Probabilistic Modeling of Rumour Stance and Popularity in Social Media

By:

Michal Lukasik

A thesis submitted in fulfilment of the requirements for the degree of Doctor of Philosophy

The University of Sheffield
Faculty of Engineering
Department of Computer Science

Abstract

Social media tends to be rife with rumours when new reports are released piecemeal during breaking news events. One can mine multiple reactions expressed by social media users in those situations, exploring users' stance towards rumours, ultimately enabling the flagging of highly disputed rumours as being potentially false. Moreover, rumours in social media exhibit complex temporal patterns. Some rumours are discussed with an increasing number of tweets per unit of time whereas other rumours fail to gain ground.

This thesis develops probabilistic models of rumours in social media driven by two applications: rumour stance classification and modeling temporal dynamics of rumours. Rumour stance classification is the task of classifying the stance expressed in an individual tweet towards a rumour. Modeling temporal dynamics of rumours is an application where rumour prevalence is modeled over time. Both applications provide insights into how a rumour attracts attention from the social media community. These can assist journalists with their work on rumour tracking and debunking, and can be used in downstream applications such as systems for rumour veracity classification.

In this thesis, we develop models based on probabilistic approaches. We motivate Gaussian processes and point processes as appropriate tools and show how features not considered in previous work can be included. We show that for both applications, transfer learning approaches are successful, supporting the hypothesis that there is a common underlying signal across different rumours. We furthermore introduce novel machine learning techniques which have the potential to be used in other applications: convolution kernels for streams of text over continuous time and a sequence classification algorithm based on point processes.

Acknowledgements

I am very grateful to my supervisors Kalina Bontcheva and Trevor Cohn for their support and guidance throughout my PhD. I am grateful to Kalina for introducing me to social media research and supporting me throughout my studies. Trevor shaped me as a researcher, it was wonderful to work together on so many projects and explore the many exciting ideas that we had.

I would like to express my gratitude to my examiners, Andreas Vlachos and Stephen Clark, who greatly helped in improving this thesis.

I was very lucky to have worked closely with Srijith P.K. It is hard to overestimate the impact he had on my PhD.

Neil Lawrence and his group greatly inspired me to enter the depths of the exciting field of machine learning. I benefited tremendously from attending the Gaussian process schools and frequently interacting with the group.

Thank you to my collaborators and colleagues: Arkaitz Zubiaga, Duy Vu, Zsolt Bitvai, Daniel Beck, Maria Liakata, Rob Procter, Varvara Logacheva, Tomasz Kusmierczyk, and others, who greatly impacted my research through numerous conversations, shared ideas, and code.

I am grateful to Richard Zens for inviting me for an internship at Google Research and to Manuel Gomez-Rodriguez for inviting me for an internship at the Max Planck Institute for Software Systems. These experiences enriched me greatly.

Thanks to all the friends I made while I was working on my PhD at both the University of Sheffield and the University of Melbourne. Because of you my journey was very enjoyable.

Last but not the least, I thank my parents who have always supported me and my brothers for putting up with me.

Contents

List of Figures

List of Tables

Nomenclature

Mathematical Notation

$\lambda(x)$ A hazard/intensity function.

$\mathbf{Rel}$ A relation function.

$\mathbf{v}$ Vectors are represented by small boldfaced letters.

$f(x)$ A function drawn from a Gaussian process.

I An identity matrix.

$k(\mathbf{v}, \mathbf{v})$ Kernel functions are denoted with small letters.

$K(X, X')$ Matrices of kernel function values are denoted with capital letters corresponding to the symbol used for the kernel function, and two arguments denoting the sequences of inputs that the kernel is being evaluated on (e.g. $K(X, X')$ denotes a matrix of kernel function evaluations, where cell (i, j) contains the kernel evaluation $k(X_{i.}, X'_{j.})$.

X Matrices are represented by capital and not boldfaced letters, unless convention for a particular quantity is different in the literature.

Data Notation

$\mathbf{p}_m^n$ The nth post from rumour R_m.

$\mathbf{w}_m^n$ The text posted in the nth post from rumour R_m.

$\mathbf{w}_n$ The text message from the nth tweet, encoded using a vector representation.

i_n The id of a user who posted the nth tweet.

m_n The rumour id corresponding to the nth tweet.

P	The set of tweets from all rumours R.
R	The set of rumours.
R_n	The nth rumour from the set of rumours.
T	The end of the observation time interval.
t_m^n	The timestamp of occurrence of post $\mathbf{p}_m^n$.
t_n	The timestamp of occurrence of the nth tweet.
U	The set of users.
V	The size of the vocabulary.
W	The matrix of vector representations of tweets, such that W_{nv} is the count of the vth word from the vocabulary in the nth tweet. We refer to the nth row of W as $\mathbf{w}_n$.
Y	The set of stances expressed in tweets around rumours.

Chapter 1

Introduction

Social media is a rich source of information about events occurring in the real world. People report stories from first hand experience, often providing a quicker and more detailed description of news than traditional news websites (Kwak et al., 2010), or even describing events not present in the mainstream news (Petrovic, 2012). Citizens are often the source of information themselves, with social media making for a platform for dissemination (Goode, 2009).

Together with the benefit of being a rich source of information and providing information at fast speed there are some downsides compared to traditional journalism. Specifically, social media is very prone to misinformation, as it is hard to verify what is quickly being spread (Mendoza et al., 2010). Thus, there is an increasing need to interpret and act upon rumours (unverified pieces of information, the veracity of which is uncertain) spreading quickly through social media, especially in circumstances where their veracity is hard to establish.

One setting in which rumours often tend to emerge are unrests and disastrous events. For instance, during an earthquake in Chile rumours spread through Twitter that a volcano had become active and that there was a tsunami warning in Valparaiso (Mendoza et al., 2010). During the riots in Ferguson in 2014 various rumours were spreading (Zubiaga et al., 2016c). Officials would have benefited from being able to learn particularly popular rumours and act on debunking them. Other examples, from the riots in England in 2011, are that rioters were going to attack Birmingham's children hospital and that animals had escaped from the zoo (Procter et al., 2013a), both untrue.

Rumours also tend to spread in other settings. There have been many reports that during the presidential campaign in the USA 2016 misinformation has been acquiring a worrying degree of popularity (Allcott and Gentzkow, 2017). Rumours tend to spread about popular

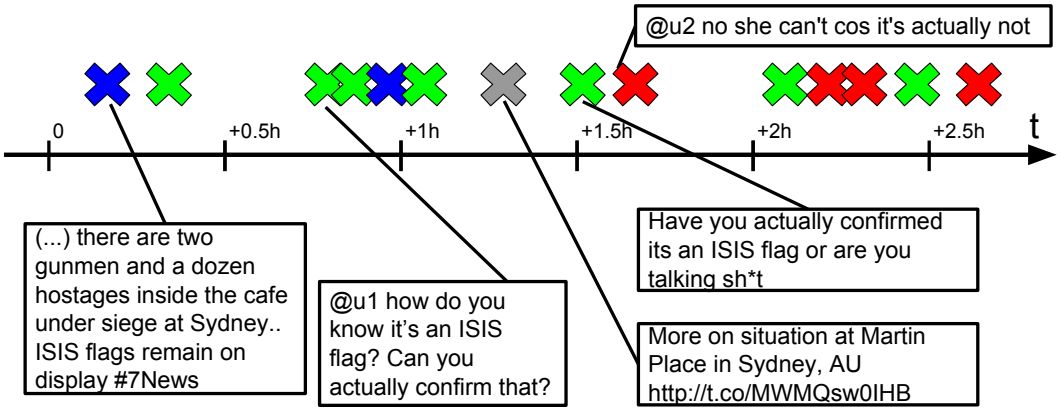


Figure 1.1: An illustrative example rumour about the ISIS flag being displayed on the cafeteria besieged in Sydney in 2014. Notice how tweets around the rumour occur at varying density, and their content denotes different stances with respect to the veracity of the rumour. Blue crosses denote tweets supporting the veracity of the rumour, green crosses denote questioning tweets, red crosses denote denying tweets, and the grey cross denotes a neutral tweet with respect to the truthfulness of the rumour.

products after (or even before) they are launched. For example, one of the most popular groups of rumours on a website analysing rumours spreading on-line (`www.emergent.info`) is about Apple and its products. Companies might be losing revenue due to rumours around their products gaining popularity.

