* binary feature for the presence or not of exclamation marks in the tweet.
- *Use of period:* binary feature for the presence or not of periods in the tweet.

Feature type #4: Tweet formatting.

- *Attachment of URL:* binary feature, capturing the use or not of URLs in the tweet.
- *Attachment of picture:* binary feature that determines if the tweet has a picture attached.
- *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; there is no need to make use of the conversational structure to extract this feature.

2.3.3 Results

Table 2.8 shows the results comparing performance of the different classifiers, both in terms of micro- and macro-F1 scores, and F1 scores by class. Due to the fact that the dataset is clearly imbalanced with a skew towards *commenting* tweets, we observe that even the majority classifier performs very well in terms of micro-averaged F1 score. In fact, the majority classifier is only slightly outperformed by other classifiers if we look at this evaluation measure. This is why we argue for an evaluation based on macro-averaged F1 score, which accounts for the ability of classifiers to produce an output that better fits to the distribution of classes. Interestingly, we observe that the conditional classifiers (i.e., MaxEnt, Linear CRF and Tree CRF) perform substantially better than the rest in terms of macro-averaged F1 score, which are the only ones to achieve a score of at least 0.4. Comparison of macro-averaged F1 scores of these three classifiers shows that the Tree CRF slightly outperforms the Linear CRF, while both perform significantly better than the non-sequential Maximum Entropy classifier. These results therefore do suggest that exploiting the sequential structure of conversations can lead to improvements on stance classification in rumourous Twitter conversations using the same set of local features.

When we look at the performance by class, we can observe that classifiers performing well only in terms of micro-averaged F1 have the tendency to perform well for the majority class (comments). Interestingly, CRF classifiers using conversational structure show remarkable improvements for other classes, especially supporting and querying tweets, where Tree CRF performs the best. However, all classifiers struggle to classify denials, with performance scores comparable to the other categories. We believe that one of the main reasons for this is that denials are one of the minority classes in the dataset. While querying tweets are also rare, some of the features like question marks are highly indicative of a tweet being a query, and hence they are easier to classify. Denials may in turn have significant commonalities with comments, given that the latter may also use negating words which may seem like denials. As shown in the confusion matrix for the Tree CRF in Table 2.9, the majority class *commenting* is being chosen in as many as 75.8% of the cases by the classifier for those tweets that are actually denials. Collection of additional denying tweets may be of help to improve performance in this class.

Classifier	Micro-F1	Macro-F1	S	D	Q	C
Majority	0.643	0.196	0.000	0.000	0.000	0.783
SVM	**0.676**	0.292	0.372	0.000	0.000	**0.796**
Random Forest	0.666	0.357	0.360	0.022	0.260	0.787
Naive Bayes	0.175	0.203	0.435	**0.147**	0.169	0.060
MaxEnt	0.666	0.400	0.352	0.062	0.396	0.789
Linear CRF	0.646	0.433	0.454	0.105	0.405	0.767
Tree CRF	0.655	**0.440**	**0.462**	0.088	**0.435**	0.773

Table 2.8: Micro- and Macro-F1 performance results, and F1 scores by class (S: supporting, D: denying, Q: querying, C: commenting)

	S	D	Q	C
S	**366 (40.4%)**	32 (3.5%)	22 (2.4%)	487 (53.7%)
D	38 (11.1%)	**22 (6.4%)**	23 (6.7%)	260 (75.8%)
Q	11 (3.1%)	10 (2.8%)	**149 (41.6%)**	188 (52.5%)
C	261 (9.0%)	91 (3.1%)	133 (4.6%)	**2,421 (83.3%)**

Table 2.9: Confusion matrix for Tree CRF (S: supporting, D: denying, Q: querying, C: commenting).

