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: Evaluation Report – Final Results

Deliverable number: D6.2.2

Deliverable date: 31 January 2017

Partners responsible: Universitaet des Saarlandes (USAAR)

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 (.....)

Sent to Project Officer:	Sent to functional mail box:	On date:
Susan.Fraser@ec.europa.eu	*CNECT-ICT*-611233*@ec.europa.eu*	17 February 2017

D6.2.2 Evaluation report - Final Results

Leon Derczynski; Michał Lukasik; Ahmet Aker; Kalina Bontcheva, University of Sheffield
Thierry Declerck; Piroska Lendvai, University of Saarland
Arkaitz Zubiaga; Maria Liakata; Rob Procter, University of Warwick
Tomás Pariente Lobo; Mateusz Radzimski, ATOS

Abstract.
FP7-ICT Collaborative Project ICT-2013-611233 PHEME
Deliverable D6.2.2 (WP6)

The deliverable describes the results of Task 6.4 in WP6 on the quantitative evaluations of the PHEME algorithms and their integration. Following the description of work, the datasets created in Task 2.1, WP7, and WP8, are used for iterative development and parameter tuning of the PHEME content analytics methods from WP3 and WP4, as well as for testing their integration into a processing pipeline. The scalability of the integrated tools is also evaluated.
Keywords: Evaluation, contradiction detection, rumour classification, integration and performance evaluation.

Project	PHEME No. 611233
Delivery Date	17 February 2016
Contractual Date	January 31, 2017
Nature	Report
Reviewed By	N/A
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

Recently people have started using social media not only to keep in touch with family and friends, but also increasingly as a news source. However, knowledge gathered from online sources and social media comes with a major caveat – it cannot always be trusted. Rumours, in particular, tend to spread rapidly through social networks, especially in circumstances where their veracity is hard to establish. For instance, during an earthquake in Chile rumours spread through Twitter that a volcano has become active and there was a tsunami warning in Valparaiso (Mendoza et al., 2010). This creates a large and real-time need for veracity assessments and feedback for social media data.

To build a research system for rumour detection and classification, we need accurate tools that can operate on very noisy text, in a variety of languages. These are vital to catching rumours as they emerge and providing the most possible information to stakeholders. Such tools often consist of multiple components and can be divided into discrete subparts. Each of these parts must be able to tolerate the variances of user-generated content (UGC) in the respective language. This places performance constraints on the system in terms of *quality*. The tools need to capture and provide enough information to enable accurate rumour recognition and classification, which is a novel demand which PHEME addresses.

Additionally, these tools also need to be able to inter-operate, and handle high volume streaming content in a timely fashion. Therefore, there are not only *quality* performance constraints on the system, but also computational performance constraints. Each service must be able to process information at an acceptable rate, handle bursts, and handle failure elegantly. Above this, common formats must be agreed by the consortium for exchanging data in a consistent and comprehensible way. To achieve such a format, we all need to know which information to add in order to supply complete information to other components in the system. Between the variety of languages, partners and subsystems in the consortium, this poses a challenging task.

This deliverable builds on work reported in the earlier Deliverable D6.2.1 and reports the final evaluation for the methods and tools developed in WP3 and WP4 in PHEME. We perform quantitative evaluation of the integrated components, as well as describing in-situ the efficiency of the integrated Kafka pipeline for rumour processing.

Contents

Chapter 1

Introduction

Recently people have started using social media not only to keep in touch with family and friends, but also increasingly as a news source. However, knowledge gathered from online sources and social media comes with a major caveat – it cannot always be trusted. Rumours, in particular, tend to spread rapidly through social networks, especially in circumstances where their veracity is hard to establish. For instance, during an earthquake in Chile rumours spread through Twitter that a volcano has become active and there was a tsunami warning in Valparaiso (Mendoza et al., 2010). This creates a large and real-time need for veracity assessments and feedback for social media data.

To build a research system for rumour detection and classification, we need accurate tools that can operate on very noisy text, in a variety of languages. These are vital to catching rumours as they emerge and providing the most possible information to stakeholders. Such tools often consist of multiple components and can be divided into discrete subparts. Each of these parts must be able to tolerate the variances of user-generated content (UGC) in the respective language. This places performance constraints on the system in terms of *quality*. The tools need to capture and provide enough information to enable accurate rumour recognition and classification, which is a novel demand which PHEME addresses.

Additionally, these tools also need to be able to inter-operate, and handle high volume streaming content in a timely fashion. Therefore, there are not only *quality* performance constraints on the system, but also computational performance constraints. Each service must be able to process information at an acceptable rate, handle bursts, and handle failure elegantly. Above this, common formats must be agreed by the consortium for exchanging data in a consistent and comprehensible way. To achieve such a format, we all need to know which information to add in order to supply complete information to other components in the system. Between the variety of languages, partners and subsystems in the consortium, this poses a challenging task.

This deliverable builds on work reported in the earlier Deliverable D6.2.1 and reports the final evaluation for the methods and tools developed in WP3 and WP4 in PHEME. We

perform quantitative evaluation of the integrated components, as well as describing in-situ the efficiency of the integrated Kafka pipeline for rumour processing.

Chapter 2 examines all the content analysis methods built so far. This covers sub-story event detection, contradiction detection and rumour classification.

Chapter 3 puts forth the technical aspects of integration in PHEME.

1.1 Relevance to PHEME

The PHEME project aims to detect and study the emergence and propagation of rumours in social media, which manifest as dubious or false claims. In order to do this, there are many empirical and technical processes that need to have high quality performance and be inter-operable. This deliverable serves to assess our progress towards both these goals.

1.1.1 Relevance to project objectives

Producing integrated research on rumour detection is a key goal of PHEME, and so we require a prototype system for sharing results and demonstrating that our outputs work not only in theory but also in practice.

In particular, this deliverable reports on a number of quantitative evaluation experiments, and thus contributes directly to objective 5, defined in the PHEME description of work, as follows:

> Test and evaluate the newly developed methods through (i) quantitative experiments on gold-standard data, acquired both through traditional domain expert annotation and crowdsourcing; and (ii) qualitative assessments in the use cases on health and digital journalism, involving key stakeholders from two focus groups.

The focus of Task 6.4 and D6.2.2 is entirely quantitative, while qualitative assessment with stakeholders is undertaken in the respective use cases (WP7 and WP8).

1.1.2 Relation to forthcoming and prior research in PHEME

This deliverable provides a single point of evaluation for much of the content analysis work in WP3 (Contextual Interpretation) and WP4 (Detecting Rumours and Veracity). Specifically, the latest results of T3.3 in WP3 are evaluated, as well as evaluation results from WP4 on contradiction detection, rumour stance detection and veracity classification. The results and findings from this document should be read in conjunction with the first evaluation round, described in D6.2.1, which covered the evaluation of tools from WP2, as well as earlier versions of the WP3 and WP4 methods evaluated here.

1.1.3 Relation to other work packages

The datasets created in Task 2.1, WP7, and WP8 are used for iterative development and evaluation. This includes tuning of and reporting on content analytics methods from WP3 and WP4.

Chapter 2

Method Evaluation

This chapter examines the latest content analysis methods built in the course of the project and reports on evaluation studies for corresponding work in WP3 (Contextual Interpretation) and WP4 (Detecting Rumours and Veracity) in PHEME. This covers sub-story event detection, contradiction detection and rumour stance detection and veracity classification.

2.1 Sub-Story Detection Evaluation

Grouping documents and messages into coherent clusters is a crucial first step in making collections human-accessible. This becomes more critical as information flows grow in size. Existing techniques can work well, though usually only offline in a batch setting, or with reasonably long documents. While social media offers a wealth more information, applying these techniques to this varied text type, in a real-world setting where humans are presented with clusters as thematic groups, or in real-time, leads to less satisfactory results. The difficulty is compounded by reliance on shingle and n-gram-based methods such as LSH and oHDP, which struggle with short or noisy texts.

However, accurately clustering documents as they emerge and continuing to do so as stories develop is a difficult problem. It is also a useful one: being able to allocate a meaningful categorization to documents in real-time aids many applications.

Social media offers rapid and broad views over many topics, and can be seen as expressing through its messages the latent variable of real-world events. However document clustering is even tougher over social media. The high linguistic variation renders lexical approaches ill-equipped to capture relevant documents, and while it offers explicit topic markers (hashtags), the correct use of these is voluntary and far from guaranteed.

2.1.1 Approach

We take a definition and model of event clusters from results in cognitive science. (D'Argembeau and Demblon, 2012) found that humans readily place observations and memories in distinct clusters of events, sharing causal and thematic links. These may not necessarily be hard clusters, that is, one element may occur in more than one cluster, but a recollection of a cluster of events will, as a cluster in the computational sense, have certain firm commonalities. Indeed, during the process of mental time travel, event clustering is vital to structuring recollection.

Because language is an expression of thoughts and ideas, and n-grams or similar may not reveal the whole picture directly, we prefer to base our approach on a higher-level conceptual framework.

Commonalities between elements in event clusters, from a cognitive point of view (D'Argembeau and Demblon, 2012), can be placed into various groups:

- Locations: common places between memories or descriptions (D'Argembeau and Mathy, 2011)
- Actors: is a particular person or organization pivotal to the theme? (D'Argembeau and Van der Linden, 2012)
- Activities: was a certain occurrence particularly poignant?
- Shared broader events
- Causal relations (Brown and Schopflocher, 1998)

Lexicalisations of these commonalities can take a variety of forms, not necessarily close to each other. This leaves no guarantee that there will be any similar terms between documents discussing the same event. The likelihood of lexical overlap decreases further in social media streams; documents are typically shorter than e.g. news articles, and contain more lexical and orthographic variation. These factors leave dominant n-gram based clustering methods, e.g. LSH (Petrović et al., 2010) and oHDP (Wang et al., 2011), ill-equipped to accurately group together all the texts that a human reader would consider to be discussing the same event.

In general, we have tried to avoid language-dependent methods and reasoning, to ease adaptation of the method to other languages. For example, we prefer not to include parts of speech in any logic, instead leaving definition of event to the ISO-TimeML standard (Pustejovsky et al., 2010). However, this isn't possible everywhere: for example, tokenization, NER and event extraction all have language-specific implementations.

2.1.2 Framework

The goal is to assign messages $M = m_0...m_n$ from a stream S into a set of clusters K. Each message should be assigned to a cluster as it arrives, without waiting on later information, and assignments are final. To achieve this, each incoming message m_i is compared to each cluster $k_0...k_n \in K$ and a matching function $f_{match}(m, k)$ scores whether the message is a good fit for that cluster. The message is then assigned to the cluster having the highest match score, whose internal representation is updated accordingly. This gives a hard clustering; a soft clustering can be achieved by assigning top-k best matches instead. If no cluster matches well enough, then a new cluster is created, consisting of just m_i. A gating threshold t_{match} is used to determine which values of f_{match} can be considered potential matches; if no cluster matches well enough, this triggers new cluster creation. Thus, t_{match} allows control of the rate of cluster creation and how coarse or fine clusters are.

As it stands, this method leads to an ever-increasing number of clusters over time. While this is possibly suitable for batch clusterings, it is not tolerable in a streaming environment. To address this, an upper threshold on the number of clusters t_K is set. Whenever this is exceeded by means of a new cluster creation, a cluster is pruned from K, to keep $|K| = t_K$. We implement two pruning policies, drawing from standard caching; *least-recently-used* or LRU, which takes the cluster that has not been updated for the longest amount of time; and *age*, which takes the cluster with the oldest creation date. This constrains processing time to $O(t_K)$ in this regard.

Each message is converted to a feature representation. This is compared to an internal cluster representation. The overlap between these two is used to generate the final feature representation for the matching function.

2.1.3 Cluster representation

Each event cluster $k_i \in K$ is represented using information about messages that have been to assigned to it and cluster-specific metadata. Event cluster representations are updated when assigned a new message. k is a tuple $\langle A, n, Z, R, \vec{e}, L\rangle$ defined as follows. $A = (a_1, a_2, ..., a_n)$ is a set of attributes. Each attribute a_i is a key:count pair, counting the number of times the key has been observed in messages added to the cluster. Attributes are: hashtags, named entities (as $(surfaceform, entityclass)$ tuples), URLs, username mentions, event lexicalisations, and locations. n is the number of messages that have been assigned to the cluster. Z comprises relevant time points (z_{start}, z_{recent}), the cluster inception time and the datestamp of the most recent message, respectively. R is a set of related message IDs seen in all messages assigned to the cluster, and the number of times they have each been observed in added messages (similar to the tuples in A). $\vec{e}$ is a vector representation of the most frequent event lexicalisation observed in the cluster. Finally, L is a parameterized minhash LSH index (Indyk and Motwani, 1998; Broder, 2000) specific

to the cluster.

To model that clusterings depend on both time and the text of related messages, we explicitly capture the currency, or age, of clusters. This is intuitively a factor in how likely an incoming message is to belong to a cluster. From the cognitive point of view, temporal distance relates well to vividness of episodic thoughts (D'Argembeau and Van der Linden, 2012). Two features capture temporal proximity in clusters, in Z. The difference between the document timestamp and candidate cluster inception is taken, in seconds. Additionally, the time elasped between document timestamp and the last addition to the cluster is taken, to model how "hot" the cluster is. Time values are scaled linearly, from a minimum -1 at identical time to maximum +1 for one day or more's delay.

While lexical overlap is less likely between social media posts, it is not worth discarding entirely. Accordingly, we build a feature comparing the LSH overlap between a candidate posts and a map kept for each cluster. Each cluster is therefore created with a minhash[1] having threshold 0.3 and 128 permutations. When a message is assigned to a cluster, the cluster's minhash is updated with that message's text.

2.1.4 Message Representation

Messages are represented using various entities, temporal and discourse metadata from the message, and finally lexical information. Each message representation $m_i \in M$ is a tuple $\langle A^m, r_{id}, z_{stamp}, R^m, \vec{h}, H \rangle$ thus. $A^m = (a_1^m, a_2^m, ..., a_n^m)$ is a set of attributes – hashtags, named entity lexicalisation:type tuples, URLs, username mentions, event surface forms, and locations. r_{id} is the message ID.[2] z_{stamp} is the creation timestamp of the message, converted to epoch time (i.e. UTC). The set R^m contains the IDs of related messages mentioned in this message's metadata. $\vec{h}$ is a vector embedding of the main event word in the message, defined as the head event of the first sentence. Lastly, H is a minhash of the message text as bytestring (128 permutations).

Two classes of entities are extracted: traditional named entities, and spatio-temporal entities. Named entities are extracted through chunking, using the entity_recognition Python module (Derczynski et al., 2015), using CRF, orthographic features and Brown clusters.

For named entity recognition, we train using the (Ritter et al., 2011) twitter dataset. To improve the ability to cope with the increased variation-caused vocabulary in social media documents, we induced Brown clusters (Brown et al., 1992) over a mixture of newswire and twitter texts.[3] Data scarcity is a problem in streaming situations where drift increases the variety of entity surface forms over time, though this drift-induced scarcity will not be highlighted well when evaluating against static collections. This named entity system

[1] Using the Python `datasketch` implementation.