The social media platform which serves as the source of our rumour datasets is Twitter (www.twitter.com). A rumour in Twitter is composed of tweets, each of which is described (among the others) by a timestamp of the tweet occurrence and the text content. In Figure 1.1 we depict an example rumour from the Sydney Siege in 2014 (Zubiaga et al., 2016c). It starts with a tweet stating the rumour that there is an ISIS flag displayed at the besieged cafeteria, and is followed by tweets around that statement. Tweets typically express stances of authors discussing the veracity of the rumour. For example a questioning tweet asks the author of the statement how they know that it is true, and a rejecting tweet states that a supporting photo has been taken before the event. Moreover, the number of tweets over different temporal intervals varies, with lower interest from the community at the beginning, and higher towards the end of the rumour lifespan.

1.1 Statement of the Problem

In this thesis we explore applications supporting journalists in tracking rumours in social media. We aim at developing tools that would provide insightful information about rumours, allowing journalists to quickly analyse people's orientation towards the veracity of a rumour and/or direct their efforts to studying particularly interesting stories. One recognized application is rumour stance classification. Rumour stance classification is the task of classifying a tweet about a rumour into one of the categories describing what stance it takes with respect to rumour veracity: supporting, denying, questioning or commenting (we formally define the task in Section 2.2). It provides a journalist with information on how users in social media react to rumours. It has also been conjectured to be a crucial first step towards making an informed judgement on the veracity of a rumour (Zubiaga et al., 2016c; Tolmie et al., 2015; Mendoza et al., 2010). Automating stance categorisation of tweets would be of great use in tracking rumours, flagging those that are largely denied or questioned as being more likely to be false.

A different, less well studied aspect of rumours, is popularity. Future patterns of tweet arrivals convey information which may prove useful to journalists, for example by helping them direct the rumour debunking effort to rumours which are going to rapidly spread, ignoring those which are not going to attract as much attention. Rumour popularity modeling can also be motivated by other applications. For instance, temporal dynamics of a rumour may provide additional information for veracity classification of a rumour. In particular, the tweeting behaviour from Internet trolls might exhibit a specific pattern, which may indicate a rumour is false (Ratkiewicz et al., 2011). Moreover, the ability to predict popular rumours could be useful for the purpose of engineering a popular story. Namely, a popularity prediction tool could be used to forecast how viral a given story would go.

1.2 Aims and Research Questions

The aims of this thesis are: exploring the rumour stance classification (1) and the rumour popularity prediction (2) problems. Below we elaborate on them and state research questions that we are going to consider.

1. The first aim of this thesis is **predicting stance of tweets regarding rumours**.

   Rumour stance classification is an established task in the NLP community (Zubiaga et al., 2017) which has found many applications. It provides journalists and authorities with tools for rumour tracking. Moreover, rumour stance classification has been

successfully used for rumour detection (Zhao et al., 2015b), and rumour veracity classification (Derczynski et al., 2015; Liu et al., 2015).

(a) **What would be a realistic and fair evaluation framework for rumour stance classification?**

A rumour in social media is represented by the posts discussing it. A realistic rumour stance classification setting should mimic how a journalist or a real world system would conduct the task. In particular, a realistic scenario would not allow for observing labels from tweets about a rumour which happened after the tweets that need to be classified. Perhaps the most realistic scenario is when no annotation is available for the target rumour. Previous work on rumour stance classification conducted evaluation via cross-validation, randomly shuffling the tweets regardless of their rumour identities or timestamps (Qazvinian et al., 2011). In this research question we seek an experimental setup respecting the time series nature of tweets, and the fact that tweets come from different rumours.

(b) **Can information about stances of tweets from one rumour be useful for predicting stances of tweets from another rumour?**

Here, we look at whether similar patterns for different stance categories occur across rumours, and whether stance information can be successfully transferred across rumours. Overall, we might expect that the way a rumour is rejected in different rumours bears similar characteristics, e.g. linguistic cues such as words *fake* and *false*. However, different rumours might exhibit their own characteristics regarding how tweets express stances. For example, some rumours might be sarcastic with users indirectly expressing their denial, whereas other rumours less so. We investigate this research question by experimenting with models learning similarities between rumours and comparing them against baselines.

(c) **Do tweet arrival times carry complementary information to text for the stance prediction task?**

Previous work on rumour stance classification focused on features based on the text content of tweets. However, tweets about a rumour occur over time. This temporal information about tweets might be useful for the rumour stance classification problem. For example, one might expect that tweets which happen early on during the rumour lifespan might be likely to be supporting. In this

research question, we look at whether temporal dynamics of tweet arrivals conveys useful information for the rumour stance classification task, and whether it can bring additional information compared to the textual content of a tweet. We inspect this research question by employing the temporal feature from tweets into a model, and investigating if the resulting approach can outperform baselines not using temporal information.

2. The second aim is **predicting rumour popularity**.

   By the popularity of a rumour we mean how widely discussed it is. It is deliberately left unspecified here what we mean by predicting popularity, as it can be defined in different ways (in the first research question we look at what might be a useful definition of predicting popularity of a rumour in social media). Predicting rumour popularity can be useful for multiple applications. It would provide journalists and authorities with tools helping direct attention to those the community is going to find particularly interesting. Also, predictions made about the popularity of a rumour can be useful for downstream applications, such as rumour veracity classification (Are popular rumours more or less likely to be true?) or rumour detection (Does popularity of rumours differ from popularity of other types of stories, e.g. news articles?).

   (a) **How can the rumour popularity prediction problem be formulated?**

   The first question we investigate is how the rumour popularity prediction problem can be defined. This could be done in different ways, with one possible approach being defining it as binary classification into popular and non-popular rumour categories. Such a setting would require defining what a popular rumour is, and various possibilities for that exist. Should a popular rumour be defined as one with the number of tweets in the next 24 hours exceeding a threshold θ, or rather one which is still going to be discussed after a future timestamp ρ? There are many ways a popular rumour could be defined, and it would likely require specifying a parameter separating popular and non-popular rumours.

   A different approach to defining the rumour popularity prediction task could be to specify it as the task of predicting how tweets about a rumour are going to arrive over time. This avoids the arbitrariness of converting the problem into binary classification, and leaves the interpretation of what a popular rumour is to a journalist looking at the output. Moreover, the information about how tweets arrive over time could be directly useful for downstream applications. For instance, the way tweets are arriving over time might reveal important information

for rumour detection and rumour veracity classification tasks.