For comparison with the state-of-the-art stance classification approach by Lukasik et al. (2016), we present results broken down by event in Table 2.10, both for their approach based on Hawkes Processes as well as our Tree CRF approach. Note that Lukasik et al. (2016) only tested their approach on four of the events, and therefore performance scores for the rest of the events are not shown. As can be observed from the four events for which we have comparable results, the Hawkes Process performs better in terms of micro-F1, and therefore accurately classifying more instances. However, the Tree CRF performs substantially better in terms of macro-F1, which shows Tree CRF's ability to better estimate the distribution of labels in what is a highly imbalanced task and hence favouring the use of conversational structure in the classification process. We deem this a strong factor in this case as even a simple majority classifier achieves high micro-F1 scores, and the challenge lies in boosting macro-F1 scores to better balance the classification.

	Tree CRF		HP Lukasik et al. (2016)	
Event	Micro-F1	Macro-F1	Micro-F1	Macro-F1
ottawashooting	0.629	**0.457**	**0.678**	0.323
ferguson	0.559	**0.390**	**0.684**	0.260
charliehebdo	0.686	**0.427**	**0.729**	0.326
sydneysiege	0.677	**0.495**	**0.686**	0.325
germanwings-crash	0.694	0.523	–	–
putinmissing	0.660	0.446	–	–
prince-toronto	0.670	0.518	–	–
ebola-essien	0.629	0.384	–	–

Table 2.10: Micro- and Macro-F1 performance results broken down by event, along with a comparison with the results obtained by Lukasik et al. (2016)'s state-of-the-art approach based on Hawkes Processes, where available.

To better understand the effect of exploiting sequential structure, we break down performance scores by the depth of tweets. By this we want to see if the sequential classifiers are consistently performing well across tweets of different depth within conversations. Figure 2.2 shows these results for tweets from depth 0 (source tweet) to depth 9. Further depths are omitted due to the small number of instances available. When we look at micro-averaged scores, we do not see a big performance difference across classifiers, except for the CRF classifiers performing better for source tweets; this is due to the fact that most of the source tweets tend to support a rumour, and hence sequential classifiers can learn this.

What is more interesting is to look again at the macro-averaged scores, where we see that the sequential approaches, especially the Tree CRF, consistently performs well for different levels of depth. More specifically, Tree CRF performs best in 7 out of 10 levels of depth analysed, with Linear CRF being better once (depth = 2) and Maximum Entropy being better twice (depth = 4 and 5).

2.4 Veracity classification

Veracity classification aims to verify whether the posts or tweets spread in social media are true or false. Similar to the stance classification section, we first briefly review related work on veracity classification. In Section 2.4.2 we again outline our classification approach by describing our features and classification methods. Next we present and discuss the classification experiments. As performed for the stance classifier we also packaged the developed approach as a GATE plugin. Details about the plugin are provided in Section 2.4.4.

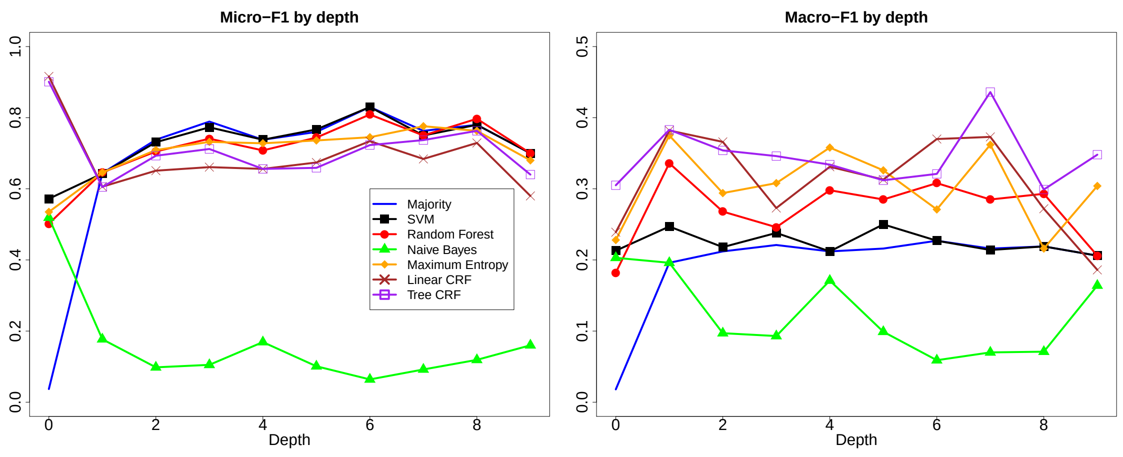

Figure 2.2: Micro- and macro-F1 scores by depth of tweet.