[2] Typically `id_str` for tweets.

[3] Parameters: 64M tokens from RCV1 cleaned as per (Liang, 2005a) with 64M English tweets from the garden hose; c=6000, min-occur=1.

Task	Precision	Recall	F1
Event	68.55	69.29	68.92
Timex	59.57	52.83	56.00
Location	81.25	64.36	71.82
Spatial entity	48.15	18.06	26.26

Table 2.1: Spatio-temporal entity recognition in tweets

recognizes person, location and organization entities.

Spatial objects are extracted again using the same toolkit and configuration, trained on both an ISO-Space corpus (Pustejovsky et al., 2011) and also the WNUT 2015 data (Baldwin et al., 2015) with geo-loc and facility annotations mapped to location. Locations found from NER are merged into these spatial objects, and this union forms the final set of location mentions found in a given document.

Temporal entities extracted are events and temporal expressions, following definitions from ISO-TimeML. To find these, we train an NER system on TimeBank 1.2 (Pustejovsky et al., 2003), a TimeML corpus (again, `entity_recognition` using cross-genre Brown clusters). This enables extraction of TimeML events, with the blended corpus for word cluster induction intended to bridge the domain gap between newswire in TimeBank and social media data. Critically, we extract only a subset of these, ignoring *state* and *i_state*-type events as they are intuitively less strong matches to the kind of event put forward in a cognitive event clustering ontology (Section 2.1.1).

We also created a spatio-temporally annotated corpus over 400 tweets, labelling events, temporal expressions and ISO-Space locations. The source data was taken from an event-centric published twitter dataset (Zubiaga et al., 2016). The resulting dataset contained 605 events, 122 timexes, 139 spatial entities and 223 locations. These were combined with existing gold-standard datasets. For the temporal entity annotation, the TempEval-2 data (Verhagen et al., 2010) was mixed in. For the spatial entity annotation, the W-NUT data (Baldwin et al., 2015) was added, mapping facility to spatial entity and geo-loc to location. Data was split 80/20 for training and evaluation. Finally, to take into account the mutually exclusive nature of spatio-temporal bisemy (where words often have a spatial and temporal sense, but cannot present both at the same time), we add a feature to the extraction system for spatial and temporal entities. We add a feature for each signal word encountered, describing the chance such terms have of taking a spatial or temporal sense, based on word lists from external lexica. For example, 69.4% of occurrences of the word "until" had a temporal sense in this corpus, so when this word is found, a feature *tempsignal=0.694* is added. This extends the technology developed earlier in PHEME and reported in D2.3.

Results are given in Table 2.1.4. These are generally good, though the spatial entities are hard to recognise. This is the same in ISO-Space recognition on newswire in large corpora, and attributable to the subjective annotation of this type: a spatial entity is defined

in ISO-Space as being an object that participates in a spatial relation.[4] Therefore, spatial entities are very hard to consistently detect without also finding spatial relations in a text, something that is beyond the scope of this work. Nevertheless, precision is not disastrous at 48.15, so spurious detections at scale are likely to comprise background noise.

Unlike A in the cluster representation, $a_i^m \in A^m$ is a set, indicating only whether or not a value appears in the message, and not the number of times. This means repetition has no effect, so the two messages *"#nlproc"* and *"#nlproc #nlproc #nlproc"* are equivalent in this regard.

Note that our temporal processing, while more sophisticated than in prior message clustering technicals and operating at both text and metadata levels, is still crude: we do not attempt to connect events to timelines, or account for tense or for reference time (Reichenbach, 1947). Instead, we assume that events discussed relate to the present. This is true for some social media messages, but not all; studies show that e.g. Twitter users select no single temporal reference style (Hu et al., 2013).

Additionally, we extract username mentions, hashtags, and expanded versions of the URLs mentioned. These both can provide commonalities between messages, bringing in the same users, topics and public conversations, or by referring to the same sources. To capture conversational context, we also extract the IDs of any other documents referred to by this one – e.g. messages that are quoted, referred to in a picture, replied to by this, or the source of a retweet.

Text pre-processing is performed by earlier stages in the PHEME pipeline. Specifically, this includes language ID and tokenisation. Both are described in D2.2.

2.1.5 Features

For a given message:cluster pair, feature tuple $X = \langle Z^x, A^x, o, r, \vec{s} \rangle$ is calculated based on candidate message m_c and cluster k_c. Times $Z = (z_{age}, z_{recency})$ represent the time between the message's creation and cluster inception / cluster's most recent addition respectively. These periods are normalized to the range [-1 .. 1] where 1 represents now and -1 a day (or more) ago, scaled linearly. Attributes $a^x \in A$ are drawn from entity lexicalization:type tuples, event words, hashtags, URLs, locations and username mentions. Each attribute is a tuple of coverage and unity $a = \langle c, u \rangle$. Coverage is the proportion of items appearing in the message that have already been seen in messages assigned to the cluster. As the cluster is likely to contain a wealth of seen items as it ages, this value is likely to decrease. We therefore scale c to increase the reward at low levels, formulating it thus:

[4] For example, in the sentences *"The house is in Tokyo. It is a good house."*, the first *house* is a spatial entity as it interacts spatially with *Tokyo*, but *house* in the second sentence is not a spatial entity, as it has no spatial interaction. Nevertheless, ISO-Space offers the breadth of definition best aligned with our approach.

$$c = \left(\frac{|a^m \cap a|}{|a|} \right)^{l_c} \tag{2.1}$$

This gives a c a range of 0.0 .. 0.1. Coefficient l_c is set to 0.6 for good general results.

Unity represents how tightly bound the message is to the cluster by reflecting what proportion of the items in this attribute, expressed in the message, are found in the cluster. Messages that for example contain only hashtags coming from a candidate cluster, with have a high unity value for hashtags. The initial unity v is determined as the number of sightings of that value in the cluster (capturing common objects is more valuable), divided by the cluster size, all divided by the number v's attribute type (i.e. hashtag, event) is seen in the message. Thus, given as above representations of cluster k (having size n) and message m, and where i is an attribute type, and a_{ij} is the number of times value j occurs in cluster attribute a having type i:

$$v = \frac{\sum_{j \in a_i} \frac{|a_{ij}|}{n}}{|a_i^m|} \tag{2.2}$$

The goal is to give higher scores to messages that present items common in a cluster as their main features of that item type. For example, a cluster where *Stockholm* is a frequently mentioned location would achieve a good unity score with a short message that mentioned *Stockholm* twice and no other locations.

This generates potentially large values, depending on the quality of the match, and needs to be scaled. Based on development experiments, the maximum useful value for v (i.e. v_{max}) is around 3, and a concave responsiveness function helps. The following scaling is applied to derive final unity u:

$$u_{unclipped} = \frac{\min_{v, v_{max}}^{l_v}}{v_{scale}} \tag{2.3}$$

$$u = \begin{cases} u_{unclipped}, & \text{if } u_{unclipped} < 1 \\ 1, & \text{otherwise} \end{cases} \tag{2.4}$$

Where coefficient $l_v = log_{v_{max}} v_{scale}$. Testing revealed v_{scale} to give good results in most cases.

Unity u has a range of 0 .. 1, and so u and c are scaled linearly to the range -1 .. 1 so they align with other features.

o is the number of in-cluster messages in the cluser LSH matching the messages's minhash, with threshold t_{lsh}, normalized by cluster size n and square rooted to give low values boosted effect. The square root is there to scale the feature so low values have slightly impact (values are from 0 to 1). r is a signal indicating related messages common to both candidate message and cluster, of value either -1 or 1.

Lastly, $\vec{s}$ represents event similarity. It is the arithmetic difference of the cluster most-common event word embedding and the message head event embedding, $\vec{s} = \vec{e} - \vec{h}$. This is intended to help capture documents describing similar events using different surface forms. This approach is available because ISO-TimeML events are always a single term, making their representations easy to find. The metric does not address the lack of a guarantee of equal scale throughout vector space[5], but we find it a sufficient proxy for event relatedness. We use GloVe embeddings (Pennington et al., 2014), with 25 dimensions, for this purpose.

In total, this gives a feature representation of how well a message compares with a cluster, based on intermediate representations of the message and the cluster. This keeps feature representations for the matching function generic, avoiding message-specific information such as particular words or times; rather, (message,cluster) comparisons are presented as a degree of overlap along the various dimensions described above. Note that the final feature vector is language-independent, easing language adaptation of the system. The generic representation is fed into our model as a vector of real values.

2.1.6 Classifier

The classifier is a deep neural net, implemented using Keras (Chollet, 2015). The first layer takes inputs from feature representations, with tanh activation. This is followed by three layers each of 80 units; two LeakyReLU layers with $k = 0.01$ (Maas et al., 2013), and a tanh layer with dropout ($\delta = 0.05$) (Hinton et al., 2012). The output layer is a single sigmoid-activation node. All layers except input and output use fan_in Gaussian initialization (He et al., 2015),

The network is trained with `adam` optimization using binary cross-entropy as the loss function. Training data is generated by building representations of the training data clusters with each individual message held out, giving k examples (where k is the number of clusters) for each gold-standard example – save for singleton messages. This generates one positive and $k-1$ negative examples for each input message, which leads to high class balance in the training data to negative matches. Such a balance is typically reflected in trained classifier performance, and so to reduce it, negative examples are randomly downsampled to 1% of the generated amount in the used training data.

This system provides the matching function f_{match}, giving a value in the range 0..1 for a given (message,cluster) pair. Threshold t_{match} gives the position of a gate for converting this output to match / no-match.

[5] I.e. that a certain *absolute* distance in different regions might not mean the same *semantic* distance.

Approach	C	H	V	NMI	ARI
LSH	0.606	0.723	0.658	0.661	0.021
oHDP	0.392	0.271	0.278	0.291	0.014
ES	**0.960**	**0.792**	**0.867**	**0.871**	**0.536**

Table 2.2: Clustering performance – Rumour dataset

2.1.7 Evaluation

We compare our system against two other algorithms, the classic LSH and also the recent high-performer oHDP (Wang et al., 2011; Srijith et al., 2016). These form our baselines.

Intrinsic Evaluation

We evaluate ES clustering with multiple metrics.

H, C, V-measure – From (Rosenberg and Hirschberg, 2007), V-measure draws on the concepts of Homogeneity and Completeness to provide a metric better suited to clustering and less dataset-dependent than F1.

Adjusted Rand Index – A classic clustering evaluation metrics, the Rand Index (RI) is a counting-pairs measure of the proportion of correct assignments (Rand, 1971).

Normalised Mutual Information – An information-theoretic evaluation measure, normalised for cluster count to allow fair evaluation of clusterings where different numbers of clusters are generated.

Results for the Pheme rumour dataset are in Table 2.2. The task here is to group together reports of claims and the discourse around them. We can see that for clustering-specific evaluation measures, ES clustering reaches high performance when applied to this streaming social media scenario, with a V-measure over 0.86. For comparison, prior approaches developed within Pheme reached good scores, but we were still able to improve on these for a better system result.

2.2 Entailment and Contradiction Detection

The utilization of social media material in journalistic workflows is increasing, demanding automated methods for the identification of mis- and disinformation. Since textual contradiction across social media posts can be a signal of rumorousness, we seek to model how claims in Twitter posts are being textually contradicted. We identify two different

contexts in which contradiction emerges: its broader form can be observed across independently posted tweets and its more specific form in threaded conversations. We define how the two scenarios differ in terms of central elements of argumentation: claims and conversation structure. We design and evaluate models for the two scenarios uniformly as 3-way Recognizing Textual Entailment tasks in order to represent claims and conversation structure implicitly in a generic inference model, while previous studies used explicit or no representation of these properties. To address noisy text, our classifiers use simple similarity features derived from the string and part-of-speech level. Corpus statistics reveal distribution differences for these features in contradictory as opposed to non-contradictory tweet relations, and the classifiers yield state of the art performance.

In this deliverable we focus on the presentation of the evaluation studies. Detailed discussion of our work analyzing and testing the use of Recognizing Textual Entailment for the detection of contradictions is given in D4.2.2 "Algorithms for Detecting Disputed Information: Final Version" (`https://www.pheme.eu/wp-content/uploads/2016/06/D422_final.pdf`) and more recently in (Lendvai and Reichel, 2016), presented at the COLING 2016 Workshop on Extra-Propositional Aspects of Meaning, from which we take most of the description of our evaluation study for this deliverable.

2.2.1 Task Definition

Assigning a veracity judgment to a claim appearing on social media requires complex procedures including reasoning on claims aggregated from multiple microposts, to establish claim veracity status (resolved or not) and veracity value (true or false). Until resolution, a claim circulating on social media platforms is regarded as a rumor (Mendoza et al., 2010). The detection of contradicting and disagreeing microposts supplies important cues to claim veracity processing procedures. These tasks are challenging to automatize not only due to the surface noisiness and conciseness of user generated content. One complicating factor is that claim denial or rejection is linguistically often not explicitly expressed, but appears without classical rejection markers or modality and speculation cues (Morante and Sporleder, 2012). Explicit and implicit contradictions furthermore arise in different contexts: in threaded discussions, but also across independently posted messages; both contexts are exemplified in Figure 2.1 on Twitter data.

Language technology has not yet solved the processing of contradiction-powering phenomena, such as negation (Morante and Blanco, 2012) and stance detection (Mohammad et al., 2016), where stance is defined to express speaker favorability towards an evaluation target, usually an entity or concept[6]. In the veracity computation scenario we can speak of *claim targets* that are above the entity level: targets are entire rumors, such as '11 people died during the Charlie Hebdo attack'. Contradiction and stance detection have so far only marginally been addressed in the veracity context (de Marneffe et al., 2012; Ferreira and Vlachos, 2016; Lukasik et al., 2016).

[6]The approach of PHEMEto stance detection is described in section 2.4 further below

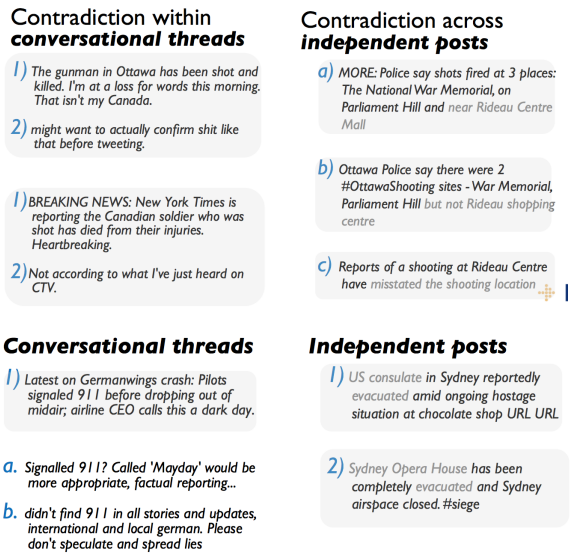

Figure 2.1: Explicit (far left: in threads, left: in independent posts) vs implicit (right: in threads, far right: in independent posts) contradictions in threaded discussions and in independent posts.