(b) **Can information about tweet arrivals from one rumour bring useful information for predicting the popularity of another rumour?**

Tweet arrivals over time can exhibit complex patterns, with varying inter-arrival times between different pairs of consecutive tweets (see Figure 1.1). At the same time, one might expect that there might be common patterns of tweet arrivals across different rumours. For example, two rumours which are highly plausible (or just funny) might exhibit very low inter-arrival time between consecutive tweets (or even diminishing inter-arrival time as rumour is discussed more). On the other hand, some changes in how tweets arrive over time may be due to unpredictable phenomena, such as an external source providing additional information about a rumour, or more interesting stories emerging. Here, we investigate whether information about how tweets arrive over time around one rumour can bring useful information when making predictions about the popularity of different rumours. We investigate this research question by experimenting with models which learn similarities between rumours and comparing them against baselines which do not learn such similarities.

(c) **Does the text content of tweets convey useful information for the rumour popularity prediction task?**

Text is an essential part of a tweet. It has been shown to be predictive of stance with respect to rumour veracity (Qazvinian et al., 2011). Here, we investigate if what is being tweeted about a rumour conveys information about whether the rumour is going to be popular. One motivation for trying the text feature is that supported and denied rumours might exhibit different temporal dynamics. Since text has been shown to indicate stance with respect to a rumour, the predominant stance in tweets can be established, and based on this different predictions about rumour popularity be made (e.g., a supported rumour might be going viral, whereas a rejected rumour not so much). However, the motivation for using the text feature goes beyond the study of tweet stances regarding rumour veracity, as a rumour which is simply found funny in the community might also be going viral. For this research question we investigate if text provides useful information into the rumour popularity prediction task. We investigate this by looking if models employing text information from posts can outperform approaches which do not use text information.

(d) **Does information about how text usage in tweets changes over time convey**

useful information for the rumour popularity prediction task?

Tweets posted at different times about a rumour may use different words. For example, a rumour which has been supported towards the beginning of its lifespan and rejected towards the end of its lifespan would be described by tweets using different words, depending on when a tweet was posted. Tweets from the beginning would probably be described by words describing support (e.g. *true*, *witness*), and tweets posted later on could use words describing denial (e.g. *fake*, *false*). Similarly, a rumour which had been found funny towards the beginning of a rumour lifespan, but later the community lost interest in it, would also likely exhibit different usage of words in tweets over time. In this research question we look at whether the change of how text is used in tweets over time conveys useful information for the rumour popularity prediction task. We investigate this by comparing models which use information about language change over time against baselines which do not use such information.

1.2.1 Scope

We consider two problems related to rumours: rumour stance classification and rumour popularity prediction. There exist other applications related to rumours which are outside of the scope of thesis. In particular, instead of running rumour detection algorithms (the purpose of which is identifying rumours in the stream of social media posts; see Section 2.4 for details about rumour detection), we make use of manually curated rumour datasets. In principle, one could use an automatic rumour detection system for a fully end-to-end tool. Furthermore, even though one motivation for the rumour stance classification task is predicting rumour veracity, we are not considering an end-to-end system for this particular problem. Nevertheless, the tools we develop can assist journalists in their efforts towards rumour debunking. Moreover, we limit ourselves to two rumour datasets coming from the Twitter social media platform: rumours from the England riots of 2011 (Procter et al., 2013a) and the PHEME rumour datasets collected around several events, mostly related to riots (Zubiaga et al., 2015a). The datasets comprise tens of thousands of tweets, and this to a large extent motivates our modeling choices. Larger datasets can be considered, however these would require adjusting the proposed models (see future work directions in Section 8.2).

1.3 Thesis Structure

The rest of this thesis is organized as follows.

- Chapter 2 provides background on rumours in social media. We describe different aspects of rumours, such as their definition and different applications that were studied. We focus on the two problems considered in this thesis: rumour stance classification and rumour popularity prediction, identifying the shortcomings of previous work that we address in this thesis. We also review several other related problems.

- Chapter 3 reviews machine learning frameworks we use for modeling rumours. First, we introduce Gaussian processes (GPs). GPs have characteristics that are useful in our settings. The model is Bayesian, which helps make robust predictions in the absence of large training data. The efficient hyperparameter selection mechanism allows for avoiding extensive heldout hypermarameter tuning. Moreover, the kernelized nature of GPs allows for modeling multiple rumours via multi-task learning kernels. We make use of GPs in Chapters 4 to 6.

  Next, we describe point processes, a framework for modeling point occurrences. Point processes can be defined using a function over time called the hazard rate, which informally can be thought of as a likelihood of a tweet occurrence at a particular point of time. The hazard rate function can be used to answer various questions, such as: *How many tweets are going to occur in an interval* $[t_S, t_E]$*?* and *When will the next tweet occur after time* t*?* This makes them very appealing for complex tasks requiring making predictions about how a phenomenon is going to propagate over time. Depending on what point process one chooses to work with, different kinds of temporal phenomena can be modeled. We describe two specific models from this framework: Log-Gaussian Cox Processes and Hawkes processes. We make use of point processes for modeling dynamics of rumours in Chapters 5 to 7.

- Chapter 4 addresses research questions 1a and 1b, i.e., what the realistic settings for rumour stance classification are and whether stances from tweets about one rumour can be transferred to make predictions about another rumour. We move away from cross validation used in previous work (Qazvinian et al., 2011), and introduce two experimental scenarios which address the time series characteristic of rumours. In the first setting the test set is composed of tweets from the target rumour, and no annotated tweets from the target rumour are available in the training set. In the second setting a small amount of annotated initial tweets is available in the training set.

Next, we motivate Gaussian processes, and show how a multi-task kernel learning can successfully leverage data from multiple rumours to make predictions for a different rumour.

- Chapter 5 addresses research questions 2a about how rumour popularity prediction can be defined, 2b about whether tweet arrivals from one rumour can be helpful to make predictions about the popularity of another rumour, and 2c about whether text from tweets provides useful information for the rumour popularity prediction task. We introduce the problem of rumour popularity prediction in a fine grained setting, leaving flexibility to an end user regarding the interpretation of the result. Namely, the task is to predict the number of tweets in the unobserved time intervals from the rumour lifespan. We motivate point processes as an appropriate framework for solving this problem. Finally, we demonstrate how information about tweet arrivals from other rumours, as well as text from tweets, can be incorporated into the point process model, yielding improvements over baseline results.

- Chapter 6 addresses research question 2d about whether information about how language in tweets around a rumour changes over time is useful for the rumour popularity prediction task. To this end, we introduce a novel convolution kernel and show how it can be used in the point process framework for modeling rumour popularity. As a result, we show that incorporating text dynamics over time improves results compared to those obtained in Chapter 5. Moreover, we demonstrate the applicability of the kernel to a range of other problems, both on synthetic and real datasets.

- Chapter 7 comes back to the first research aim, and addresses the research question 1c about whether dynamics of tweet arrivals from a rumour can provide useful information for rumour stance classification. Inspired by the work on rumour popularity prediction from Chapters 5 and 6 we incorporate rumour dynamics over time by employing the point process framework. As a result, we consider the Hawkes process model, which explicitly models influences between tweets. We find this characteristic particularly useful when modeling rumour stance classification, as we can learn how tweets expressing different stances influence one another over time. We show how the Hawkes process based approach achieves competitive results compared to the methods from Chapter 4.

- We conclude this thesis in Chapter 8 by revisiting the research questions. We also discuss directions for future work.

1.4 Published Material

Parts of this thesis have been published in conference proceedings.