2.4.1 Related work

Castillo et al. (2011) distinguish microblogs that are "NEWS" and "CHAT" where the former reports an event or fact that can be of interest of others and the latter is a message that is purely based on personal/subjective opinions and/or conversations among friends. The microblogs in the NEWS are investigated for rumour credibility. Decision trees based on J48 are used to train classifiers to classify microblogs into NEWS and CHAT categories and then classify the microblogs in the NEWS category by whether they are credible or not – the authors report that they used various other machine learning approaches such as Bayes networks and SVM but mention that decision trees based on J48 were superior to them. In both cases microblogs collected from Twitter are used and manually annotated through Mechanical Turk. The authors use four categories of features: message-based, user-based, topic-based, propagation-based features. The message-based features consider characteristics of messages such as the length of a message, whether the message contains exclamation/question mark, number of positive/negative sentiment words, whether the message contains a hashtag and whether it is a re-tweet. The user-based features entail information about the user such as registration age, number of followers, number of followees and the number of tweets the user has authored in the past. Topic-based features aggregate information from the previous two feature types such as the fraction of tweets that contain URLs, the fraction of tweets with hashtags, etc. Finally, the propagation-based features consider characteristics related to the messaging tree such as depth of the retweet tree or the number of initial tweets of a topic. In the NEWS/CHAT classification task the authors report performances scores of 89.2% in terms of accuracy, 89.1% in precision, 89.1% in recall and 89.1% in F1-measure. In the credibility classification an accuracy of 86% is reported (the same figure is reported for precision/recall and F-1 score).

Kwon et al. (2013) investigate also the rumour veracity classification problem following a supervised approach. The authors propose their own dataset and make it publicly available. They also propose features which fall into the following categories: temporal, structural and linguistic. The temporal features aim to capture how rumours spread over time. The structural features model the connectivity between the users who posted about the rumour. Finally, the linguistic features are obtained through the Linguistic Inquiry and Word Count (LIWC) dictionaries Tausczik and Pennebaker (2010). As baseline features proposed by Castillo et al. (2011) are adopted. Using Random Forest and Logistic Regression the authors perform feature selection on the proposed categories of features to find the significant features. Using these significant features and three different classifiers (Decision tree, Random Forest and SVM) they perform the rumour veracity classification. The results show that features found as significant by Random Forest, and Random Forest as the classifier lead to best performance in terms of accuracy (90%), precision (93.5%), recall (89.2%) and F1-measure (89.3%). The best results using the baseline features adopted from Castillo et al. (2011) are obtained through SVM with 81.1% accuracy, 89.1% precision, 75.3% recall and 78.8% F1-measure. The authors also show that the combination of significant features identified by Random Forest and baseline features lead to quality degradation.

Slightly later, Yang et al. (2012) tackle also the veracity classification of microblogs but use instead of Twitter the messaging platform Sina Weibo – so to say a Chinese version of Twitter. The authors adopt features from earlier studies and extend them by two features: client-based and location-based features. The client-based features entail information about the software that was used to perform the messaging. The location-based features hold information on whether the message was sent within the same country where the event happened or not. The authors report that adding these two features on top of earlier reported features leads to a substantial boost on the accuracy figures of veracity classification. For instance, adding the two features on top of the propagation-based features reported by Castillo et al. (2011) leads to an increase of 6.3% accuracy. Unfortunately, the authors do not combine all the features and report results on them. For the classification purpose the authors use SVM with the RBF kernel Buhmann (2003).