T4.2 Detecting Disputed Information: Entailment and Contradictions of PHEMEhas been investigating the advantages of incorporating claim target and conversation context as premises in the Recognizing Textual Entailment (RTE) framework for contradiction detection in rumorous tweets. Our goals are manifold: (a) to offer richer context in contradiction modeling than what would be available on the level of individual tweets, the typical unit of analysis in previous studies; (b) to train and test supervised classifiers for contradiction detection in the RTE inference framework; (c) to address contradiction detection at the level of text similarity only, as opposed to semantic similarity (Xu et al., 2015); (d) to distinguish and focus on two different contradiction relationship types, each involving specific combinations of claim target mention, polarity, and contextual proximity, in particular:

1. **Independent contradictions**: Contradictory relation between independent posts, in which two tweets contain different information about the same claim target that cannot simultaneously hold. The two messages are independently posted, i.e., not occurring within a structured conversation.

2. **Disagreeing replies**: Contradictory relation between a claim-originating tweet and a direct reply to it, whereby the reply expresses disagreement with respect to the claim-introducing tweet.

Contradiction between independently posted tweets typically arises in a broad discourse setting, and may feature larger distance in terms of time, space, and source of

information. The claim target is mentioned in both posts in the contradiction pair, since these posts are uninformed about each other or assume uninformedness of the reader, and thus do not or can not make coreference to their shared claim target. Due to the same reason, the polarity of both posts with respect to the claim can be identical. Texts paired in this type of contradiction resemble those of the recent Interpretable Semantic Similarity shared task (Agirre et al., 2016) that calls to identify five chunk level semantic relation types (equivalence, opposition, specificity, similarity or relatedness) between two texts that originate from headlines or captions. Disagreeing replies are more specific instances of contradiction: contextual proximity is small and trivially identifiable by means of e.g. social media platform metadata, for example the property encoding the tweet ID to which the reply was sent, which in our setup is always a thread-initiating tweet. The claim target is by definition assumed to be contained in the thread-initiating tweet (sometimes termed as claim- or rumor-source tweet). It can be the case that the claim target is not contained in the reply, which can be explained by the proximity and thus shared context of the two posts. The polarity values in source and reply must by definition be different; we refer to this scenario as Disagreeing replies. Importantly, replies may not contain a (counter-)claim on their own but some other form to express disagreement and polarity – for example in terms of speculative language use, or the presence of extra-linguistic cues such as a URL pointing to an online article that holds contradictory content. Such cues are difficult to decode for a machine, and their representation for training automatic classifiers is largely unexplored. Note that we do not make assumptions or restrictions about how the claim target is encoded textually in any of the two scenarios.

In this study, we tackle both contradiction types using a single generic approach: we recast them as three-way RTE tasks on pairs of tweets. The findings of our previous study in which semantic inference systems with sophisticated, corpus-based or manually created syntactico-semantic features were applied to contradiction-labeled data indicate the lack of robust syntactic and semantic analysis for short and noisy texts (see Chapter 3 in the Pheme deliverable D4.2.2). This motivates our current simple text similarity metrics in search of alternative methods for the contradiction processing task.

In Sections 2.2.2 and 2.2.3 present and motivate the collections and the features used for modeling. After the description of method and scores in Section 2.2.4, findings are discussed in Section 2.2.5.

2.2.2 Data

The two datasets corresponding to our two tasks are drawn from a freely available, annotated social media corpus[7] that was collected from the Twitter platform[8] via filtering on event-related keywords and hashtags in the Twitter Streaming API. We worked with

[7] https://figshare.com/articles/PHEME_rumour_scheme_dataset_journalism_use_ case/2068650

[8] twitter.com

event	ENT	CON	UNK	#uniq clms	#uniq tws	ENT	CON	UNK	#uniq clms	#uniq tws
chebdo	143	34	486	36	736	647	427	866	27	199
gwings	39	6	107	13	176	461	257	447	4	29
ottawa	79	37	292	28	465	555	377	168	18	125
ssiege	112	59	456	37	697	332	317	565	21	143
	373	136	1341	114	2074	1995	1378	2046	70	496

Table 2.3: *Threads* (left) and *iPosts* (right) RTE datasets compiled from 4 crisis events: amount of pairs per entailment type (*ENT, CON, UNK*), amount of unique rumorous claims (*#uniq clms*) used for creating the pairs, amount of unique tweets discussing these claims (*#uniq tws*).

English tweets related to four events: the Ottawa shooting[9], the Sydney Siege[10], the Germanwings crash[11], and the Charlie Hebdo shooting[12]. Each event in the corpus was pre-annotated as explained in (Zubiaga et al., 2015b) for several rumorous claims[13] – officially not yet confirmed statements lexicalized by a concise proposition, e.g. "Four cartoonists were killed in the Charlie Hebdo attack" and "French media outlets to be placed under police protection". The corpus collection method was based on a retweet threshold, therefore most tweets originate from authoritative sources using relatively well-formed language, whereas replying tweets often feature non-standard language use.

Tweets are organized into threaded conversations in the corpus and are marked up with respect to stance, certainty, evidentiality, and other veracity-related properties; for full details on released data we refer to (Zubiaga et al., 2015b). The dataset on which we run disagreeing reply detection (henceforth: *Threads*) was converted by us to RTE format based on the threaded conversations labeled in this corpus. We created the Threads RTE dataset drawing on manually pre-assigned Response Type labels by (Zubiaga et al., 2015b) that were meant to characterize source tweet – replying tweet relations in terms of four categories. We mapped these four categories onto three RTE labels: a reply pre-labeled as *Agreed* with respect to its source tweet was mapped to *Entailment*, a reply pre-labeled as *Disagreed* was mapped to *Contradiction*, while replies pre-labeled as *AppealforMoreInfo* and *Comment* were mapped to *Unknown*. Only direct replies to source tweets relating to the same four events as in the independent posts RTE dataset were kept. There are 1,850 tweet pairs in this set; the proportion of contradiction instances amounts to 7%. The *Threads* dataset holds *CON*, *ENT* and *UNK* pairs as exemplified below. Conform the RTE format, pair elements are termed *text* and *hypothesis* – note that directionality between *t* and *h* is assumed as symmetric in our current context so *t* and *h*

[9]https://en.wikipedia.org/wiki/2014_shootings_at_Parliament_Hill,_Ottawa

[10]https://en.wikipedia.org/wiki/2014_Sydney_hostage_crisis

[11]https://en.wikipedia.org/wiki/Germanwings_Flight_9525

[12]https://en.wikipedia.org/wiki/Charlie_Hebdo_shooting

[13]*Rumor*, *rumorous claim* and *claim* are used interchangeably throughout the paper to refer to the same concept.

are assigned based on token-level length.

- **CON** <t>We understand there are two gunmen and up to a dozen hostages inside the cafe under siege at Sydney.. ISIS flags remain on display 7News</t> <h>not ISIS flags</h>
- **ENT** <t>Report: Co-Pilot Locked Out Of Cockpit Before Fatal Plane Crash URL Germanwings URL</t> <h>This sounds like pilot suicide.</h>
- **UNK** <t>BREAKING NEWS: At least 3 shots fired at Ottawa War Memorial. One soldier confirmed shot - URL URL</t> <h>All our domestic military should be armed, now.</h>.

The independently posted tweets dataset (henceforth: *iPosts*) that we used for contradiction detection between independently emerging claim-initiating tweets is described in (Lendvai et al., 2016). This collection is holds 5.4k RTE pairs generated from about 500 English tweets using semi-automatic 3-way RTE labeling, based on semantic or numeric mismatches between the rumorous claims annotated in the data. The proportion of contradictory pairs (*CON*) amounts to 25%. The two collections are quantified in Table 2.3. *iPosts* dataset examples are given below.

- **CON** <t>12 people now known to have died after gunmen stormed the Paris HQ of magazine CharlieHebdo URL URL</t> <h>Awful. 11 shot dead in an assault on a Paris magazine. URL CharlieHebdo URL</h>
- **ENT** <t>SYDNEY ATTACK - Hostages at Sydney cafe - Up to 20 hostages - Up to 2 gunmen - Hostages seen holding ISIS flag DEVELOPING..</t> <h>Up to 20 held hostage in Sydney Lindt Cafe siege URL URL</h>
- **UNK** <t>BREAKING: NSW police have confirmed the siege in Sydney's CBD is now over, a police officer is reportedly among the several injured.</t> <h>Update: Airspace over Sydney has been shut down. Live coverage: URL sydneysiege</h>.

2.2.3 Text similarity features

Data preprocessing on both datasets included screen name and hashtag sign removal and URL masking. Then, for each tweet pair we extracted vocabulary overlap and local text alignment features. The tweets were part-of-speech-tagged using the Balloon toolkit (Reichel, 2012) (PENN tagset, `https://catalog.ldc.upenn.edu/docs/LDC95T7/cl93.html`), normalized to lowercase and stemmed using an adapted version of the Porter stemmer (Porter, 1997). Content words were defined to belong to the set of nouns, verbs, adjectives, adverbs, and numbers, and were identified by their part of speech labels. All punctuation was removed.

Vocabulary overlap

Vocabulary overlap was calculated for content word stem types in terms of the Cosine similarity and the F1 score. The Cosine similarity of two tweets is defined as $C(X,Y) = \frac{|X \cap Y|}{\sqrt{|X| \cdot |Y|}}$, where X and Y denote the sets of content word stems in the tweet pair.

The F1 score is defined as the harmonic mean of precision and recall. Precision and recall here refer to covering the vocabulary X of one tweet by the vocabulary Y of another tweet (or vice versa). It is given by $F1 = 2 \cdot \frac{\frac{|X \cap Y|}{|X|} \cdot \frac{|X \cap Y|}{|Y|}}{\frac{|X \cap Y|}{|X|} + \frac{|X \cap Y|}{|Y|}}$. Again the vocabularies X and Y consist of stemmed content words. Just like the Cosine index, the F1 score is a symmetric similarity metric.

These two metrics are additionally applied to the content word POS label inventories within the tweet pair, which gives the four features *cosine, cosine_pos, f_score*, and *f_score_pos*, respectively.

Local alignment

The amount of stemmed word token overlap was measured by applying local alignment of the token sequences using the Smith-Waterman algorithm (Smith and Waterman, 1981). We chose a score function rewarding zero substitutions by $+1$, and punishing insertions, deletions, and substitutions each by 0-reset. Having filled in the score matrix H, alignment was iteratively applied the following way:

while $\max(H) \geq t$

– trace back from the cell containing this maximum the path leading to it until a zero-cell is reached
– add the substring collected on this way to the set of aligned substrings
– set all traversed cells to 0.

The threshold t defines the required minimum length of aligned substrings. It is set to 1 in this study, thus it supports a complete alignment of any pair of permutations of x. The traversed cells are set to 0 after each iteration step to prevent that one substring would be related to more than one alignment pair. This approach would allow for two restrictions: to prevent cross alignment not just the traversed cells $[i, j]$ but for each of these cells its entire row i and column j needs to be set to 0. Second, if only the longest common substring is of interest, then the iteration is trivially to be stopped after the first step. Since we did not make use of these restrictions, in our case the alignment supports cross-dependencies and can be regarded as an iterative application of a longest common substring match.

From the substring pairs in tweets x and y aligned this way, we extracted two text similarity measures:

- *laProp*: the proportion of locally aligned tokens over both tweets $\frac{m(x)+m(y)}{n(x)+n(y)}$
- *laPropS*: the proportion of aligned tokens in the shorter tweet $\frac{m(\hat{z})}{n(\hat{z})}, \hat{z} = \arg\min_{z \in \{x,y\}}[n(z)]$,

where $n(z)$ denotes the number of all tokens and $m(z)$ the number of aligned tokens in tweet z.

Corpus statistics

Figures 2.2 and 2.3 show the distribution of the features introduced above each for a selected event in both datasets. Each figure half represents a dataset; each subplot shows the distribution of a feature in dependence of the three RTE classes for the selected event in that dataset.

The plots indicate a general trend over all events and datasets: the similarity features reach highest values for the ENT class, followed by CON and UNK. Kruskal-Wallis tests applied separately for all combinations of features, events and datasets confirmed these trends, revealing significant differences for all boxplot triplets ($p < 0.001$ after correction for type 1 errors in this high amount of comparisons using the false discovery rate method of (Benjamini and Yekutieli, 2001)). Dunnett post hoc tests however clarified that for 16 out of 72 comparisons (all POS similarity measures) only UNK but not ENT and CON differ significantly ($\alpha = 0.05$). Both datasets contain the same amount of non-significant cases. Nevertheless, these trends are encouraging to test whether an RTE task can be addressed by string and POS-level similarity features alone, without syntactic or semantic level tweet comparison.

2.2.4 RTE classification experiments for Contradiction and Disagreeing Reply detection

In order to predict the RTE classes based on the features introduced above, we trained two classifiers: Nearest (shrunken) centroids (NC) (Tibshirani et al., 2003) and Random forest (RF) (Statistics and Breiman, 2001), using the R wrapper package *Caret* [14] with the methods *pam* and *rf*, respectively. To derive the same number of instances for all classes, we applied separately for both datasets resampling without replacement, so that the total data amounts about 4,550 feature vectors equally distributed over the three classes, the majority of 4,130 belonging to the iPosts data set. Further, we centered and scaled the

[14] https://cran.r-project.org/web/packages/caret/index.html

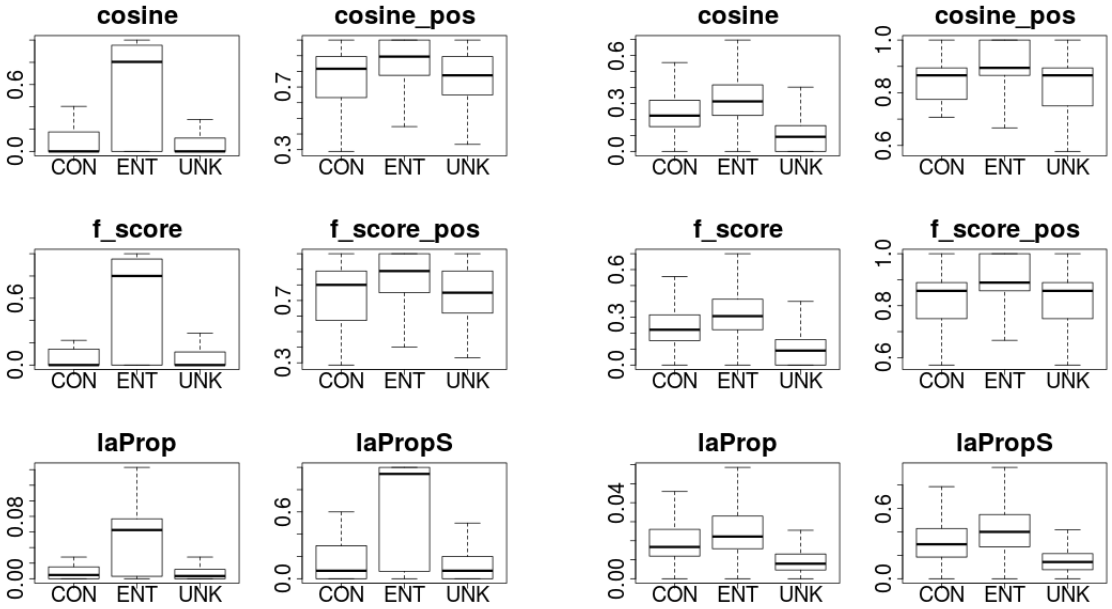

Figure 2.2: Distributions of the similarity metrics by tweet pair class for the event *chebdo* in the *Threads* (**left**) and the *iPosts* dataset (**right).**