- Rumour stance classification work described in Chapter 4 is an extended version of the work published as

  > Lukasik, M., Cohn, T., and Bontcheva, K. (2015a). Classifying tweet level judgements of rumours in social media. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

  Software and dataset used for experiments can be found at `https://github.com/mlukasik/rumour-classification`.

  The chapter has been submitted to Journal of the Association for Information Science and Technology, and is currently under review. The preprint version has been released as

  > Lukasik, M., Bontcheva, K., Cohn, T., Zubiaga, A., Liakata, M., and Procter, R. (2016a). Using Gaussian processes for rumour stance classification in social media. *CoRR*, abs/1609.01962.

- Work described in Chapter 5 includes the work published as

  > Lukasik, M., Cohn, T., and Bontcheva, K. (2015b). Point process modelling of rumour dynamics in social media. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (ACL-IJCNLP)*, pages 518–523.

  and

  > Lukasik, M., Srijith, P. K., Cohn, T., and Bontcheva, K. (2015c). Modeling tweet arrival times using log-Gaussian Cox processes. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 250–255.

- Chapter 6 includes the work published as

  > Lukasik, M. and Cohn, T. (2016). Convolution kernels for discriminative learning from streaming text. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence*.

- Chapter 7 includes the work published as

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

Software and dataset used for experiments can be found at `https://github.com/mlukasik/seqhawkes`.

The author was the main contributor to the above publications, with collaborators supporting writing and evaluation.

The author has also contributed to other publications about modeling rumours in social media. Work published as

Zubiaga, A., Kochkina, E., Liakata, M., Procter, R., and Lukasik, M. (2016a). Stance classification in rumours as a sequential task exploiting the tree structure of social media conversations. In *Proceedings of the 26th International Conference on Computational Linguistics (COLING)*, pages 2438–2448

extends the work on rumour stance classification, exploring further features and methods, whereas work published as

Srijith, P. K., Lukasik, M., Bontcheva, K., and Cohn, T. (2017). Longitudinal modeling of social media with Hawkes process based on users and networks. In *Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*

extends the work on rumour popularity modeling, also exploring additional features and methods. The author was not the main contributor to these works, and they are not included in this thesis.

Chapter 2

Rumours in Social Media

This chapter gives an overview of research on rumours in social media. We start by introducing the formal definition of a rumour. Afterwards, we provide an overview of research in two applications regarding rumours in social media: rumour stance classification and rumour popularity prediction. We also review several other related research areas.

2.1 Rumour Definition

There have been multiple attempts at defining rumours in the literature. Most of them are complementary to one another, with slight variations depending on the context of their analyses. The core concept that most researchers agree on matches the definition that major dictionaries provide. For instance the Oxford English Dictionary[1] definition that a rumour as "a currently circulating story or report of uncertain or doubtful truth" matches DiFonzo and Bordia (2007) definition of rumours as "unverified and instrumentally relevant information statements in circulation."

Researchers have long looked at the properties of rumours to understand their diffusion patterns and to distinguish them from other kinds of information that people habitually share (Donovan, 2007). Allport and Postman (1947) claimed that rumours spread due to two factors: people want to find meaning in things and, when faced with ambiguity, they try to find meaning by telling stories. The latter factor also explains why rumours tend to change over time by becoming shorter, sharper and more coherent. This is the case, it is argued, because such changes make rumours explain things more clearly. In another work, Rosnow (1991) claimed that there are four important factors for rumour transmission. Rumours must be outcome-relevant to the listener, must increase personal anxiety, be somewhat credible

[1] `http://www.oxforddictionaries.com/definition/english/rumour`

and be uncertain. Furthermore, Shibutani (1969) defined rumours to be "a recurrent form of communication through which men [sic] caught together in an ambiguous situation attempt to construct a meaningful interpretation of it by pooling their intellectual resources. It might be regarded as a form of collective problem-solving".

In contrast to these theories, Guerin and Miyazaki (2006) state that a rumour is a form of relationship-enhancing talk. Building on their previous work, they recall that many ways of talking serve the purpose of forming and maintaining social relationships. Rumours, they say, can be explained by such means.

In our work, we adhere to the widely accepted fact that rumours are unverified pieces of information. More specifically, similar as Zubiaga et al. (2016c), we regard a rumour in the context of breaking news, as a "circulating story of questionable veracity, which is apparently credible but hard to verify, and produces sufficient skepticism and/or anxiety so as to motivate finding out the actual truth".

We deal with rumours in social media where they are represented as sets of tweets discussing a particular rumour. Typically, as we discuss in Section 2.5, it is assumed that tweets replying or retweeting a tweet about a rumour are about the same rumour too. In our work we do not deal with automatic rumour detection, and assume it has already been determined what tweets pertain to a rumour (in our case via manual annotation, however, in principle, this can be done using a rumour detection system; see Section 2.4 for a discussion about rumour detection).

2.2 Rumour Stance Classification

Rumour stance classification is the task of classifying tweets around rumours into categories denoting their attitude regarding the rumour veracity: supporting, denying, questioning or commenting. The categories vary from work to work, however the underlying concept is the same: finding individual judgements of a rumour in the community.

The rumour stance classification task recently became popular, with numerous papers (Zeng et al., 2016b; Zubiaga et al., 2016a; Liu et al., 2015; Hamidian and Diab, 2016) and a recent shared task (Derczynski et al., 2017) demonstrating interest from the research community. Below, we formally define the task and discuss different aspects of previous work, emphasizing where we seek improvements.

Definition of the Task Individual tweets may discuss the same rumour in different ways, with each user expressing their own stance towards the rumour. Within this scenario, the tweet-level rumour stance classification task is defined as that in which a classifier has to

determine the stance of each tweet towards the rumour. In the rumour stance classification task, given the tweet t_i as input, the classifier has to determine which of the set of stances $Y = \{\text{supporting, denying, questioning, commenting}\}$ applies to the tweet, $y(t_i) \in Y$. Figure 1.1 depicts an example rumour about an ISIS flag being displayed on the cafeteria besieged in Sydney. Example stances with respect to the rumour are shown, and we resort to this example when discussing definitions of stances below.

The set of considered categories varies from work to work, however do not significantly differ (i.e. are either subsets of one another, or it is possible to convert categories across the different schemes). In Section 2.5.2 we show how a different scheme introduced by Zubiaga et al. (2016c) can be converted to the scheme of four stances used in this study.

Definition of the Stances Below we define different stances that a tweet can take with respect to a rumour. The definitions align with or generalize the definitions from previous work (Qazvinian et al., 2011; Hamidian and Diab, 2015; Zeng et al., 2016b).

Supporting A supporting tweet expresses or suggests a belief that a rumour is true. It can provide a support for a rumour either by linking a supposedly factual content (a picture, a url to a story), by providing a description of how the author or their acquaintances witnessed the rumour, or by explaining why the story seems credible. The support can also be expressed simply by expressing the feelings that the story triggered in a user. An example supporting tweet is the tweet stating a rumour: "(...) there are two gunmen and a dozen hostages inside the cafe under siege at Sydney.. ISIS flags remain on display #7News", as shown in Figure 1.1.

Denying A denying tweet expresses a disbelief in a rumour. It can undermine its credibility, or explain why a user thinks it is not credible. A tweet can provide any kind of evidence of a similar nature to that listed in the supporting case, e.g. links to websites debunking a rumour, witnessing stories that undermine a rumour. An example denying tweet is "@u2 no she can't cos it's actually not", as shown in Figure 1.1. Notice how negation is used to express opinion about the rumour.