More, recently, Liu et al. (2015) use approaches reported by Yang et al. (2012) and Castillo et al. (2011) as baseline systems and compare them against their proposed approach that make use of so called “verification features”. These features are determined based on insights from journalists and include source credibility, source identity, source diversity, source and location witness, event propagation and belief identification. In belief identification results of rumour stance classification are used as features. The authors show that the proposed approach outperforms the two baselines. They also show that when adding the belief identification to the other features the results are significantly better than when not adding them to the feature set. The best results range from 78% to 89% in terms of accuracy depending on how many tweets are used to verify the rumour (5 tweets to 400 tweets). The experiments are performed on their own data set using SVM as a classifier. However, the authors mention that they also investigated Random Forest and decision trees but SVM gave the best results. Unfortunately the paper does not provide much detail about this comparison.

In the same year, Ma et al. (2015) proposed to model the features over time to perform message veracity classification. The authors adopt features from earlier studies (as described above) as well as machine learning approaches used in those studies (J48, SVM with the RBF kernel). Experiments are performed on Twitter as well as on Sina Weibo datasets. Ma et al. use SVM with linear kernel and report that this linear SVM combined with the proposed approach that models the features over time leads to best performance. On the Twitter dataset reported by Castillo et al. (2011) the authors report an accuracy of 89%. For Sina Weibo, the authors collect their own dataset and run existing approaches on them. The proposed setting reaches an accuracy of 84.6% where decision trees with J48 leads to 77.4%, SVM with the RBF kernel to 77.9% and random forests to 81.5%.

Another study that tackles rumour veracity classification on Sina Weibo is reported by Wu et al. (2015). The authors use message propagation trees to extract features from tweets. Three categories of features are considered: message-based, user-based and repost-based. Two systems reported by earlier studies Castillo et al. (2011) and Yang et al. (2012) are adopted for the evaluation. As the machine learning approach, the authors use SVM with a hybrid kernel technique consisting of random walk kernel Borgwardt et al. (2005) and RBF kernel. The results reported are in favour of the proposed hybrid approach leading to 91% accuracy. The baselines achieve 85.4% Castillo et al. (2011) and 77.2% Yang et al. (2012). In the experiments the authors use rumours with at least 100 reposts. This opens the question about how well the proposed approach will perform when it is applied to newly born rumours where there are only few messages available.

Vosoughi (2015) tackles the veracity classification problem using three categories of features (linguistic, user oriented and temporal propagation related features) and speech recognition inspired machine learning approaches such as Dynamic Time Wrapping (DTW) and Hidden Markov Models (HMMs). Evaluations are performed on Twitter data gathered by the author. Results show that HMMs with 75% accuracy are superior to DTWs that lead to only 71% accuracy. The authors also report that the best performing features are those in the temporal propagation category leading to 70% accuracy. The linguistic features lead to 64% and the user oriented features to 65% accuracy. It is also important to note that the authors, similar to Wu et al. (2015), work with rumours that exceed a popularity threshold, in this case 1000 tweets.

More recently, Chua and Banerjee (2016) published an analysis of various features on the tweet veracity classification task. The authors analysed six categories of features: comprehensibility, sentiment, time-orientation, quantitative details, writing style and topic. Rumours gathered by the authors are used along with the binomial logistic regression to tackle the task in a supervised fashion. Unlike previous studies, Chua et al. report only features that are significantly important rather than an indication of the overall performance of the classifier. These features are: negation words (comprehensibility category), past, present, future POS in the tweets (time-orientation category), discrepancy, sweat and exclusion features (writing style category) and finally home, leisure, religion and sex topic features (topic category).

classifier	**in domain tweets (in %)**	**Ottawa shooting**	**ferguson riots**	**Charlie Hebdo**	**German wings**	**Sydney siege**	**macro mean (Acc)**
SVM	0	64.52	94.12	50.46	83.33	66.0	71.69
Random Forest	0	75.85	94.12	76.44	84.34	69.52	80.05
Bayes	0	81.33	94.21	74.73	84.59	79.85	82.94
J48	0	79.65	94.46	80.56	87.44	82.24	84.87
IBk	0	89.12	94.29	85.36	95.9	82.0	**89.33**

Table 2.11: Different classifier performances on veracity classification using the setting (1). Setting (1) does not contain any testing data in the training process.