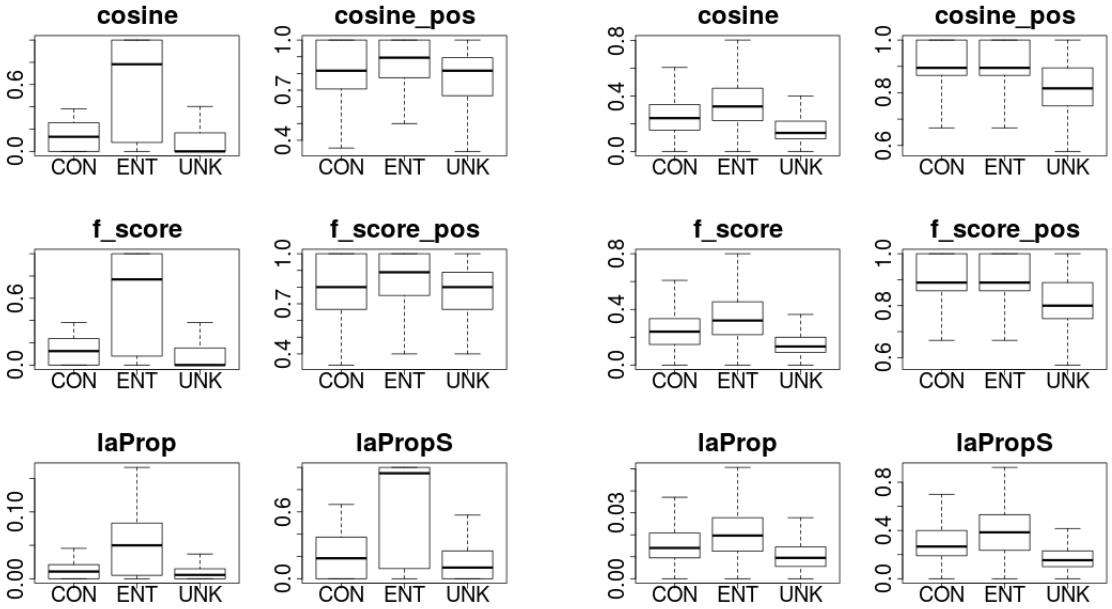

Figure 2.3: Distributions of the similarity metrics by tweet pair class for the event *ssiege* in the *Threads* (**left**) and the *iPosts* dataset (**right).**

feature matrix. Within the Caret framework we optimized the tunable parameters of both classifiers by maximizing the F1 score. This way the NC shrinkage delta was set to 0, which means that the class reference centroids are not modified. For RF the number of variables randomly sampled as candidates at each split was set to 2. The remaining parameters were kept default.

The classifiers were tested on both datasets in a 4-fold event-based held-out setting, training on three events and testing on the remaining one (4-fold cross-validation, CV), quantifying how performance generalizes to new events with unseen claims and unseen targets. The CV scores are summarized in Tables 2.4 and 2.5. It turns out generally that classifying CON is more difficult than classifying ENT or UNK. We observe a dependency of the classifier performances on the two contradiction scenarios: for detecting CON, RF achieved higher classification values on Threads, whereas NC performed better on iPosts. General performance across all three classes was better in independent posts than in conversational threads.

Definitions of contradiction, the genre of texts and the features used are dependent on end applications, making performance comparison nontrivial. On a different subset of the Threads data in terms of events, size of evidence, 4 stance classes and no resampling, (Lukasik et al., 2016) report .40 overall F-score using Gaussian processes, cosine similarity on text vector representation and temporal metadata. Our previous experiments were done using the Excitement Open Platform incorporating syntactico-semantic processing and 4-fold CV. For the non-resampled Threads data we reported .11 F1 on CON via training on iPosts[15]. On the non-resampled iPosts data we obtained .51 overall F1 score (Lendvai et al., 2016), F1 on CON being .25[16] .

	CON	ENT	UNK
F1 (RF/**NC**)	0.33/**0.35**	0.55/0.59	0.51/0.57
precision	0.35/0.40	0.54/0.61	0.54/0.57
recall	0.32/0.34	0.58/0.59	0.56/0.67
accuracy		0.47/0.51	
wgt F1		0.48/**0.51**	
wgt prec.		0.51/0.55	
wgt rec.		0.47/0.51	

Table 2.4: *iPosts* dataset. Mean and weighted (wgt) mean results on held-out data after event held-out cross validation for the Random Forest (RF) and Nearest Centroid (NC) classifiers.

[15]See Pheme Deliverable 4.2.2. https://www.pheme.eu/wp-content/uploads/2016/06/D422_final.pdf

[16]See Pheme Deliverable 4.2.2. https://www.pheme.eu/wp-content/uploads/2016/06/D422_final.pdf

	CON	ENT	UNK
F1 (**RF**/NC)	**0.37**/0.11	0.45/0.50	0.40/0.36
precision	0.42/0.07	0.52/0.56	0.34/0.31
recall	0.35/0.20	0.41/0.47	0.50/0.61
accuracy		0.42/0.39	
wgt F1		**0.43**/0.32	
wgt prec.		0.47/0.33	
wgt rec.		0.42/0.39	

Table 2.5: *Threads* dataset. Mean and weighted (wgt) mean results on held-out data after event held-out cross validation for the Random forest and Nearest Centroid classifiers (RF/NC).

We proposed to model two types of contradictions: in the first both tweets encode the claim target (iPosts), in the second typically only one of them (Threads). The Nearest Centroid algorithm performs poorly on the CON class in Threads where textual overlap is typically small especially for the CON and UNK classes, in part due to the absence of the claim target in replies. However, the Random Forest algorithm's performance is not affected by this factor. The advantage of RF on the Threads data can be explained by its property of training several weak classifiers on parts of the feature vectors only. By this boosting strategy a usually undesirable combination of relatively long feature vectors but few training observations can be tackled, holding for the Threads data that due to its extreme skewedness (cf. Table 2.3) shrunk down to only 420 datapoints after our class balancing technique of resampling without replacement. Results indicate the benefit of RF classifiers in such sparse data cases.

The good performance of NC on the much larger amount of data in iPosts is in line with the corpus statistics reported in section 2.2.3, implying a reasonably small amount of class overlap. The classes are thus relatively well represented by their centroids, which is exploited by the NC classifier. However, as illustrated in Figures 2.2 and 2.3, the majority of feature distributions are generally better separated for ENT and UNK, while CON in its mid position shows more overlap to both other classes and is thus overall a less distinct category.

2.2.5 Conclusions and Future Work

The detection of contradiction and disagreement in microposts supplies important cues to factuality and veracity assessment, and is a central task in computational journalism. We developed classifiers in a uniform, general inference framework that differentiates two tasks based on contextual proximity of the two posts to be assessed, and if the claim target may or may not be omitted in their content. We utilized simple text similarity metrics that proved to be a good basis for contradiction classification.

Text similarity was measured in terms of vocabulary and token sequence overlap. To derive the latter, local alignment turned out to be a valuable tool: as opposed to standard global alignment (Wagner and Fischer, 1974), it can account for crossing dependencies and thus for varying sequential order of information structure in entailing text pairs, e.g. in ”the cat chased the mouse” and ”the mouse was chased by the cat”, which are differently structured into topic and comment. We expect contradictory content to exhibit similar trends in variation with respect to content unit order – especially in the Threads scenario, where entailment inferred from a reply can become the topic of a subsequent replying tweet. Since local alignment can resolve such word order differences, it is able to preserve text similarity of entailing tweet pairs, which is reflected in the relative *laProp* boxplot heights in Figures 2.2 and 2.3.

We have run leave-one-event-out evaluation separately on the independent posts data and on the conversational threads data, which allowed us to compare performances on collections originating from the same genre and platform, but on content where claim targets in the test data are different from the targets in the training data. Our obtained generalization performance over unseen events turns out to be in line with previous reports. Via downsampling, we achieved a balanced performance on both tasks across the three RTE classes; however, in line with previous work, even in this setup the overall performance on contradiction is the lowest, whereas detecting the lack of contradiction can be achieved with much better performance in both contradiction scenarios.

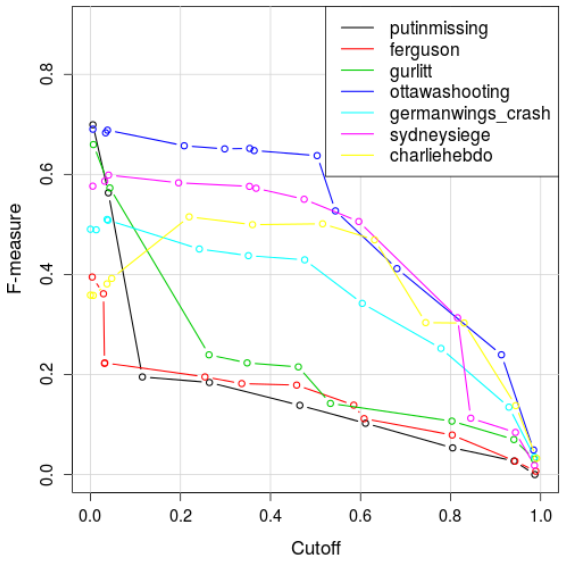

Figure 2.4: Test F1 scores for the rumor classifier, on each .

2.3 Rumour Classification

The rumor classifier was presented in (Lukasik et al., 2015a), Chapter 4 and then improved and extended in (Tolosi et al., 2016). A machine learning model consisting of a decision tree trained on the complete Journalism dataset (Zubiaga et al., 2015a) was developed for the purpose of inferring whether a Twitter thread is a rumor or not. The model is language independent, meaning that it does not use language-specific features and it is context independent, meaning that it does not wait for enough tweets to accumulate on a certain event in order to collect context-based features.

Our evaluation followed a leave-one-topic-out approach, namely we trained on all but one topic (eg. we trained on all but CharlieHebdo tweets) and tested on the left-out topic. The average F1 score over all topics was 0.65, a rather good performance for a language-agnostic and context-agnostic classifier. The performance is comparable to the context-independent baseline reported in (Qazvinian et al., 2011).

Figure 2.4 shows the F1 measure for each of the left-out topics, where on the x-axis we represented the probability cutoff at which non-rumor vs. rumor classification is decided. This probability is the direct output of the decision tree model. We observed that there is a large variation in performance over the different topics, suggesting that the model is domain dependent. We described topic-dependent biases in detail in (Tolosi et al., 2016) and how they influence model performance.

text	position
Birmingham Children's hospital has been attacked. F***ing morons. #UKRiots	support
Girlfriend has just called her ward in Birmingham Children's Hospital & there's no sign of any trouble #Birminghamriots	deny
Birmingham children's hospital guarded by police? Really? Who would target a childrens hospital #disgusting #Birminghamriots	question

Table 2.6: Tweets on a rumour about hospital being attacked during 2011 England Riots.

2.4 Rumour Stance Classification

As detailed in (Lukasik et al., 2015a), we carry out tweet-level stance classification automatically, in order to assist in (near) real-time rumour monitoring by journalists and authorities (Procter et al., 2013). This information is used as an important feature in veracity classification, as detailed in Deliverable D4.3.2.

In this deliverable we evaluate tweet-level stance classification on unseen rumours, based on a training set of other already annotated rumours. The experiments here go beyond those reported in the earlier Deliverable D6.2.1.

We apply Gaussian Processes and multi-task learning methods, following the problem formulation introduced in Lukasik et al. (2015a), which we also describe next.

2.4.1 Problem formulation

Let R be a set of rumours, each of which consists of tweets discussing it, $\forall_{r \in R}\ T_r = \{t^r_1, \cdots, t^r_{r_n}\}$. $T = \cup_{r \in R} T_r$ is the complete set of tweets from all rumours. Each tweet is classified as supporting, denying or questioning with respect to its rumour: $y(t_i) \in \{s, d, q\}$.

We formulate the problem in two different settings. First, we consider the Leave One Out (LOO) setting, which means that for each rumour $r \in R$, we construct the test set equal to T_r and the training set equal to $T \setminus T_r$. This is the most challenging scenario, where the test set contains an entirely unseen rumour.

The second setting is Leave Part Out (LPO). In this formulation, a very small number of initial tweets from the target rumour is added to the training set $\{t^r_1, \cdots, t^r_{r_k}\}$. This scenario becomes applicable typically soon after a rumour breaks out and journalists have started monitoring and analysing the related tweet stream. The experimental section investigates how the number of initial training tweets influences classification performance on a fixed test set, namely: $\{t^r_{r_l}, \cdots, t^r_{r_n}\}$, $l > k$.

The tweet-level stance classification problem here assumes that tweets from the training set are already labelled with the rumour discussed and the attitude expressed towards that. This information can be acquired either via manual annotation as part of expert analysis, as is the case with our dataset, or automatically, e.g. using pattern-based rumour detection Zhao et al. (2015). Our method is then used to classify the stance expressed in each new tweet from the test set.

2.4.2 Classifiers

Gaussian Processes for Classification

Gaussian Processes are a Bayesian non-parametric machine learning framework that has been shown to work well for a range of NLP problems, often beating other state-of-the-art methods Cohn and Specia (2013); Lampos et al. (2014); Beck et al. (2014); Preotiuc-Pietro et al. (2015).

A Gaussian Process defines a prior over functions, which combined with the likelihood of data points gives rise to a posterior over functions explaining the data. The key concept is a kernel function, which specifies how outputs correlate as a function of the input. Thus, from a practitioner's point of view, a key step is to choose an appropriate kernel function capturing the similarities between inputs.

We use Gaussian Processes as this probabilistic kernelised framework avoids the need for expensive cross-validation for hyperparameter selection.[17] Instead, the marginal likelihood of the data can be used for hyperparameter selection.

The central concept of Gaussian Process Classification (GPC; Rasmussen and Williams (2005)) is a latent function f over inputs $\mathbf{x}$: $f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}'))$, where m is the mean function, assumed to be 0 and k is the kernel function, specifying the degree to which the outputs covary as a function of the inputs. We use a linear kernel, $k(\mathbf{x}, \mathbf{x}') = \sigma^2 \mathbf{x}^\top \mathbf{x}'$. The latent function is then mapped by the probit function $\Phi(f)$ into the range $[0, 1]$, such that the resulting value can be interpreted as $p(y = 1|\mathbf{x})$.