Questioning A questioning tweet (also referred to as querying (Zubiaga et al., 2017)) poses a question with respect to a rumour or expresses a doubt in its truth (however, does not explicitly reject it as in the case of the denying category). It is not as direct as tweets from previous categories. Questioning tweets are often replies to supporting or denying tweets. An example questioning tweet is "@u1 how do you know its an ISIS flag? Can you actually confirm that?", as shown in Figure 1.1. Notice how questioning words and question marks are denoting the stance of the tweet.

Commenting A commenting tweet does not exhibit any clear stance with respect to a rumour, or even tries to change the topic without taking any stance with respect to a rumour. An example questioning tweet is "More on situation at Martin Place in Sydney, AU `http://t.co/MWMQsw0IHB`", as shown in Figure 1.1. Notice how the tweet does not express any opinion about the rumour, but rather provides a link to news article talking about the whole event.

Motivation Previous work analysed stances expressed in tweets around rumours, and provided motivation as to why automatic rumour stance classification is an important task. Procter et al. (2013b) conducted an analysis of a large dataset of tweets related to riots in the UK that took place in August 2011. After grouping the tweets into topics, they were manually categorised into different classes, namely: 1. media reports, which are tweets sent by mainstream media accounts or journalists connected to media, 2. pictures, which are tweets containing a link to images, 3. rumours, which are tweets claiming or counter claiming something without giving any source, 4. reactions, which are tweets containing responses of users to the riots or specific events related to the riots. What is interesting for the purposes of our work is that the authors observed the following four-step pattern repeatedly occurring across the collected rumours: 1. a rumour is initiated by someone claiming it may be true, 2. a rumour spreads together with its reformulations, 3. counter claims appear, 4. a consensus emerges about the credibility of the rumour. This led the authors to the conclusion that the process of "inter-subjective sense making" by Twitter users plays a key role in exposing false rumours.

In another work, Mendoza et al. (2010) manually analysed the data from the earthquake in Chile in 2010. The authors selected 7 confirmed truths and 7 false rumours, each consisting of close to 1000 tweets or more. The veracity value of the selected stories was corroborated, and each tweet from each of the news items was manually classified regarding its stance into one of the categories: affirmation, denial, questioning, unknown or unrelated. The study showed that a much higher percentage of tweets about false rumours are shown to deny the respective rumours (approximately 50%). This is in contrast to rumours later proven to be true, where only 0.3% of tweets were denials. Based on this, the authors claimed that rumour veracity can be detected using aggregate analysis of the stance expressed in tweets. Moreover, Zubiaga et al. (2016c) described a study on temporal aspects of rumour propagation and support. The authors report a tendency for users to support true rumours more than false rumours as time passes. In summary, findings of Procter et al. (2013b), Mendoza et al. (2010) and Zubiaga et al. (2016b) provide motivation for research into automating stance classification, suggesting it to be a good indicator for rumour verac-

ity.

Number of classes In the initial work on rumour stance classification, Qazvinian et al. (2011) merged the denying and questioning classes into a single class, resulting in a binary classification problem of supporting vs denying-or-questioning. Moreover, tweets belonging to the commenting class were removed from the dataset, with the justification that it was a small number of tweets in the authors' dataset. Other works have also considered simplifying scenarios, with Hamidian and Diab (2015) and Zeng et al. (2016b) classifying tweets into supporting and denying classes. Narrowing down the number of classes makes the problem less realistic, because in real world applications it is not possible to reject tweets based on their label without classifying them first. Moreover, information about the distribution of all classes might bring more in-depth information than just about supporting and denying tweets.

Evaluation Initial work on rumour stance classification considered an evaluation of the task, where tweets from multiple rumours were gathered into a single dataset, which was then cross-validated to obtain the final results (Qazvinian et al., 2011). Thus, by pooling together tweets from all the rumours in their collections, both in training and test data, the separation of rumours was ignored.

Another problem with the evaluation from previous work is that temporal dependencies between tweets are ignored (Qazvinian et al., 2011; Zeng et al., 2016b). During cross-validation, later tweets may be used for training a classifier, which is then applied for classifying earlier tweets. This is an example of overfitting by "reprobleming": conducting cross-validation evaluation on time-series data ignores an important aspect of the problem. Similarly, ignoring rumour dependencies could be interpreted as overfitting by "reprobleming".[2]

Methods Different approaches have been considered for rumour stance classification. The seminal work by Qazvinian et al. (2011) considered logistic regression, and follow up work tried to outperform this method. Zeng et al. (2016b) explored the use of three different classifiers: Random Forests, Naive Bayes and Logistic Regression, and found Random Forests to perform the best.

Liu et al. (2015) introduced rule-based methods for stance classification, which were shown to outperform the approach by Qazvinian et al. (2011). Similarly, Zhao et al. (2015b) used regular expressions instead of an automated method for rumour stance classification.

[2]One reference to overfitting by "reprobleming" can be found at `http://hunch.net/?p=22`

The prior methods ignored an important characteristic of the rumour stance classification problem: the sequential nature of the task. After our seminal work (described in Chapter 7) on using a sequence classification for rumour stance classification was published as Lukasik et al. (2016b), further sequence classification methods have been explored for the task. Zubiaga et al. (2016a) employed a sequence classification approach based on Conditional Random Fields (CRFs), and demonstrated improvements over baselines (this paper was a result of a project collaboration, although it is not included as a contribution to the thesis). Similarly, Recurrent Neural Networks (RNNs) have been shown to be a competitive approach in the recent RumourEval shared task (Derczynski et al., 2017).

Features Various features have been considered in research on rumour stance classification. The seminal work considered text content from tweets, from which unigrams, bigrams, POS tags and URL counts were extracted, as well as user ids (Qazvinian et al., 2011). This set of features has been extended in the follow up work after Hamidian and Diab (2015), who additionally considered Twitter- and network-specific features and the so-called pragmatic features. As Twitter-specific features, the authors considered: hashtag strings from a tweet, whether a tweet is a re-tweet or a reply, and a time feature, which is a binary indicator whether a tweet has been posted during the manually-chosen five days. For the pragmatic features, the authors considered: sentiment label for a tweet coming from a deep neural network (Socher et al., 2013), emoticons, named entities, and events extracted using the tool for extracting those from Twitter (Ritter et al., 2011). Overall, the content features coming from Qazvinian et al. (2011) have been found to be the most useful, with network features bringing little improvement. Moreover, in their follow up work, Hamidian and Diab (2016) applied a latent variable model from Guo and Diab (2013) for modeling tweets in a vector space, and reported improvements over previously used features. In a different work, Zeng et al. (2016b) incorporated: n-gram features, part of speech tags, URLs, the Linguistic Inquiry and Word Count features (LIWC) (Tausczik and Pennebaker, 2010), and tweet sentiment features obtained using an external classifier. A work that followed ours from Zubiaga et al. (2016a) employed Twitter information involving conversation threads into the task.

It is worth noting that most of the work mentioned has been published after our first results (as described in Chapter 4) were published in Lukasik et al. (2015a). Moreover, we focus on investigating features of rumours not considered before: varying characteristics of stances expressed around different rumours (Lukasik et al., 2015a), and varying characteristics of rumours over time (Lukasik et al., 2016b).