2.4.2 Method

As for the development of the stance classification component, we use the same classifiers with the same parameter settings. For more details see Section 2.2.2. We also make use of all the features discussed in Section 2.2.2. In addition to those features we use the output of the stance classifier as feature:

- **Stance class**: We use the stance classifier described in Section 2.2 to predict the stance for each tweet. This predicted output is used for the veracity classification as additional feature along the features described in Section 2.2.2.

2.4.3 Experimental setup and results

For the veracity classification task, we use the same dataset as well as the same settings for training and testing described in Section 2.2.3. Similar to the stance classification, we perform the veracity classification at the tweet level. This means each tweet is either classified as “1” (the rumour discussed in the tweet is classified as true) or “0” (the rumour discussed in the tweet is classified as false). As in stance classification we first analyse the different machine learning approaches on the veracity classification task. Results of this analysis are shown in Table 2.11. From the results we can see that the best performing approach is the instance based learning (IBk) method. Its performance is above 89% accuracy on the setting 1 where no testing data is used in the training. The least performing approach is the SVM classifier leading to only 71% accuracy. Because of its great performance we will be using only the IBk classifier for the veracity classification task.

Using the IBk classifier we perform the veracity classification using all the features described in Section 2.2.2 excluding the stance feature (**veracity classification without stance** as well as after adding the stance feature on top of those features (**veracity classification with stance**). Both results are shown in Tables 2.12 and 2.13. From the results we can see that both settings (with and without stance feature) achieve accuracy values around 90%. Furthermore, we also see that the accuracy values increase constantly the

in domain tweets (in %)	**Ottawa shooting**	**ferguson riots**	**Charlie Hebdo**	**Germanwings**	**Sydney siege**	**macro mean (Acc)**
0	64.52	94.12	50.0	83.33	66.0	71.59
0	89.12	94.29	85.36	95.9	82.0	89.33
10	89.25	94.45	85.89	98.02	82.93	90.11
20	90.13	94.49	87.45	98.23	83.6	90.78
30	89.88	94.55	89.55	98.46	84.42	91.37
40	91.19	94.96	90.1	98.19	85.12	91.91
50	91.89	94.65	91.14	98.68	85.67	92.41
60	91.89	94.44	91.79	99.4	85.91	92.69

Table 2.12: Veracity classification results in accuracy obtained using the instance based learning method (IBk). Results are obtained without the stance feature. The first line entails the majority voting results.

in domain tweets (in %)	**Ottawa shooting**	**ferguson riots**	**Charlie Hebdo**	**Germanwings**	**Sydney siege**	**macro mean (Acc)**
0	92.71	94.29	86.0	96.99	83.03	90.6
10	93.03	94.45	87.57	98.38	84.14	91.51
20	92.99	94.49	88.48	98.65	84.42	91.81
30	93.16	94.55	90.69	98.46	85.09	92.39
40	93.45	94.96	91.74	98.19	85.61	92.79
50	94.61	94.65	91.98	98.68	86.17	93.22
60	94.89	94.44	92.48	99.4	86.24	93.49

Table 2.13: Veracity classification results in accuracy obtained using the instance based learning method (IBk). Results are obtained with the stance feature.

more testing instances are added to the training data. Finally, the results show that the use of stance as additional feature helps improve the performance of the veracity classification.

As discussed in Section 2.2.3, accuracy may be affected by imbalance data and thus may be biased towards the dominant class. Thus we also report the F1 measure (see Table 2.14)[12]. Finally, we also report the confusion matrix in Table 2.15 to analyse the classifier's confusion in classification.