The GPC posterior is calculated as

$$p(f^*|X, \mathbf{y}, \mathbf{x}_*) = \int p(f^*|X, \mathbf{x}_*, \mathbf{f}) \frac{p(\mathbf{y}|\mathbf{f})p(\mathbf{f})}{p(\mathbf{y}|X)} d\mathbf{f},$$

where $p(\mathbf{y}|\mathbf{f}) = \prod_{j=1}^{n} \Phi(f_j)^{y_j}(1 - \Phi(f_j))^{1-y_j}$ is the Bernoulli likelihood of class y. After calculating the above posterior from the training data, this is used in prediction, i.e.,

$$p(y_* = 1|X, \mathbf{y}, \mathbf{x}_*) = \int \Phi(f_*)\, p(f_*|X, \mathbf{y}, \mathbf{x}_*)\, df_*\,.$$

[17] There exist frequentist kernel methods, such as SVMs, which additionally require extensive heldout parameter tuning.

The above integrals are intractable and approximation techniques are required to solve them. There exist various methods to deal with calculating the posterior; here we use Expectation Propagation (EP; Minka and Lafferty (2002)). In EP, the posterior is approximated by a fully factorised distribution, where each component is assumed to be an unnormalised Gaussian.

In order to conduct multi-class classification, we perform a one-vs-all classification for each label and then assign the one with the highest likelihood, amongst the three (supporting, denying, questioning). We choose this method due to interpretability of results, similar to recent work on occupational class classification Preotiuc-Pietro et al. (2015).

Intrinsic Coregionalisation Model In the Leave-Part-Out (LPO) setting initial labelled tweets from the target rumour are observed as well, as opposed to the Leave-One-Out (LOO) setting. In the case of LPO, we propose to weigh the importance of tweets from the reference rumours depending on how similar their characteristics are to the tweets from the target rumour available for training. To handle this with GPC, we use a multiple output model based on the Intrinsic Coregionalisation Model (ICM; Álvarez et al. (2012)). This model has already been applied successfully to NLP regression problems Beck et al. (2014) and it can also be applied to classification ones. ICM parametrizes the kernel by a matrix which represents the extent of covariance between pairs of tasks. The complete kernel takes form of

$$k((\mathbf{x}, d), (\mathbf{x}', d')) = k_{data}(\mathbf{x}, \mathbf{x}')B_{d,d'} ,$$

where B is a square coregionalisation matrix, d and d' denote the tasks of the two inputs and k_{data} is a kernel for comparing inputs $\mathbf{x}$ and $\mathbf{x}'$ (here, linear). We parametrize the coregionalisation matrix $B = \boldsymbol{\kappa} I + \boldsymbol{v}\boldsymbol{v}^T$, where $\boldsymbol{v}$ specifies the correlation between tasks and the vector κ controls the extent of task independence. Note that in case of LOO setting this model does not provide useful information, since no target rumour data is available to estimate similarity to other rumours.

Hyperparameter selection We tune hyperparameters $\mathbf{v}$, $\boldsymbol{\kappa}$ and σ^2 by maximizing evidence of the model $p(\mathbf{y}|X)$, thus having no need for a validation set.

Methods We consider GPs in three different settings, varying in what data the model is trained on and what kernel it uses. The first setting (denoted GP) considers only target rumour data for training. The second (GPPooled) additionally considers tweets from reference rumours (i.e. other than the target rumour). The third setting is GPICM, where an ICM kernel is used to weight influence from tweets from reference rumours.

Baselines

To assess and compare the efficiency of Gaussian Processes for rumour stance classification, we also experimented with five more baseline classifiers, all of which were imple-

mented using the scikit Python package (Pedregosa et al., 2011): (1) *majority classifier*, which is a naive classifier that labels all the instances in the test set with the most common class in the training set, (2) *logistic regression* (MaxEnt), (3) *support vector machines* (SVM), (4) *naive bayes* (NB) and (5) *random forest* (RF). The selection of these baselines is in line with the classifiers used in recent research on stance classification (Zeng et al., 2016), who found that random forests, followed by logistic regression, performed best.

2.4.3 Datasets

The PHEME dataset includes tweets associated with the following five events:

- **Ferguson unrest:** Citizens of Ferguson in Michigan, USA, protested after the fatal shooting of an 18-year-old African American, Michael Brown, by a white police officer on August 9, 2014.
- **Ottawa shooting:** Shootings occurred on Ottawa's Parliament Hill in Canada, resulting in the death of a Canadian soldier on October 22, 2014.
- **Sydney siege:** A gunman held as hostages ten customers and eight employees of a Lindt chocolate café located at Martin Place in Sydney, Australia, on December 15, 2014.
- **Charlie Hebdo shooting:** Two brothers forced their way into the offices of the French satirical weekly newspaper Charlie Hebdo in Paris, killing 11 people and wounding 11 more, on January 7, 2015.
- **Germanwings plane crash:** A passenger plane from Barcelona to Düsseldorf crashed in the French Alps on March 24, 2015, killing all passengers and crew on board. The plane was ultimately found to have been deliberately crashed by the co-pilot of the plane.

In this case, we perform 5-fold cross-validation, having four events in the training set and the remaining event in the test set for each fold.

2.4.4 Features

We conducted a series of preprocessing steps in order to address data sparsity. All words were converted to lowercase; stopwords have been removed[18]; all emoticons were replaced by words[19]; and stemming was performed. In addition, multiple occurrences of a character were replaced with a double occurrence (Agarwal et al., 2011), to correct for

[18] We removed stopwords using the English list from Python's NLTK package.

[19] We used the dictionary from: `http://bit.ly/1rX1Hdk` and extended it with: :o, :|, =/, :s, :S, :p.

Table 2.7: Counts of tweets with supporting, denying or questioning labels in each event collection on the PHEME dataset.

Dataset	Rumours	Supporting	Denying	Questioning
Ottawa shooting	58	161	76	63
Ferguson riots	46	192	82	94
Charlie Hebdo	74	235	56	51
Germanwings crash	68	67	12	28
Sydney siege	71	222	89	99
Total	287	877	315	335

misspellings and lengthenings, e.g., *looool*. All punctuation was also removed, except for ., *!* and *?*, which we hypothesize to be important for expressing emotion. Lastly, usernames were removed as they tend to be rumour-specific, i.e., very few users comment on more than one rumour.

After pre-processing the text data, we use either the resulting bag of words (BOW) feature representation and replace all words with their Brown cluster ids (Brown). Brown clustering is a hard hierarchical clustering method (Liang, 2005b; Derczynski and Chester, 2016). It clusters words based on maximizing the probability of the words under the bigram language model, where words are generated based on their clusters. In previous work it has been shown that Brown clusters yield better performance than directly using the BOW features (Lukasik et al., 2015b).

In our experiments, the clusters used were obtained using 1000 clusters acquired from a large scale Twitter corpus (Owoputi et al., 2013), from which we can learn Brown clusters aimed at representing a generalisable Twitter vocabulary. Retweets are removed from the training set to prevent bias (Llewellyn et al., 2014). More details on the Brown clusters that we used as well as the words that are part of each cluster are available online[20].

During the experimentation process, we also tested additional features, including the use of the bag of words instead of the Brown clusters, as well as using word embeddings trained from the training sets (Mikolov et al., 2013). However, results turned out to be substantially poorer than those we obtained with the Brown clusters. We conjecture that this was due to the little data available to train the word embeddings; further exploring use of word embeddings trained from larger training datasets is left future work. In order to focus on our main objective of proving the effectiveness of a multi-task learning approach, as well as for clarity purposes, since the number of approaches to show in the figures increases if we also consider the BOW features, we only show results for the classifiers relying on Brown clusters as features.

[20]http://www.cs.cmu.edu/ ark/TweetNLP/cluster_viewer.html

2.4.5 Evaluation Measures

Accuracy is often deemed a suitable evaluation measure to assess the performance of a classifier on a multi-class classification task. However, the classes are clearly imbalanced in our case, with varying tendencies towards one of the classes in each of the rumours. We argue that in these scenarios the sole evaluation based on accuracy is insufficient, and further measurement is needed to account for category imbalance. This is especially necessary in our case, as a classifier that always predicts the majority class in an imbalanced dataset will achieve high accuracy, even if the classifier is useless in practice. To tackle this, we use both micro-averaged and macro-averaged F1 scores. Note that the micro-averaged F1 score is equivalent to the well-known accuracy measure, while the macro-averaged F1 score complements it by measuring performance assigning the same weight to each category.

Both of the measures rely on precision (Equation 2.5) and recall (Equation 2.6) to compute the final F1 score.

$$\text{Precision}_k = \frac{tp_k}{tp_k + fp_k} \tag{2.5}$$

$$\text{Recall}_k = \frac{tp_k}{tp_k + fn_k} \tag{2.6}$$

where tp_k (true positives) refer to the number of instances correctly classified in class k, fp_k is the number of instances incorrectly classified in class k, and fn_k is the number of instances that actually belong to class k but were not classified as such.

The above equations can be used to compute precision and recall for a specific class. Precision and recall for all the classes in a problem with c classes are computed differently if they are microaveraged (see Equations 2.7 and 2.8) or macroaveraged (see Equations 2.9 and 2.10).

$$\text{Precision}_{\text{micro}} = \frac{\sum_{k=1}^{c} tp_k}{\sum_{k=1}^{c} tp_k + \sum_{k=1}^{c} fp_k} \tag{2.7}$$

$$\text{Recall}_{\text{micro}} = \frac{\sum_{k=1}^{c} tp_k}{\sum_{k=1}^{c} tp_k + \sum_{k=1}^{c} fn_k} \tag{2.8}$$

$$\text{Precision}_{\text{macro}} = \frac{\sum_{k=1}^{c} \text{Precision}_k}{c} \tag{2.9}$$

$$\text{Recall}_{\text{macro}} = \frac{\sum_{k=1}^{c} \text{Recall}_k}{c} \tag{2.10}$$

After computing microaveraged and macroveraged precision and recall, the final F1 score is computed in the same way, i.e., calculating the harmonic mean of the precision and recall in question (see Equation 2.11).

$$\text{F1} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{2.11}$$

After computing the F1 score for each fold, we compute the micro-averaged score across folds.

2.4.6 Results

First, we look at the results on each dataset separately. Then we complement the analysis by aggregating the results from both datasets, which leads to further understanding the performance of our classifiers on rumour stance classification.

Comparison of Classifiers

We show the results for the LOO and LPO settings in the same figure, distinguished by the training size displayed in the X axis. In all the cases, labelled tweets from the remainder of the rumours (rumours other than the test/targer rumour) are used for training, and hence the training size shown in the X axis is in addition to those. Note that the training size refers to the number of labelled instances that the classifier is making use of from the target rumour. Thus, a training size of 0 indicates the LOO setting, while training sizes from 10 to 50 pertain to the LPO setting.

Figure 2.5 and Table 2.8 show how micro-averaged and macro-averaged F1 scores for the England riots dataset change as the number of tweets from the target rumour used for training increases. We observe that, as initially expected, the performance of most of the methods improves as the number of labelled training instances from the target rumour increases. This increase is especially remarkable with the GP-ICM method, which gradually increases after having as few as 10 training instances. GP-ICM's performance keeps improving as the number of training instances approaches 50[21] Two aspects stand out from analysing GP-ICM's performance:

- It performs poorly in terms of micro-averaged F1 when no labelled instances from the target rumour are used. However, it makes very effective use of the labelled training instances, overtaking the rest of the approaches and achieving the best results. This proves the ability of GP-ICM to make the most of the labelled instances from the target rumour, which the rest of the approaches struggle with.

[21] Note that 50 tweets represent, on average, less than 7% of the whole rumour, with the rest of the rumour yet to be observed.

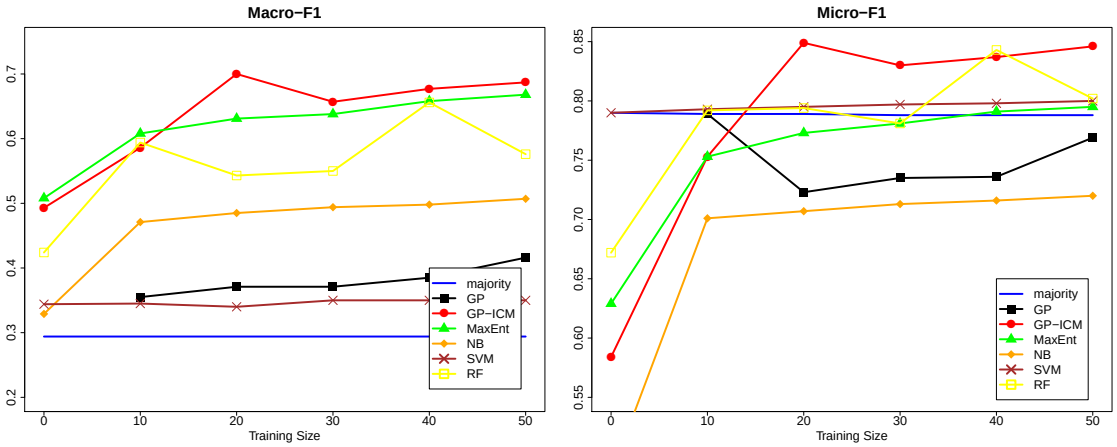

Figure 2.5: Micro-F1 and Macro-F1 scores for different methods versus the size of the target rumour used for training in the LPO setting on the England riots dataset. The test set is fixed to all but the first 50 tweets of the target rumour.

- Irrespective of the number of labelled instances, GP-ICM is robust when evaluated in terms of macro-averaged F1. This means that GP-ICM is managing to determine the distribution of classes effectively, assigning labels to instances in the test set in a way that is better distributed than the rest of the classifier.

Despite the saliency of GP-ICM, we notice that two other baseline approaches, namely MaxEnt and RF, achieve competitive results that are above the rest of the baselines, but still perform worse than GP-ICM.

Table 2.8: Micro-F1 and Macro-F1 scores for different methods on the England riots dataset.

	Macro-F1						Micro-F1					
	0	10	20	30	40	50	0	10	20	30	40	
Majority	0.294	0.294	0.294	0.294	0.294	0.294	**0.79**	**0.789**	0.789	0.788	0.788	0
GP		0.355	0.371	0.371	0.385	0.416		**0.789**	0.723	0.735	0.736	0
GP-ICM	0.493	0.586	**0.7**	**0.657**	**0.677**	**0.687**	0.584	0.753	**0.849**	**0.83**	0.837	**0**
MaxEnt	**0.508**	**0.608**	0.631	0.638	0.658	0.668	0.629	0.753	0.773	0.781	0.791	0
NB	0.329	0.471	0.485	0.494	0.498	0.507	0.485	0.701	0.707	0.713	0.716	
SVM	0.344	0.345	0.34	0.35	0.35	0.35	0.79	0.793	0.795	0.797	0.798	
RF	0.424	0.594	0.543	0.55	0.656	0.576	0.672	0.792	0.794	0.781	**0.843**	0

The results from the PHEME dataset are shown in Figure 2.6 and Table 2.9. Overall, we can observe that results are lower in this case than they were for the riots dataset.

The reason for this can be attributed to the following two observations: on the one hand, each fold pertaining to a different event in the PHEME dataset means that the classifier encounters a new event in the classification, where it will likely find new vocabulary, which may be more difficult to classify; on the other hand, the PHEME dataset is more prominently composed of tweets that are replying to others, which are likely shorter and less descriptive on their own and hence more difficult to get meaningful features from. Despite the additional difficulty in this dataset, we are interested in exploring if the same trend holds across classifiers, from which we can generalise the analysis to different types of classifiers.