Applications The rumour stance classification task is popular in the social media community due to its usefulness. The task can be used by journalists to automatically analyze rumours, and find particularly interesting rumours based on how they are considered by the community. Rumour stance classification has been considered in the PHEME project, where journalists benefited from the tools for analysing rumours (Derczynski et al., 2015).

Moreover, rumour stance classification has been shown to be useful for down-stream applications. Since it was shown that community stance about rumour veracity correlates with actual rumour veracity (Mendoza et al., 2010), rumour stance classification has been largely inspired by the rumour veracity classification task. Liu et al. (2015) explicitly applied rumour stance classification for classifying veracity of rumours, and concluded that tweet stances provide important information.

Researchers have shown how classifying the stance of tweets from an event can help determine whether it is a rumour or not (Zhao et al., 2015b; Ma et al., 2015). In particular, the questioning class has been conjectured to be very important for determining whether an event is a rumour. This is because uncertainty about rumour veracity is one characteristic of a rumour, and questioning tweets denote uncertainty in the community.

2.3 Modeling Temporal Dynamics of Rumours

There is a plethora of work on modelling the temporal nature of social media. However, there is not much work on modeling rumours specifically. Below we review previous work that has been done on both modeling meme and rumour popularity.

Meme Dynamics Modeling meme activity in social media is an active research field, with work dealing with various data sources and problems pertaining to them. Meme dynamics was quantitatively studied by Leskovec et al. (2009), where memes are defined as short phrases that travel through on-line text. In another work, Preotiuc-Pietro and Cohn (2013) modeled hash tag frequency in Twitter, predicting their popularity into the future. Gaussian processes have been shown to work well on this task, which we also employ in our experiments. Dzogang et al. (2016) analyzed seasonal fluctuations in collective mood expressed in tweets, and correlated them against Wikipedia searches over time.

Multiple works incorporate network information when modeling meme spread. For example, an Independent Cascade Model defines a spread of a meme over a network at discrete steps, and each infected node infects each of its neighbours with some edge dependent probability (Kempe et al., 2003). Other types of models have also been considered for modeling spread of memes over both time and network of connections, such as

Hawkes processes (Yang and Zha, 2013) and Cox Processes (Zhao et al., 2015a) (we discuss Hawkes processes in Section 3.2.5 and one type of a Cox Process in Section 3.2.4). A popular research direction is also network inference under observation of individual posts occurrences. In particular, Gomez Rodriguez et al. (2010) considered this problem under the cascade model, whereas Yang and Zha (2013) considered it under the Hawkes process model. In this thesis we do not consider network information, and instead focus on temporal and textual aspects of rumour propagation.

Another problem pertaining to information propagation in social media is maximization of meme visibility. This can be obtained through either the task of selecting the set of most influential users who would start the meme spread (Kempe et al., 2003), or through deciding on when the posting activity would happen so that it would be seen by the neighbours in the social network (Karimi et al., 2016; Zarezade et al., 2016). In our work, we do not consider the problem of controlling the activity, but rather try to predict it.

Submodular optimization has been used for tackling problems related to information diffusion in social media. For example, Kempe et al. (2003) considered selecting a subset of users in a social network that would allow for the widest spread of a meme. Leskovec et al. (2007) dealt with selecting a subset of blogs to monitor in order to maximize the exposure to memes spreading over the blogosphere. In another work, Gomez-Rodriguez and Schölkopf (2012) employed submodular optimization for network inference from observed cascades of information propagation.

Rumour Dynamics There have been several descriptive studies of rumours in social media regarding their dynamics. Procter et al. (2013a) analyzed rumours in tweets about the 2011 London riots and showed that they follow similar lifecycles, thus providing a cue that this task could be approached in a machine learning paradigm. Friggeri et al. (2014) showed how Facebook constitutes a rich source of rumours and conversation threads on the topic. The authors analyzed dynamics of rumour reshares and deletions of posts. Karlova and Fisher (2012) proposed a sociological model of information diffusion (however not formalized nor evaluated on real data), which tries to explain the process of misinformation and disinformation spread. In this diffusion model, before propagating the information piece, a receiver uses credibility or deception cues in order to filter the false information. It is also noticed, that cues to credibility are used by deceivers in order to hide their deception from the listener.

In another work, Nel et al. (2010) looked at information propagation across web pages, where referring from a web page to a source is a sign of propagation of information. The authors use various descriptors (e.g. the ratio between the number of sources and the number

of links), and use them to cluster events by their publishing behavior. Even though the motivation for their study is rumour detection, they do not experiment with rumours.

After our work on rumour dynamics was published as Lukasik et al. (2015b), Zeng et al. (2016a) considered modeling retransmission rate of tweets from rumours, but did not consider predicting rumour popularity into the future (which we do in Chapters 5 and 6).

2.4 Other Related Problems

Apart from rumour stance classification and rumour popularity modeling, other related applications have been considered in the literature. Below, we briefly review a few related problems.

Rumour Detection Rumour detection is the task of identifying rumours in a stream of social media posts. It can be thought of as clustering posts, followed by classifying each cluster into rumour and non-rumour categories. One setting for rumour detection is retrieving tweets around already known rumours (Qazvinian et al., 2011; Hamidian and Diab, 2015, 2016). This is useful for dealing with long-standing rumours, and can be called rumour tracking, since already known rumours are being followed (Zubiaga et al., 2017). Another setting is where new rumours are being detected from the social media stream. Zhao et al. (2015b) built a rumour detection system in Twitter based on tweets expressing uncertainty about a rumour, with five regular expressions (e.g. *Is this true?*) being crafted for identifying such tweets. Similarly, Vosoughi (2015) bases his rumour detection algorithm on speech act classification of tweets, with tweet categories being: *assertion*, *recommendation*, *expression*, *question*, *request*, *miscellaneous*. Zubiaga et al. (2016b) considered a different approach, where they classify tweets in a breaking news story into whether they constitute a rumour or not, not conducting explicit classification of tweets regarding their stance. The authors employed the Conditional Random Fields classification algorithm, and reported improvements over the method by Zhao et al. (2015b).

Rumour Veracity Classification Predicting rumour veracity is one of the main applications regarding rumours in social media (Derczynski et al., 2015). Rumour veracity classification can be defined as the task of classifying a rumour into whether it is true, false or unverified. There is an increasing interest in the scientific community in the problem of determining rumour veracity, as demonstrated, for example, by a special issue on rumours and social media (Papadopoulos et al., 2016). Middleton and Krivcovs (2016) described an approach for geoparsing social media posts in real-time, which can be of help to determine

the veracity of rumours by tracking down the poster's location. The contribution of Hamdi et al. (2016) to rumour resolution is to build an automated system that rates the level of trust of users in social media, hence enabling to get rid of users with low reputation. Castillo et al. (2011) considered classifying credibility of newsworthy events in Twitter. To this end, a system composed of two steps was developed: first, an event is classified as newsworthy (or not), and afterwards, its credibility is assessed. Rumour veracity classification has also been considered by Vosoughi (2015), who used linguistic features, propagation features and user id within the Hidden Markov Model, where the latent state at each time step denotes the temporarily perceived veracity.