2.4.4 GATE plugin

As with the stance classification component, we packaged the veracity classifier as a GATE-plugin. Pre-processing steps as in stance classification are applied to process the

[12]These results are obtained with the stance as addition feature.

in domain tweets (in %)	**Ottawa shooting**	**ferguson riots**	**Charlie Hebdo**	**Germanwings**	**Sydney siege**	**macro mean (F1)**
0	95.78	94.44	91.87	98.42	88.62	93.83
10	95.94	94.74	92.77	99.16	89.5	94.42
20	95.87	94.82	93.38	99.31	89.76	94.63
30	96.02	94.92	94.77	99.21	90.15	95.01
40	96.21	95.42	95.38	99.07	90.55	95.33
50	96.8	95.02	95.53	99.31	90.85	95.5
60	96.94	94.71	95.78	99.69	90.77	95.58

Table 2.14: IBk classification results in weighted F1 over the 2 classes.

	false	true
false	155.6	17.6
true	5.6	133.8

Table 2.15: Confusion matrix for the IBk classifier after adding 60% testing instances into the training data. The figures are averaged over the 5 data sets

tweets. In addition, the stance classifier plugin described in Section 2.2.4 is added as additional pre-processing step as the output of the stance classification is used as feature in the veracity classification. Apart from these pre-processing resources, similar to the stance classifier plugin, the GATE learning framework is required to perform the prediction. The entire process is saved as a pipeline and can be loaded into GATE.

Chapter 3

Discussion

We have evaluated the development of an automatic processing pipeline that detects misinformation and disinformation that start off as rumours in the context of breaking news. Such a pipeline has been developed, as consisting of three different stages and corresponding components: (1) *rumour detection*, which identifies reports that are unverified at the time of posting; (2) *stance classification*, which determines how different posts discussing a rumour support, deny, query or comment on it; and (3) *veracity classification*, which determines if the information underlying a rumour is truthful or not. The combination of all three components in a single pipeline can ultimately identify misinformation and disinformation having a timeline of tweets associated with an event as input.

The rumour detection task has proven challenging, as only a fourth of the reports we observe constitute rumours. Despite this being a low percentage, we have also found that it is crucial to promptly identify them as they tend to be widely spread. The brevity of a tweet posits a challenge for rumour detection, where tweets do not necessarily provide evidence of being a rumour, such as the use of hedging words such as 'reportedly'. Given that the content of a tweet may not be indicative of being a rumour, we have explored the use of a sequential classifier that leverages the context learnt throughout the event, so that the classifier can analyse how the tweets fits in the story. Through comparison with other baseline, non-sequential classifiers, we have found that the sequence is indeed of great help to identify rumours, leading to over 40% improvement in performance.

The subsequent task on stance classification has been explored in two different settings. The first is by classifying the stance of tweets in isolation, where one looks at the tweet's content. This is handy for its application in a streaming scenario, where tweets pour in and it is not straightforward to put together related context. To complement this, the second setting has been developed by exploiting the conversational structure produced from people responding to rumourous tweets. Where tweets respond to one another, it is easier to identify relationships and therefore build context. The exploitation of the conversational structure for stance classification has been performed by using a sequential classifier. Interestingly, we have found that exploiting the conversational structure a classifier can boost performance as it has more context. The caveat of the latter approach is that its application to a streaming scenario is very challenging, given that conversations are not retrieved in the same stream of tweets, and Twitter does not provide a suitable API endpoint that enables to retrieve this data easily. We believe that a hybrid approach that mostly relies in isolate tweets, but makes use of the conversational structure of tweets where this is available, can lead to a competitive and scalable performance.

The final task, namely the veracity classification task, has been tackled at the tweet level, classifying the rumour underlying each tweet as true or false. Using features inferred from the content of the tweet, we show that accuracy values of around 90% can be achieved.

The three components rely on numerous language-independent features which makes them easily adaptable to other languages. Such features are for instance those which are based on statistics derived from the tweets themselves, as well as contextual features that can be learnt throughout the event. There are only a few features which rely on some manually created thesauri such as *suprisescore* but those thesauri are small and can be easily collected for each new language. However, there are also some features such as POS, sentiment, mood, named entity and negations. These features require pre-processing resources, which can sometimes be hard to find for less studied languages. Porting these features thus assumes the availability of such resources in the respective language.