One striking difference with respect to the results from the riots dataset is that, in this case, the classifiers, including GP-ICM, are not gaining as much from the inclusion of labelled instances from the target rumour. This is likely due to the heterogeneity of each of the events in the PHEME dataset. Here a diverse set of rumourous newsworthy pieces of information are discussed pertaining to the selected events as they unfold. By contrast, each rumour in the riots dataset is more homogeneous, as each rumour focuses on a specific story.

Interestingly, when we compare the performance of different classifiers, we observe that GP-ICM again outperforms the rest of the approaches, both in terms of micro-averaged and macro-averaged F1 scores. While the micro-averaged F1 score does not increase as the number of training instances increases, we can see a slight improvement in terms of macro-averaged F1. This improvement suggests that GP-ICM does still take advantage of the labelled training instances to boost performance, in this case by better distributing the predicted labels.

Again, as we observed in the case of the riots dataset, two baselines stand out, MaxEnt and RF. They are very close to the performance of GP-ICM for the PHEME dataset, event outperforming it in a few occasions. In the following subsection we take a closer look at the differences among the three classifiers.

Table 2.9: Micro-F1 and Macro-F1 scores for different methods on the PHEME dataset.

	Macro-F1						Micro-F1					
	0	10	20	30	40	50	0	10	20	30	40	
Majority	0.243	0.242	0.241	0.24	0.24	0.24	0.574	0.569	0.566	0.562	0.561	0
GP		0.436	0.494	0.497	0.52	0.515		0.555	0.587	0.617	0.612	0
GP-ICM	0.576	0.572	**0.607**	**0.584**	**0.601**	**0.597**	**0.679**	0.655	**0.669**	0.664	**0.675**	**0**
MaxEnt	**0.577**	**0.576**	0.583	0.583	0.579	0.577	0.66	0.655	0.657	0.654	0.653	
NB	0.355	0.352	0.345	0.345	0.345	0.343	0.371	0.373	0.366	0.363	0.363	0
SVM	0.249	0.25	0.249	0.25	0.246	0.247	0.568	0.563	0.561	0.556	0.555	0
RF	0.556	0.578	0.579	0.58	0.579	0.577	0.657	**0.668**	0.667	**0.667**	0.668	0

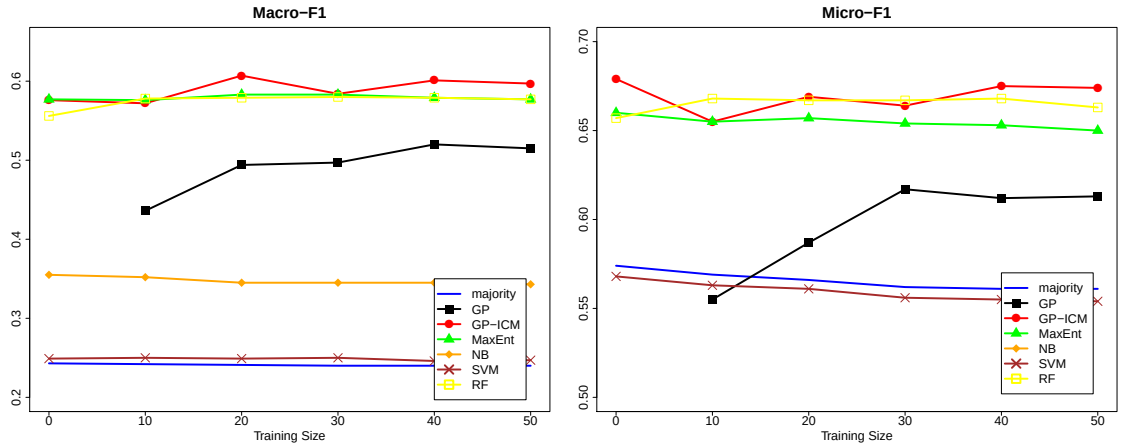

Figure 2.6: Micro-F1 and Macro-F1 scores for different methods versus the size of the target rumour used for training in the LPO setting on the PHEME dataset. The test set is fixed to all but the first 50 tweets of the target rumour.

Analysing the Performance of the Best-Performing Classifiers

We delve into the results of the best-performing classifiers, namely GP-ICM, MaxEnt and RF, looking at their per-class performance. This will help us understand when they perform well and where it is that GP-ICM stands out achieving the best results.

Tables 2.10 and 2.11 show per-class F1 measures for the aforementioned three best-performing classifiers for the England riots dataset and the PHEME dataset, respectively. They also show statistics of the mis-classifications that the classifiers made, in the form of percentage of deviations towards the other classes.

Looking at the per-class performance analysis, we observe that the performance of GP-ICM varies when we look into Precision and Recall. Still, in all the dataset-class pairs, GP-ICM performs best in terms of either Precision or Recall, even though never in both. Moreover, it is generally the best in terms of F1, achieving the best Precision and Recall. The only exception is with MaxEnt classifying questioning tweets more accurately in terms of F1 for the England riots.

When we look at the deviations, we see that all the classifiers suffer from the datasets being imbalanced towards supporting tweets. This results in all classifiers classifying numerous instances as supporting, while they are actually denying or questioning. This is a known problem in rumour diffusion, as previous studies have found that people barely deny or question rumours but generally tend to support them irrespective of their actual veracity value (Zubiaga et al., 2016). While we have found that GP-ICM can tackle the imbalance issue quite effectively and better than other classifiers, this caveat posits the need for further research in dealing with the striking majority of supporting tweets in the context of rumours in social media.

Table 2.10: Per-class precision, recall and F1 scores for the best-performing classifiers on the England riots dataset.

Class	Classifier	Performance			Deviations		
		P	R	F1	S	D	Q
supporting (S)	GP-ICM	0.887	**0.931**	**0.909**	—	0.86%	6.03%
	MaxEnt	**0.926**	0.818	0.869	—	11.60%	6.56%
	RF	0.859	0.914	0.885	—	1.21%	7.41%
denying (D)	GP-ICM	**0.862**	0.483	**0.619**	49.52%	—	2.22%
	MaxEnt	0.500	**0.739**	0.597	22.69%	—	3.41%
	RF	0.696	0.311	0.430	67.51%	—	1.43%
questioning (Q)	GP-ICM	**0.478**	0.608	0.535	34.28%	4.92%	—
	MaxEnt	0.461	**0.647**	**0.538**	29.48%	5.78%	—
	RF	0.367	0.472	0.413	42.00%	10.79%	—

2.4.7 Discussion

Experimentation with two different approaches based on Gaussian Processes (GP and GP-ICM) and comparison with respect to a set of competitive baselines over two rumour datasets enables us to gain generalisable insight on rumour stance classification on Twitter. This is reinforced by the fact that the two datasets are very different from each other. The first dataset, collected during the England riots in 2011, is a single event that we have split into folds, each fold belonging to a separate rumour within the event; hence, all the rumours are part of the same event. The second dataset, collected within the PHEME project, includes tweets for a set of five newsworthy events, where each event has been assigned a separate fold; therefore, the classifier needs to learn from four events and test on a new, unknown event, which has proven more challenging.

Results are generally consistent across datasets, which enables us to generalise conclusions well. We observe that while GP itself does not suffice to achieve competitive results, GP-ICM does instead help boost the performance of the classifier substantially to even outperform the rest of the baselines in the majority of the cases.

GP-ICM has proven to consistently perform well in both datasets, despite their very different characteristics, being competitive not only in terms of micro-averaged F1, but also in terms of macro-averaged F1. GP-ICM manages to balance the varying class distributions effectively, showing that its performance is above the rest of the baselines in accurately determining the distribution of classes. This is very important in this task of rumour stance classification, owing to the fact that even if a classifier that is 100% accurate is unlikely, a classifier that accurately guesses the overall distribution of classes can be of great help. If a classifier makes a good estimation of the number of denials in an

Table 2.11: Per-class precision, recall and F1 scores for the best-performing classifiers on the PHEME dataset.

Class	Classifier	Performance			Deviations		
		P	R	F1	S	D	Q
supporting (S)	GP-ICM	**0.731**	0.825	**0.776**	—	7.12%	10.34%
	MaxEnt	0.715	0.814	0.761	—	12.08%	6.55%
	RF	0.696	**0.860**	0.770	—	7.42%	6.55%
denying (D)	GP-ICM	**0.540**	0.313	**0.396**	50.36%	—	18.35%
	MaxEnt	0.427	**0.325**	0.369	50.54%	—	16.97%
	RF	0.494	0.285	0.362	58.48%	—	13.00%
questioning (Q)	GP-ICM	0.594	**0.647**	**0.619**	27.21%	8.13%	—
	MaxEnt	0.635	0.569	0.600	29.54%	13.52%	—
	RF	**0.657**	0.552	0.600	34.16%	10.68%	—

aggregated set of tweets, it can be useful to flag those potentially false rumours with high level of confidence.

Another factor that stands out from GP-ICM is its capacity to perform well when a few labelled instances of the target rumour are leveraged in the training phase. GP-ICM effectively exploits the knowledge garnered from the few instances from the target rumour, outperforming the rest of the baselines even when its performance was modest when no labelled instances were used from the target rumour.

In light of these results, we deem GP-ICM the most competitive approach to use when one can afford to get a few instances labelled from the target rumour. The labels from the target rumour can be obtained in practice in different ways: (1) having someone in-house (e.g. journalists monitoring breaking news stories) label a few instances prior to running the classifier, (2) making use of resources for human computation such as crowdsourcing platforms to outsource the labelling work, or (3) developing techniques that will attempt to classify the first few instances, incorporating in the training set those for which a classification with high level of confidence has been produced. The latter presents an ambitious avenue for future work that could help alleviate the labelling task.

On the other hand, in the absence of labelled data from the target rumour, which is the case of the LOO setting, the effectiveness of the GP-ICM classifier is not as prominent. For this scenario, other classifiers such as MaxEnt and Random Forests have proven more competitive and one could see them as better options. However, we do believe that the remarkable difference that the reliance on the LPO setting produces is worth exploiting where possible.

in domain tweets (in %)	Ottawa shooting	ferguson riots	Charlie Hebdo	Germanwings	Sydney siege	macro mean (Acc)
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.

2.5 Rumour Veracity Classification

Unlike the previous stance classification task, rumour veracity classification aims to verify whether the rumour is likely to be true or false. In the deliverable D4.3.2 we described our approach to veracity classification in detail. Here in this section we aim to provide a summary of our approach and its results but also include our last activities which differ from what has been reported in D4.3.2.

In D4.3.2 we extracted 35 feature types such as bag of words, brown clusters, regular expressions, sentiment, etc. The features have been fed to the learning algorithm to do the classification.

We investigated various classification methods including Support Vector Machines (SVM), J48 Tree, Bayes, Random Forest and Instance Based Learning. Our experiments have been performed on the PHEME rumour data set. On this data set the best performing system is the Instance Based Learning approach (89% accuracy) and the least one the SVM classifier (71%). Thus we reported detailed results using the Instance Based Learning method. As summary of these results are shown in Table 2.12.

From the results we can see that the Instance Based Learning (IBk) method achieves around 89% to 92% accuracy. The first figure is obtained using training data that comprises all the rumours but the testing one whereas the second figure is obtained using all the rumours plus 60% of the testing instances as the training data. Accuracy may be biased when the data is unbalanced in terms of positive (true) and negative (false) instances. Thus we also performed F1-measure. For both 0% and 60% settings the F1-measure results are around 90% and 93% respectively.

These experiments have been performed (1) on tweet level and (2) allowing some data from the same event within the training process. The point (1) means that each tweet is treated indepently and its veracity is assessed in isolation without taking into its rumour

context. Certainly this is a limitation and thus we aimed to adress this. Second point (2) involves more making the classifier not to see anything that is related to the same evant. Note an event can have several rumours. In the previous setting we regarded rumours as isolated units and trained on n-1 rumours and tested on the nth rumour. Now we train on rumours of n-1 events and test on rumours of the nth event.

The current devolopments thus invovle: a rumour level veracity classification adressing (1) and training and testing on rumours of an entire event. In the following we give more details about the recent activities.

2.5.1 Feature Extraction

In the deliverable D4.3.2 we described 35 different feature types. These features are extracted for each tweet, i.e. tweet level. For rumour level classification we continue performing the tweet level feature extraction using the same feature types. However, unlike previously we merge the feature values of all features accross all tweets[22] within the same rumour to form rumour level feature extraction. Each feature is treated as numeric and for combining the different values of each feature we perform a summation over all the values.

2.5.2 Results and Packaging

Similar to the experiments performed in the deliverable D4.3.2 we continued using Instance Based Learning for classification purpose. The accuracy result on the same data used in D4.3.2 is 73.72%. On top of this data we added more data[23] and re-run the experiments. First we added around 75 rumours/non-rumours for each event. This results in 75.7% accuracy. We continued enriching the data to include 100 rumours/non-rumours. This resulted in 77.09% accuracy. Interestingly adding more data did not add further value to the results. Same was true when less than 75 rumours/non-rumours were added into the training data.

We packaged the new classifier as GATE plugin. The plugin is aware of the dynamic nature of incoming tweets. In the pipeline each incoming tweet is clustered based on its rumour. The veracity classifier listens to these new incoming tweets and extracts from each new tweet its rumour information. Once it knows the rumour id of the new tweet it checks whether there have been any previous tweets about that rumour. If yes it merges the new tweet with those previous tweets and performs feature value merging. Based on the current merged feature values the classifier performs veracity classification on the new tweet. Note that the veracity class of the new tweet might be different from what has been

[22] Actually we obtained best performances when we disregarded the first 20% of the tweets and performed the feature merging on the recent 80% of tweets.

[23] https://figshare.com/articles/PHEME_dataset_of_rumours_and_non-rumours/4010619

said about the ealier tweets within the same rumour. It is important to treat the rumour class information of the new/last tweet as the rumour class for the entire rumour. It is basically an up-date.

Chapter 3

Integration Evaluation

3.1 Introduction

The PHEME Integrated Framework is meant to scale. In order to ensure the scalability of the solution the project is on the one hand improving the performance and scalability of the different individual components as reported in previous sections. On the other hand, from the integration perspective, the project is following a global integration strategy to ensure the performance and scalability of the overall system. This global integration strategy presents project-wide approaches orthogonal to the individual scaling plans and common for most technical components. The focus is on integration aspects to provide an infrastructure to integrate components while enabling big data scaling techniques, such as scaling up, scaling out, parallelisation, etc. This global integration strategy takes into account limits of individual components to align them into a common plan.

From the integration and scalability perspectives, pipelines should be able to increase the throughput and decrease the latency as much as possible. In order to do that, enabling parallelisation means processing of several inputs coming from components in a pipeline with other identical components that work in parallel. In an optimal scenario, it is simply adding more processing units for the same components that work slower compared to other components in a pipeline. More processing units can be provided to components by scaling horizontally or vertically. Scaling horizontally is achieved by adding more nodes (computers) to a system, while vertical scaling can be achieved by running the whole pipeline on a faster machine.