Stance Classification Stance classification is the task of evaluating stances expressed in text regarding a given target. Apart from being considered in the context of tweets about rumours in social media, stance classification has been considered in other domains. Ferreira and Vlachos (2016) worked on stance classification of news articles with respect to rumours they report. Stance classification has also been considered in non-rumour applications, with examples of political leaning classification (Zhou et al., 2011) and debate stance classification (Sridhar et al., 2014; Hasan and Ng, 2013) for detection of agreement and disagreement. Mohammad et al. (2016) define stance detection as a three-way classification problem, where labels are: *positive*, *negative*, and *neutral*. The authors consider the Twitter domain, and define two SemEval Stance Detection tasks. In the first task, in the presence of labelled data about stance expressed in tweets around a few topics: *Climate Change is a Real Concern*, *Feminist Movement*, *Atheism*, *Legalization of Abortion* and *Hillary Clinton*, the task is to assign stances to test tweets from the same categories. In the second task, given annotation about stances in tweets about the above mentioned topics, predictions are made on a topic unobserved in training data: *Donald Trump*. Successful approaches employed neural networks and distant supervision for adding annotation in the absence of training data about the target concept (Augenstein et al., 2016). The task differs from the rumour stance classification in that rumours are typically of smaller volume (as shown by our datasets described in Section 2.5), and have been shown to exhibit varying proportions of stances over short periods of time (Procter et al., 2013a), characteristics which are not necessarily shared with other social media events in general. Nevertheless, similar techniques may be used for addressing the rumour stance classification problem in different domains. In particular, we demonstrate usefulness of a sequence classification approach in Chapter 7.

Fact Checking Fact checking is the task of evaluating truthfulness of claims expressed in natural language, usually made by public figures. Vlachos and Riedel (2014) defined the

task in open domain settings, not restricting it to any particular source of information (such as social media). Fact checking is not necessarily equivalent to rumour veracity classification. First, the claims are not necessarily rumours, as they do not necessarily attract a large interest from the community, or disagreements about their truthfulness. Vlachos and Riedel (2015) limited the scope to 16 numerical properties of countries (such as population), and employed distant supervision for identification and verification of claims. In the follow-up work, the system has been extended to include temporal expressions, so that the temporal context of the claim could be taken into account (Thorne and Vlachos, 2017). In another branch of work, researchers proposed methods for detecting facts from the data. Hassan et al. (2014) proposed a tool for extracting and monitoring facts in the data stream. In another work, Hassan et al. (2015) considered the problem of classifying sentences into factual and non-factual, allowing journalists to focus on the check-worthy sentences.

Modeling Disinformation Disinformation is defined by the English Oxford Dictionary (EOD) as: "deliberately false information". Karlova and Fisher (2012) note that disinformation is very often viewed in the literature to be a type of misinformation (English Oxford Dictionary defines misinformation as "wrong or misleading information"). Gupta et al. (2013) study verbal deception (defined by the authors as a disinformation happening during a discourse between two entities) and emphasize, that what is false in the context of disinformation is defined by the state of beliefs of the speaker, rather then what is objectively true. Therefore, the common sense definition of truth does not play a big role here. This is different than in the case of misinformation. As for the studies on modeling disinformation, Fornaciari et al. (2013) analyzed the personality type of an author as a feature for deception detection. The authors base their personality analysis on the Big5 traits (Norman, 1963): *extraversion*, *emotional stability/neuroticism*, *agreeableness*, *conscientiousness*, *openness to experience*, all of which can be either quantitatively measured on a continuous scale from -1 to 1 or on nominal scale Y, O, N. The authors demonstrated the usefulness of personality cues for misinformation detection by outperforming baselines, and conducted clustering of user personality types. In another study, Ratkiewicz et al. (2011) considered the task of detecting political astroturf (practice of masking the sponsors of posts, which is a form of disinformation through pretending to be independent) in Twitter. The authors employed sentiment features as well as many network features (such as number of users and edges between them, or the mean size of connected components in the user graph). Ratkiewicz et al. (2011) concluded that network features are more discriminative than sentiment features for the task of detecting political astroturf.

Rumour	#Tweets	#Supports	#Denies	#Questions
Army bank	177	62	42	73
Children's hospital	1415	796	487	132
London Eye	632	177	295	160
McDonald's	190	177	0	13
Miss Selfridge's	3157	3150	0	7
Police beat girl	796	783	4	95
London zoo	844	616	129	99
Total	7297	5761	957	579

Table 2.1: Counts of tweets with supporting, denying or questioning labels in each rumour collection from the England riots dataset.

2.5 Rumour Datasets

In this section we review the rumour datasets in the literature. We focus on two datasets that we make use of in this thesis: the England riots datasets introduced by Procter et al. (2013b) and the PHEME datasets introduced by Zubiaga et al. (2016c), and follow with an overview of other datasets.

2.5.1 England Riots Dataset

The England riots dataset consists of seven rumours circulating on Twitter during the England riots in 2011 (see Table 2.1). The rumours were as follows (Procter et al., 2013b):

- The Army was being mobilised in London to deal with the rioters (*Army bank*).
- Rioters were gathering to attack Birmingham's Children's Hospital (*Children's hospital*).
- Rioters had set the London Eye on fire (*London Eye*).
- Rioters had broken into a McDonalds and set about cooking their own food (*McDonald's*).
- A store belonging to the Miss Selfridge retail group had been set on fire in Manchester (*Miss Selfridge's*).
- Police had beaten a sixteen year old girl (*Police beat girl*).
- Rioters had attacked London Zoo and released the animals (*London zoo*).

	position
Children's hospital	
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
Army bank	
Is it true the army has assembled at bank and they are now looting London zoo??? Seriously wtf #londonriots	support
Are you hearing news of the army preparing at Bank? #londonriots	question
In my defence I was looking at the pics on my phone! So once and for THE ARMY IS NOT IN BANK. #londonriots	deny
London Eye	
breaking news: rioters have pushed the london eye over #Londonriots	support
Oh my god! This can't be happening at London Eye! #Londonriots #Londonriot #Prayforlondon `http://twitpic.com/6372vo`	support
WTF ?1 is the londoneye really on fire ? #londonriots `http://t.co/EpPZcwR`	question
RT @user286: 02:28 - No reports of the London Eye being set alight. #LondonRiots	deny

Table 2.2: Tweets pertaining to rumours about: *Children's hospital*, *Army bank* and *London Eye* during 2011 England Riots.

	position
McDonald's	
RT @user287: teenagers in london bumrush mcdonalds to cook they're own food? #riotswag	support
RT @user288: @user289: Shop looted and youths storm McDonaldž019s and start cooking their own food	support
Daily Mail reporting ppl cooking their own McDonald's during the #tottenham riot. Can't your wack reporters find more meaningful details?	question
Miss Selfridge's	
RT @user289: Man who sets fire to Miss Selfridge during #manchesterriots has his home set on fire. `http://t.co/AbxhdCY`	support
This is the dickhead that set fire to Miss Selfridges! #NameAndShame #ManchesterRiots `http://bit.ly/o67BgS`; how could you!	support
London zoo	
RT @user293 #tottenham Monkeys in a zoo. Get London Zoo in to round 'em up. Put them on show and let's have at them. Idiots	support
RT @user294: #londonriots oh my god - reports of tigers roaming around Primrose Hill #londonzoobreakin `http://t.co/j2DjbOZ`	support
Can anyone confirm/dispel rumors about zoo animals on the loose at #Londonriots ? I'm prepared for it to be lies...but on the off chance...	question
Just heard a tiger's been released from London Zoo - but then heard that it's not true `http://t.co/QGDxIPI` #londonriots #zoo via @user295	deny

Table 2.3: Tweets pertaining to rumours about: *McDonald's*, *Miss Selfridge's* and *London zoo* during 2011 England Riots.

	position
Police beat girl	
RT @user290: Is anyone ever gna show the 16 year old girl getting beaten by the police in tottenham? #riotsdebate	question
RT @user291: Police attacking 16-year-old girl in Tottenham, setting the whole thing off. http://ow.ly/5Yy6y #londonriots #ukriots	support
RT @user292: 16 YEAR OLD GIRL BATTERED BY #TOTTENHAM RIOT POLICE - THIS IS WHAT STARTED IT ALL!: http://tumblr.com/xjh3yqwfr3	support
#Tottenham; Is this the footage of 16yr old girl beaten by police? http://t.co/ZFD8TPk NSFW. Looks like about 10 cops setting on someone	question

Table 2.4: Tweets pertaining to rumours about *Police beat girl* during 2011 England Riots.