Despite the fact that our rumour detection component largely relies on context learnt throughout the event, one of the limitations in stance and veracity classification is that we narrowed our experiments only to tweets. We believe that knowledge beyond the tweets may play a major role in further improving performance on these tasks. Such additional knowledge could be e.g. derived through the URLs contained in the tweets and used to achieve better performance. We are currently working on this where we aim integrate deep learning on related news articles and use the resulting models in the prediction processes. We do, however, leverage the sequence of the stream of tweets, which enables us to garner additional knowledge about the discursive nature of rumours. While discourse has been exploited for rumour detection and stance classification, we also plan to test it for veracity classification.

Finally, two of the components presented in this deliverable, i.e. the rumour stance classification and rumour veracity classification components, have been released as open source and also integrated into the GATE text analytics framework (https://gate.ac.uk) as plugins. The code is also available from the PHEME website (http://www.pheme.eu/software-downloads).

Bibliography

Baroni, M., Dinu, G., and Kruszewski, G. (2014). Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. In *ACL (1)*, pages 238–247.

Bontcheva, K., Derczynski, L., Funk, A., Greenwood, M. A., Maynard, D., and Aswani, N. (2013). TwitIE: An open-source information extraction pipeline for microblog text. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing*. Association for Computational Linguistics.

Borgwardt, K. M., Ong, C. S., Schönauer, S., Vishwanathan, S., Smola, A. J., and Kriegel, H.-P. (2005). Protein function prediction via graph kernels. *Bioinformatics*, 21(suppl 1):i47–i56.

Buhmann, M. D. (2003). Radial basis functions: theory and implementations. *Cambridge Monographs on Applied and Computational Mathematics*, 12:147–165.

Castillo, C., Mendoza, M., and Poblete, B. (2011). Information credibility on twitter. In *Proceedings of the 20th international conference on World wide web*, pages 675–684. ACM.

Celli, F., Ghosh, A., Alam, F., and Riccardi, G. (2016). In the mood for sharing contents: Emotions, personality and interaction styles in the diffusion of news. *Information Processing & Management*, 52(1):93–98.

Chen, D. and Manning, C. (2014). A fast and accurate dependency parser using neural networks. In *Proceedings of EMNLP 2014*.

Chua, A. Y. and Banerjee, S. (2016). Linguistic predictors of rumor veracity on the internet. In *Proceedings of the International MultiConference of Engineers and Computer Scientists*, volume 1.

Cunningham, H., Maynard, D., and Bontcheva, K. (2011). *Text processing with gate*. Gateway Press CA.

Derczynski, L., Lukasik, M., Srijith, P., Bontcheva, K., Hepple, M., Lobo, T. P., and Radzimski, M. (2016). D6. 2.1 evaluation report-interim results. In *PHEME Project Deliverable*.

GPy (2012–2015). GPy: A Gaussian process framework in Python. `http://github.com/SheffieldML/GPy`.

Guo, W. and Diab, M. (2012). Modeling sentences in the latent space. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 864–872. Association for Computational Linguistics.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., and Witten, I. H. (2009). The weka data mining software: an update. *ACM SIGKDD explorations newsletter*, 11(1):10–18.

Hamidian, S. and Diab, M. (2015a). Rumor detection and classification for twitter data. *The Fifth International Conference on Social Media Technologies, Communication, and Informatics, SOTICS, IARIA*, pages 71–77.

Hamidian, S. and Diab, M. T. (2015b). Rumor detection and classification for twitter data. In *Proceedings of the Fifth International Conference on Social Media Technologies, Communication, and Informatics (SOTICS)*.

Hamidian, S. and Diab, M. T. (2016). Rumor identification and belief investigation on twitter. In *Proceedings of NAACL-HLT*, pages 3–8.

Karlova, N. and Fisher, K. (2012). Plz RT: A social diffusion model of misinformation and disinformation for understanding human information behaviour. In *Proceedings of an International Conference on Information Seeking in Context*, ISIC '12.