Within PHEME project partners implemented a big data IT Infrastructure using a physical Infrastructure and a virtualization layer in Sheffield premises. However, nothing precludes to install PHEME in other cloud-based environment or scale-up or scale-ot the system. The Sheffield setting comprises four different servers, one acting as head node and two acting as working nodes. This infrastructure provides enough flexibility to be virtualised and hosts the main components and the underlying software infrastructure required for integration. This installation is used to showcase the project results and it is

where the bulk of the work on integration has been done.

This big data infrastructure allows enough flexibility to achieve the necessary scalability. For instance, new nodes could be added and the workload of the different components and pipelines could be parallelised to improve the response time and avoid bottlenecks if necessary. This has been tested in some particular scenarios. This means that although the project has a limited budget in terms of infrastructure, nothing prevents achieving better throughput in a potential commercial scenario by horizontally scaling the infrastructure.

3.1.1 Pipeline components performance measures

This section provides an overview on the performance of individual components of the Pheme pipeline. The performance was measured for each component separately. There are two principal metrics:

- latency – amount of time that each component adds to the processing of a single tweet, that is: how much time each tweet spends in each component
- throughput (tweet rate) – maximum amount of tweets that can be processed by the pipeline in a defined amount of time

The latency measures were performed without any pipeline load, by sending a single tweet individually through each component. Then the process was repeated in order to account for possible anomalies, etc. The result is a distribution of latencies for each component. It is important that this measurement is repeated many times in order to obtain reliable results by average over many cases. Then, average latencies does not show a complete picture of how a component behaves. Latency and throughput very often change in time. This is why we need also to show how components behave in worst cases. For this reason it is very common also to include measures for 95 and 99 percentiles latencies. With this information we know better what can we expect and what happens in some specific corner cases, when unusual latencies occur, such as: garbage collector pauses, some periodic processes that can slow down overall processing, etc.

Latency measures

Below we include charts and tables detailing performance measures for each individual pipeline component. Language id on Figure 3.1 and Table 3.1, spacio-temporal entity annotation on Figure 3.2 and Table 3.2, event clustering on Figure 3.3 and Table 3.3, concept annotation on on Figure 3.4 and Table 3.4, SDQC on Figure 3.5 and Table 3.5 and veracity classification on Figure 3.6 and Table 3.6.

Table 3.1: Latency measures for the language id component

Measure	Latency
mean	0.009 s
median	0.008 s
.95 percentile	0.013 s
.99 percentile	0.014 s

Table 3.2: Latency measures for the spacio-temporal entity annotation component

Measure	Latency
mean	0.010 s
median	0.010 s
.95 percentile	0.013 s
.99 percentile	0.013 s

Table 3.3: Latency measures for the event clustering component

Measure	Latency
mean	0.05 s
median	0.07 s
.95 percentile	0.08 s
.99 percentile	0.09 s

Table 3.4: Latency measures for the concept annotation component

Measure	Latency
mean	0.9 s
median	0.88 s
.95 percentile	1.12 s
.99 percentile	1.27 s

Table 3.5: Latency measures for the SDQC component

Measure	Latency
mean	0.42 s
median	0.42 s
.95 percentile	0.43 s
.99 percentile	0.44 s

Table 3.6: Latency measures for the veracity classification component

Measure	Latency
mean	0.11 s
median	0.11 s
.95 percentile	0.12 s
.99 percentile	0.14 s

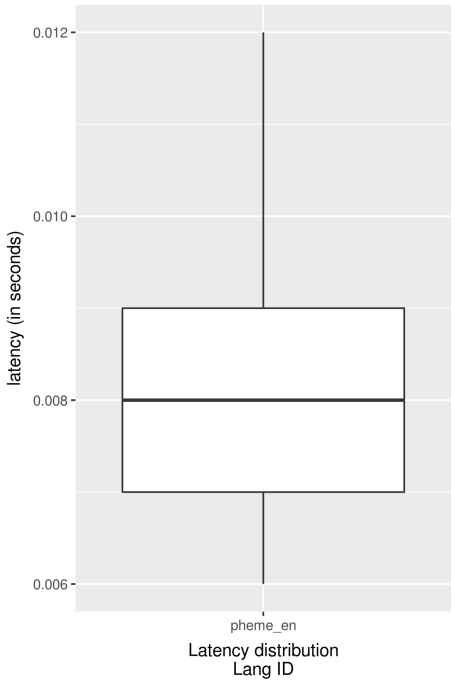

Figure 3.1: Distribution of latency times for the language id component

Throughput measures

Throughput on the other hand is measured by simulating full load of the pipeline at each stage and analyzing the amount of output messages in time. This is done for each component separately for a couple of minutes. Again, throughput can vary in time, so it is important to provide not only average measures but also the lowest quantiles to get the idea of the possible (although rare) ”worst-case” numbers. Table 3.7 shows and overview of the throughput metrics of all components.

Table 3.7: My caption

Component	Mean	Median	.10 percentile
lang id	550 tweets/s	550 tweets/s	500 tweets/s
spacio-temporal entities	37 tweets/s	37 tweets/s	11 tweets/s
event clustering	29.5 tweets/s	10 tweets/s	1 tweet/s
concept annotation	2.4 tweets/s	1 tweets/s	1 tweet/s
SDQC	2.5 tweets/s	1.5 tweets/s	1 tweet/s
Veracity	12 tweets/s	10 tweets/s	2 tweets/s

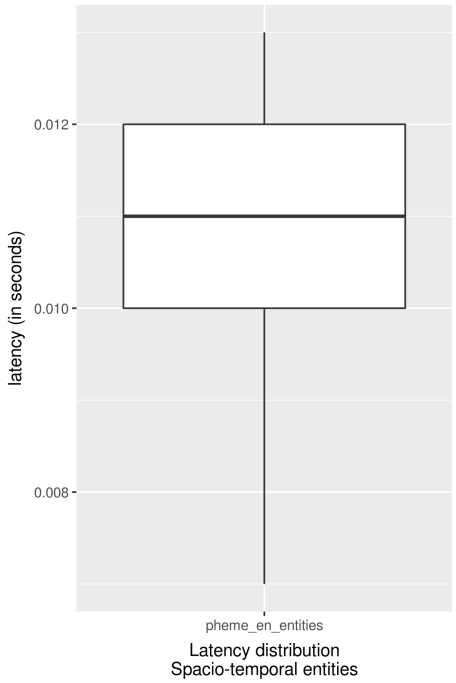

Figure 3.2: Distribution of latency times for the spacio-temporal entity annotation component

Overall pipeline metrics

The same measurements were performed with the whole integrated pipeline. Results for both: latency and throughput are shown in the Figure 3.7

Capture responsiveness

No matter how much effort we put into lowering the latency of the Pheme pipeline, it is important that the data that is fed to the pipeline is as recent as possible. While Twitter provides streaming API that outputs data in a timely fashion, you can only use one stream at the same time, which is not flexible enough for many use cases (especially for the journalism dashboard). When we need to monitor more topics we need to rely on the Search API, which works in a request/response manner. This way we can ”simulate” the stream by pro-actively polling for new data every now and then and stream only the newest updates. This however requires a good degree of fine tunning in order to find a proper balance between the number of monitored events, how ”big” in terms of data those events are, and still conforming to the Twitter API limits. ATOS Capture component was improved in this aspect by lowering the average delay between twitter creation time and the moment it is retrieved by Capture and streamed down the pipeline. Figure 3.8 shows

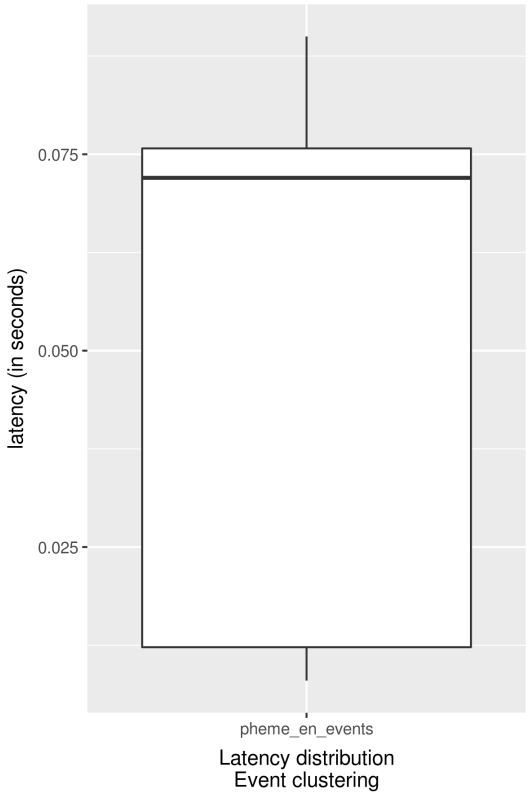

Figure 3.3: Distribution of latency times for the event clustering component

Table 3.8: Latency measures for the language id component

Measure	Delay
median	25 8 s
.95 percentile	55.1 s

the improvement of the overall latency in the process of tweets retrieval.

The measure of Capture latency is shown in Table 3.8. Note that using the Twitter Search API the newest tweets are available after about 16 seconds (as measured by creation time).

3.1.2 Dealing with bursty data rates

While the capacity of the Pheme pipeline can be scaled to handle bigger traffic, sometimes it is important to ensure proper responsiveness. Kafka broker natively provide means for alleviating data peaks, by buffering topics. If data can not be processed instantly, it will simply wait for the client to consume it. This is perfect situation to give consumers more

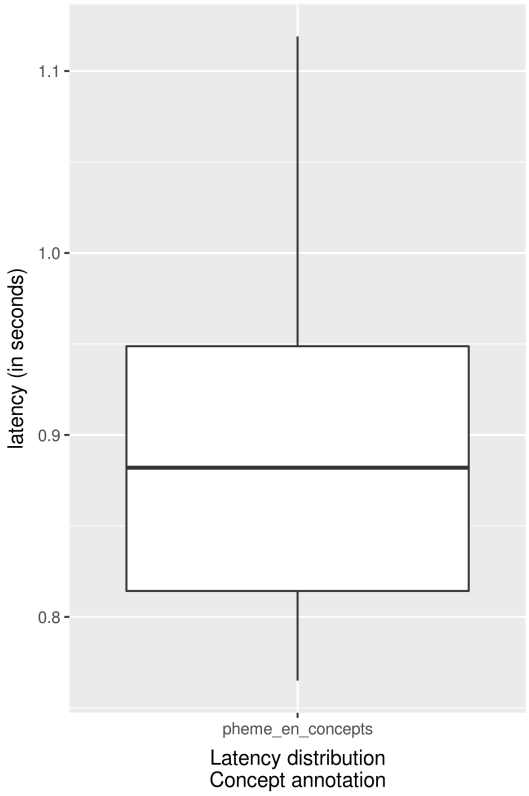

Figure 3.4: Distribution of latency times for the concept annotation component

time to cope with bursty data. This situation however is not always desirable:

- Sometimes a data peak can occur and last for a long time - how long should we maintain such data in the queues? Can we allow queues to grow infinitely?
- When we can not process data in a timely fashion, the queue grows. As the queue grows, every new data coming into the pipeline has to wait until all previous data is processed. This may be OK for long-term, batch processes, but is not suitable for live, interactive work.
- Growing queues are increasing latency of the overall pipeline. One could imagine a situation where a journalist need to research on a very recent topic, but need to wait few hours until other bursty event is processed. His event might already be irrelevant by that time.

Addressing those issues require a good deal of fine tuning and making trade-offs between latency and drop-rate. This is why we decided to configure the journalism pipeline to favor low latency but allow message dropping at the same time, and medical pipeline to reduce drop-rate but allow for higher latencies for bursty data.

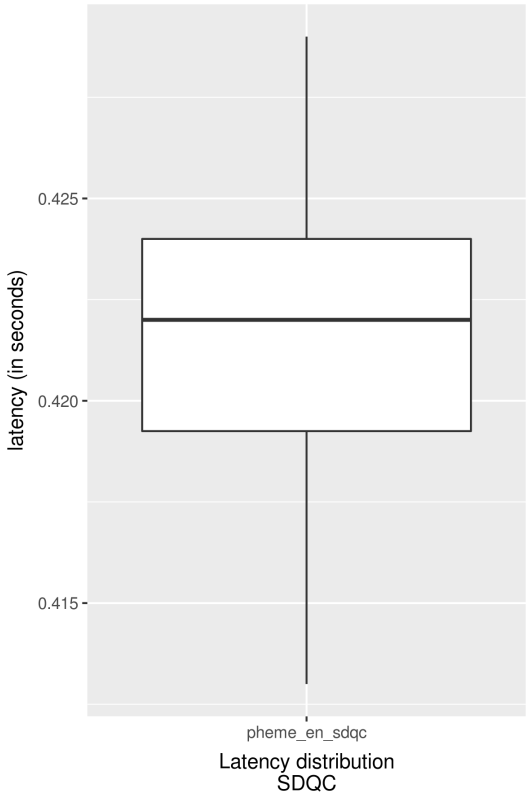

Figure 3.5: Distribution of latency times for the SDQC component

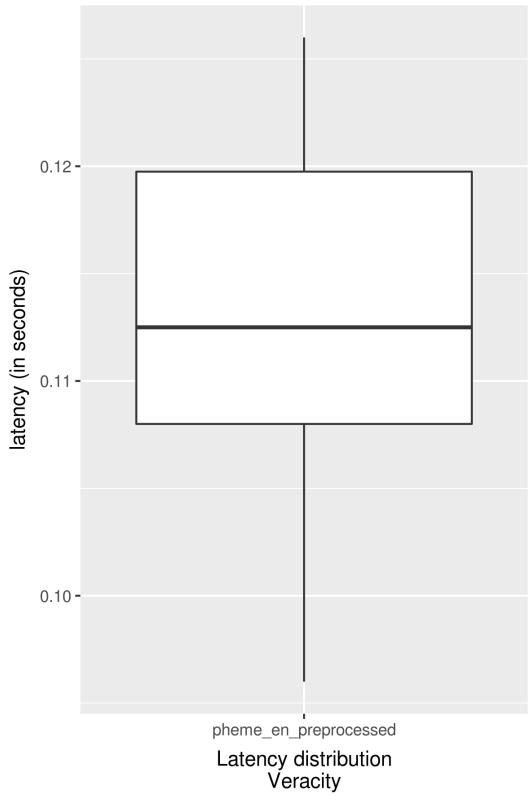

Figure 3.6: Distribution of latency times for the veracity classification component