Kwon, S., Cha, M., Jung, K., Chen, W., and Wang, Y. (2013). Prominent features of rumor propagation in online social media. In *2013 IEEE 13th International Conference on Data Mining*, pages 1103–1108. IEEE.

Lafferty, J., McCallum, A., and Pereira, F. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the eighteenth international conference on machine learning, ICML*, volume 1, pages 282–289.

Liu, X., Nourbakhsh, A., Li, Q., Fang, R., and Shah, S. (2015). Real-time rumor debunking on twitter. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 1867–1870. ACM.

Lukasik, M., Cohn, T., and Bontcheva, K. (2015). Classifying tweet level judgements of rumours in social media. *arXiv preprint arXiv:1506.00468*.

Lukasik, M., Srijith, P. K., Vu, D., Bontcheva, K., Zubiaga, A., and Cohn, T. (2016). Hawkes processes for continuous time sequence classification: an application to rumour stance classification in twitter. In *Proceedings of the 54th Meeting of the Association for Computational Linguistics*, pages 393–398. Association for Computer Linguistics.

Ma, J., Gao, W., Wei, Z., Lu, Y., and Wong, K.-F. (2015). Detect rumors using time series of social context information on microblogging websites. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 1751–1754. ACM.

Mendoza, M., Poblete, B., and Castillo, C. (2010a). Twitter under crisis: Can we trust what we RT? In *1st Workshop on Social Media Analytics*, SOMA'10, pages 71–79, Washington, DC, USA.

Mendoza, M., Poblete, B., and Castillo, C. (2010b). Twitter under crisis: can we trust what we rt? In *Proceedings of the first workshop on social media analytics*, pages 71–79. ACM.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

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

Qazvinian, V., Rosengren, E., Radev, D. R., and Mei, Q. (2011). Rumor has it: Identifying misinformation in microblogs. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1589–1599. Association for Computational Linguistics.

Ratkiewicz, J., Conover, M., Meiss, M., Gonalves, B., Flammini, A., and Menczer, F. (2011). Detecting and tracking political abuse in social media. In *5th International AAAI Conference on Weblogs and Social Media*, ICWSM'11, Barcelona, Spain.

Socher, R., Perelygin, A., Wu, J. Y., Chuang, J., Manning, C. D., Ng, A. Y., and Potts, C. (2013). Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the conference on empirical methods in natural language processing (EMNLP)*, volume 1631, page 1642. Citeseer.

Somasundaran, S. and Wiebe, J. (2009). Recognizing stances in online debates. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1*, pages 226–234. Association for Computational Linguistics.

Sutton, C. and McCallum, A. (2011). An introduction to conditional random fields. *Machine Learning*, 4(4):267–373.

Tausczik, Y. R. and Pennebaker, J. W. (2010). The psychological meaning of words: Liwc and computerized text analysis methods. *Journal of language and social psychology*, 29(1):24–54.

Tolmie, P., Procter, R., Rouncefield, M., Liakata, M., and Zubiaga, A. (2015). Microblog analysis as a programme of work. *arXiv preprint arXiv:1511.03193*.

Vosoughi, S. (2015). *Automatic detection and verification of rumors on Twitter*. PhD thesis, Citeseer.

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

Wu, K., Yang, S., and Zhu, K. Q. (2015). False rumors detection on sina weibo by propagation structures. In *2015 IEEE 31st International Conference on Data Engineering*, pages 651–662. IEEE.

Yang, F., Liu, Y., Yu, X., and Yang, M. (2012). Automatic detection of rumor on sina weibo. In *Proceedings of the ACM SIGKDD Workshop on Mining Data Semantics*, page 13. ACM.

Zeng, L., Starbird, K., and Spiro, E. S. (2016). # unconfirmed: Classifying rumor stance in crisis-related social media messages. In *Tenth International AAAI Conference on Web and Social Media*.

Zhao, Z., Resnick, P., and Mei, Q. (2015). Enquiring minds: Early detection of rumors in social media from enquiry posts. In *Proceedings of the 24th International Conference on World Wide Web*, pages 1395–1405. ACM.

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