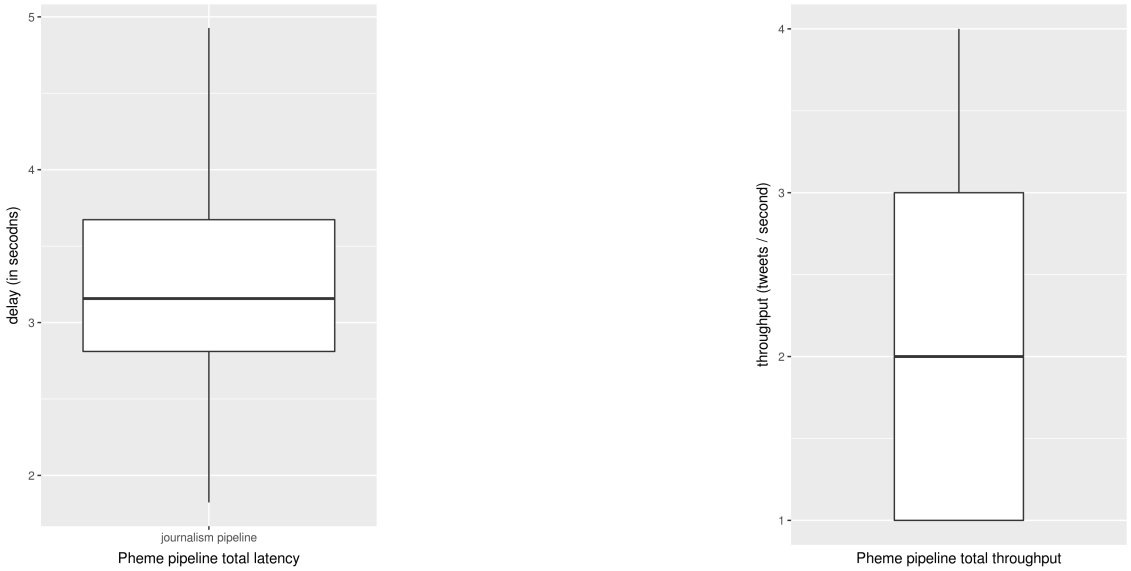

Figure 3.7: Distributions of performance metrics for the complete Pheme pipeline

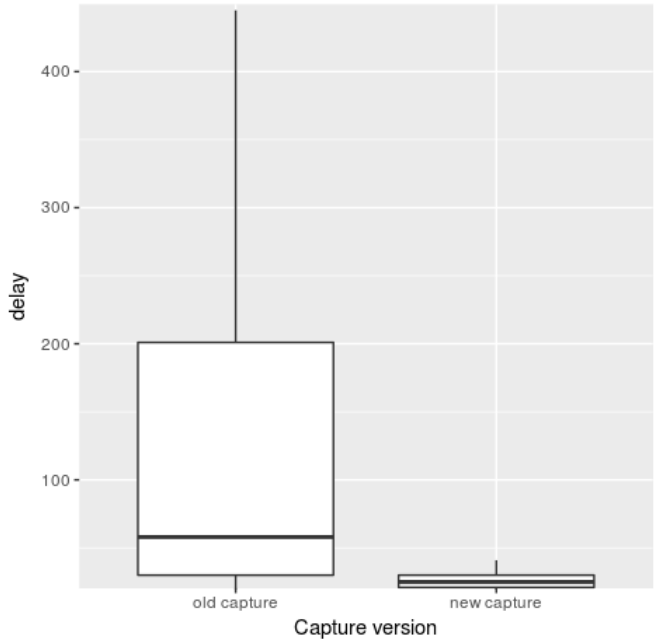

Figure 3.8: Capture latency improvement

Chapter 4

Conclusion

In this deliverable we presented evaluation studies performed in the second part of the project, also building on the former version of this document: D6.2.1 Evaluation Report – Interim Results. We report on 2 types of evaluation, the one dealing with particular components, which are building blocks for the detection and classification of rumours, and on the integration platform that has to make sure that the veracity computation is running in a stable way and can also scale. For the latter, which using Kafka and the underlying Capture infrastructure, we can report on good results, but also on different strategies to be applied for different tasks. Those strategies require fine tuning and have to consider trade-offs between latency and drop-rate, in order to obtain an optimal performances. The integration platform fulfilled its main task, consisting in the delivery of the PHEME Integrated Veracity Framework, which could be tested and run by the partners involved in eh the two use cases of PHEME.

Besides this evaluation study on the integration platform, we presented also detailed evaluation study of methods developed in the context of central components of the PHEMEframework: Sub-Story detection, Entailment and Contradiction detection, Rumour Classification and Rumour Stance Classification. In all cases, we could show adequation of even improvements in comparison with former versions of the involved algorithms or with respect to available gold standards or baselines. As the reader of this document can see, the described algorithms and their results have been published in international workshops and conferences, stressing thus the interest of the community to the results of PHEME, and also ensuring that our work can be-reused in other frameworks or shared tasks.

Bibliography

Agarwal, A., Xie, B., Vovsha, I., Rambow, O., and Passonneau, R. (2011). Sentiment analysis of twitter data. In *Proceedings of the Workshop on Languages in Social Media*, LSM '11, pages 30–38.

Agirre, E., Gonzalez-Agirre, A., Lopez-Gazpio, I., Maritxalar, M., Rigau, G., and Uria, L. (2016). Semeval-2016 task 2: Interpretable semantic textual similarity. In *Proceedings of the 10th International Workshop on Semantic Evaluation, SemEval@NAACL-HLT 2016, San Diego, CA, USA, June 16-17, 2016*, pages 512–524.

Álvarez, M. A., Rosasco, L., and Lawrence, N. D. (2012). Kernels for vector-valued functions: A review. *Found. Trends Mach. Learn.*, 4(3):195–266.

Baldwin, T., Kim, Y.-B., de Marneffe, M. C., Ritter, A., Han, B., and Xu, W. (2015). Shared tasks of the 2015 workshop on noisy user-generated text: Twitter lexical normalization and named entity recognition. *ACL-IJCNLP*.

Beck, D., Cohn, T., and Specia, L. (2014). Joint emotion analysis via multi-task Gaussian processes. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '14, pages 1798–1803.

Benjamini, Y. and Yekutieli, D. (2001). The control of the false discovery rate in multiple testing under dependency. *Annals of Statistics*, 29:1165–1188.

Broder, A. Z. (2000). Identifying and filtering near-duplicate documents. In *Annual Symposium on Combinatorial Pattern Matching*, pages 1–10. Springer.

Brown, N. R. and Schopflocher, D. (1998). Event clusters: An organization of personal events in autobiographical memory. *Psychological Science*, 9(6):470–475.

Brown, P. F., Desouza, P. V., Mercer, R. L., Pietra, V. J. D., and Lai, J. C. (1992). Class-based n-gram models of natural language. *Computational Linguistics*, 18(4):467–479.

Chollet, F. (2015). Keras. `https://github.com/fchollet/keras`.

Cohn, T. and Specia, L. (2013). Modelling annotator bias with multi-task Gaussian processes: An application to machine translation quality estimation. In *51st Annual Meeting of the Association for Computational Linguistics*, ACL '13, pages 32–42.

D'Argembeau, A. and Demblon, J. (2012). On the representational systems underlying prospection: Evidence from the event-cueing paradigm. *Cognition*, 125(2):160–167.

D'Argembeau, A. and Mathy, A. (2011). Tracking the construction of episodic future thoughts. *Journal of Experimental Psychology: General*, 140(2):258.

D'Argembeau, A. and Van der Linden, M. (2012). Predicting the phenomenology of episodic future thoughts. *Consciousness and Cognition*, 21(3):1198–1206.

de Marneffe, M.-C., Manning, C. D., and Potts, C. (2012). Did it happen? The pragmatic complexity of veridicality assessment. *Computational Linguistics*, 38(2):301–333.

Derczynski, L., Augenstein, I., and Bontcheva, K. (2015). USFD: Twitter NER with Drift Compensation and Linked Data. *Proceedings of the Workshop on Noisy User-Generated Text*.

Derczynski, L. and Chester, S. (2016). Generalised Brown Clustering and Roll-Up Feature Generation. In *Proceedings of the conference of the Association for Advancement of Artificial Intelligence*. AAAI.

Ferreira, W. and Vlachos, A. (2016). Emergent: a novel data-set for stance classification. In *Proceedings of NAACL*.

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1026–1034.

Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. R. (2012). Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*.

Hu, Y., Talamadupula, K., Kambhampati, S., et al. (2013). Dude, srsly?: The Surprisingly Formal Nature of Twitter's Language. In *Proc. ICWSM*.

Indyk, P. and Motwani, R. (1998). Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the ACM symposium on Theory of computing*, pages 604–613. ACM.

Lampos, V., Aletras, N., Preotiuc-Pietro, D., and Cohn, T. (2014). Predicting and characterising user impact on twitter. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, EACL'14, pages 405–413.

Lendvai, P., Augenstein, I., Bontcheva, K., and Declerck, T. (2016). Monolingual social media datasets for detecting contradiction and entailment. In *LREC*.

Lendvai, P. and Reichel, U. D. (2016). Contradiction detection for rumorous claims. *CoRR*, abs/1611.02588.

Liang, P. (2005a). Semi-supervised learning for natural language. Master's thesis, Massachusetts Institute of Technology.

Liang, P. (2005b). *Semi-Supervised Learning for Natural Language*. PhD thesis, Department of Electrical Engineering and Computer Science at the Massachusetts Institute of Technology.

Llewellyn, C., Grover, C., Oberlander, J., and Klein, E. (2014). Re-using an argument corpus to aid in the curation of social media collections. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation*, LREC'14, pages 462–468.

Lukasik, M., Bontcheva, K., Cohn, T., Tolosi, L., and Georgiev, G. (2015a). D4.3.1 Algorithms for Detecting Misinformation and Disinformation: Initial Prototype. Technical report, University of Sheffield.

Lukasik, M., Cohn, T., and Bontcheva, K. (2015b). Classifying tweet level judgements of rumours in social media. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 2590–2595.

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 *ACL (2)*. The Association for Computer Linguistics.

Maas, A. L., Hannun, A. Y., and Ng, A. Y. (2013). Rectifier nonlinearities improve neural network acoustic models. In *Proc. ICML*, number 1.

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

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Minka, T. and Lafferty, J. (2002). Expectation-propagation for the generative aspect model. In *Proceedings of the Eighteenth Conference on Uncertainty in Artificial Intelligence*, UAI'02, pages 352–359.

Mohammad, S. M., Kiritchenko, S., Sobhani, P., Zhu, X., and Cherry, C. (2016). SemEval-2016 Task 6: Detecting stance in tweets. In *Proceedings of the International Workshop on Semantic Evaluation*, SemEval '16, San Diego, California.

Morante, R. and Blanco, E. (2012). *sem 2012 shared task: Resolving the scope and focus of negation. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics - Volume 1: Proceedings of the Main Conference and the Shared Task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation*,

SemEval '12, pages 265–274, Stroudsburg, PA, USA. Association for Computational Linguistics.

Morante, R. and Sporleder, C. (2012). Modality and negation : an introduction to the special issue. *Computational linguistics*, 38(2):223–260.

Owoputi, O., Dyer, C., Gimpel, K., Schneider, N., and Smith, N. A. (2013). Improved part-of-speech tagging for online conversational text with word clusters. In *Proceedings of NAACL*, pages 380–390.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al. (2011). Scikit-learn: Machine learning in python. *The Journal of Machine Learning Research*, 12:2825–2830.

Pennington, J., Socher, R., and Manning, C. D. (2014). GloVe: Global Vectors for Word Representation. In *Proc. EMNLP*, volume 14, pages 1532–43.

Petrović, S., Osborne, M., and Lavrenko, V. (2010). Streaming first story detection with application to Twitter. In *Proceedings of NAACL*, pages 181–189. ACL.

Porter, M. F. (1997). Readings in information retrieval. chapter An Algorithm for Suffix Stripping, pages 313–316. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Preotiuc-Pietro, D., Lampos, V., and Aletras, N. (2015). An analysis of the user occupational class through twitter content. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015*, pages 1754–1764.

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

Pustejovsky, J., Hanks, P., Sauri, R., See, A., Gaizauskas, R., Setzer, A., Radev, D., Sundheim, B., Day, D., Ferro, L., et al. (2003). The TimeBank corpus. In *Proc. Corpus Linguistics*, volume 2003, page 40.

Pustejovsky, J., Lee, K., Bunt, H., and Romary, L. (2010). ISO-TimeML: An International Standard for Semantic Annotation. In *Proc. LREC*.

Pustejovsky, J., Moszkowicz, J. L., and Verhagen, M. (2011). ISO-Space: The annotation of spatial information in language. In *Proceedings of the Sixth Joint ISO-ACL SIGSEM Workshop on Interoperable Semantic Annotation*, pages 1–9.

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*, EMNLP '11, pages 1589–1599.

Rand, W. M. (1971). Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical association*, 66(336):846–850.

Rasmussen, C. E. and Williams, C. K. I. (2005). *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press.

Reichel, U. D. (2012). PermA and Balloon: Tools for string alignment and text processing. In *Proc. Interspeech*, page paper no. 346, Portland, Oregon.

Reichenbach, H. (1947). The tenses of verbs. In *Elements of Symbolic Logic*. Macmillan.

Ritter, A., Clark, S., Etzioni, O., et al. (2011). Named entity recognition in tweets: an experimental study. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1524–1534. ACL.

Rosenberg, A. and Hirschberg, J. (2007). V-Measure: A Conditional Entropy-Based External Cluster Evaluation Measure. In *Proc. CoNLL*, volume 7, pages 410–420.

Smith, T. F. and Waterman, M. S. (1981). Identification of common molecular subsequences. *Journal of Molecular Biology*, 147(1):195–197.

Srijith, P., Hepple, M., Bontcheva, K., and Preotiuc-Pietro, D. (2016). Sub-story detection in Twitter with hierarchical Dirichlet processes. *Information Processing & Management*.

Statistics, L. B. and Breiman, L. (2001). Random forests. In *Machine Learning*, pages 5–32.

Tibshirani, R., Hastie, T., Narasimhan, B., and Chu, G. (2003). Class prediction by nearest shrunken centroids, with applicaitons to dna microarrays. *Stat Sci*, page 117.

Tolosi, L., Tagarev, A., and Georgiev, G. (2016). An Analysis of Event-Agnostic Features for Rumour Classification in Twitter. *International AAAI Conference on Web and Social Media*.

Wagner, R. A. and Fischer, M. J. (1974). The string-to-string correction problem. *J. ACM*, 21(1):168–173.

Wang, C., Paisley, J. W., and Blei, D. M. (2011). Online Variational Inference for the Hierarchical Dirichlet Process. In *Proc. AISTATS*, volume 2, page 4.

Xu, W., Callison-burch, C., and Dolan, W. B. (2015). Semeval-2015 task 1: Paraphrase and semantic similarity in twitter (pit).

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). Early detection of rumors in social media from enquiry posts. In *International World Wide Web Conference Committee (IW3C2)*.

Zubiaga, A., Liakata, M., Procter, R., Bontcheva, K., and Tolmie, P. (2015a). Crowdsourcing the annotation of rumourous conversations in social media. In *Proceedings of the 24th International Conference on World Wide Web*, WWW '15 Companion, pages 347–353. International World Wide Web Conferences Steering Committee.

Zubiaga, A., Liakata, M., Procter, R., Bontcheva, K., and Tolmie, P. (2015b). Towards Detecting Rumours in Social Media. *CoRR*, abs/1504.04712.

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.