The dataset was collected by tracking a long set of keywords associated with the event. The dataset was analysed and annotated manually as supporting, questioning, or denying a rumour, by a team of social scientists studying the role of social media during the riots (Procter et al., 2013b). As can be seen from the dataset overview in Table 2.1, different rumours exhibit varying proportions of supporting, denying and questioning tweets, which was also observed in other studies of rumours (Mendoza et al., 2010; Qazvinian et al., 2011). These variations in the number of tweets for each class across rumours poses a challenge when building a model of rumour stances.

In tables 2.2, 2.3 and 2.4 we report sample tweets from the England riots rumours together with their stances. Each rumour is initiated by a tweet stating the rumour (in Tables 2.2 to 2.3 the first tweet we display for each rumour is the initiating tweet). Notice how supporting tweets often exhibit strong emotional reaction, usually negative, expressed by words such as "F***ing" in the case of the *Army bank* rumour, and "wtf" in the case of the *London Eye* rumour. Questioning tweets express doubt about rumour veracity, with example phrases like "really?" for the *Children's hospital* rumour, "can anyone confirm" for the *London zoo* rumour, "is this the footage of" for the *Police beat girl* rumour. Denying tweets often contain words expressing negation, e.g. in "no reports of" for the *London Eye* rumour or "not true" for the *London zoo* rumour.[3]

[3]In the later chapters we use Brown clusters trained on a large scale Twitter corpus as one approach to representing text. Negating words are clustered together, as shown in the on-line summary of the Brown clusters at http://www.cs.cmu.edu/~ark/TweetNLP/cluster_viewer.html (e.g. cluster 001000).

This dataset does not contain all the tweets pertaining to these rumours. It is therefore not suitable for rumour popularity modeling, because, when predicting popularity into the future, for reliable evaluation one needs access to information about how many tweets occurred in total (see Section 5.4 for description of our evaluation of rumour popularity prediction). The next dataset that we consider is composed of fully observed rumours, which is partially due to how rumours are defined, and thus is applicable for rumour popularity modeling.

2.5.2 PHEME Dataset

The PHEME dataset we make use of is associated with five different events and was collected as part of the PHEME FP7 research project. The dataset is described in detail in Zubiaga et al. (2016c, 2015b). In contrast to the England riots dataset, rather than collecting tweets from the whole event and then clustering tweets into different rumours, the PHEME datasets were collected by tracking conversations initiated by rumour tweets. This was done by first collecting tweets containing a set of keywords associated with a story unfolding in the news. Next, the most retweeted tweets were selected, and conversations around them were collected. By conversations we mean tweets that are in the conversation tree where a node is a seed tweet, and edges are reply-to actions. Notice that in the England riots dataset replying tweets might be missing, whereas the PHEME dataset comprises replying tweets by definition. In Table 2.5 we report the basic statistics about the five largest events from the PHEME dataset.

A subset of the PHEME rumour datasets have been annotated for stance (Zubiaga et al., 2016c). While the authors annotated and released 9 datasets, here we make use of the 5 largest datasets. This dataset includes tweets associated with the following five events, each of which is comprised of multiple rumours:

- **Ferguson unrest:** Citizens of Ferguson (USA) protested after the fatal shooting of an 18-year-old African American, Michael Brown, by a white police officer on August 9, 2014. Example rumours around this event are: Fox News was not reporting Ferguson riots at the time, and police not allowing anybody to enter Ferguson from the outside.

- **Ottawa shooting:** Shootings occurred on Ottawa's Parliament Hill in Canada, resulting in the death of a Canadian soldier on October 22, 2014. Example rumours around this event are: a soldier being fatally shot, and the soldier who was killed having a six-year-old son.

- **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. Example rumours around this event are: the gunman or the hostages making contact with the media, and hostages escaping from the cafe.

- **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. Example rumours around this event are: the suspects saying they want to die as martyrs, and later on about the suspect being dead.

- **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. Example rumours around this event are: the co-pilot suffering from depression, and the co-pilot being a convert to Islam.

Zubiaga et al. (2016c) relied on a different scheme for the tweet annotation for stances than the one used by Procter et al. (2013b). Authors annotated tree-structured conversation threads where a source tweet initiates a rumour and a number of replies follow responding to it. Given this structure, the source tweet of a Twitter conversation is annotated as *supporting*, *denying* or *underspecified*, and each subsequent tweet is annotated as *agreed*, *disagreed*, *appeal for more information* (*questioning*) or *commenting* with respect to the source tweet (notice this annotation is not with respect to the previous tweet in the conversation, but only the first tweet posted about the rumour, which we call the source tweet). In order to align stance annotations to the ones from the England riots dataset, we convert these labels into the four including *supporting*, *denying*, *questioning* and *commenting*. To perform this conversion, we first remove rumours where the source tweet is annotated as *underspecified*[4], keeping the rest of the source tweets as *supporting* or *denying*. For the subsequent tweets, we keep their label as is for the tweets that are *questioning* or *commenting*. To convert those tweets that agree or disagree into *supporting* or *denying*, we apply the following set of rules: (1) if a tweet agrees to a supporting source tweet, we label it *supporting*, (2) if a tweet agrees to a denying source tweet, we label it *denying*, (3) if a tweet disagrees to a supporting source tweet, we label it *denying* and (4) if a tweet disagrees to a denying tweet, we label it *supporting*. We summarise the details of the datasets and frequencies of different stances in Table 2.6. Note that the *commenting* label accounts for the majority of the tweets. Table 2.7 shows tweets from example rumours together with the derived annotations. Overall, similar characteristics can be observed as in the case of England

[4]It is not clear what a supporting or denying tweet with respect to an underspecified tweet should be labeled as.

Dataset	#Rumours	#Tweets	$\frac{\#\text{Tweets}}{\#\text{Rumours}}$
Ottawa	475	6021	12.68
Ferguson riots	291	6334	21.77
Charlie Hebdo	458	6397	13.97
Sydney siege	522	7632	14.62
Germanwings	238	2018	8.48

Table 2.5: Statistics of rumours from the PHEME datasets (Zubiaga et al., 2016c).

Dataset	#Rumours	#Tweets	#Supports	#Denies	#Questions	#Comments
Ottawa	58	782	161	76	64	481
Ferguson riots	46	1017	161	82	94	680
Charlie Hebdo	74	1053	236	56	51	710
Sydney siege	71	1124	89	223	99	713
Germanwings	68	386	177	12	28	169

Table 2.6: Statistics and distribution of labels for the annotated subset of PHEME rumour datasets (Zubiaga et al., 2016c). Each dataset consists of a number of rumours (reported in the first column). The leftmost columns show the aggregated counts of tweets expressing different stances for all rumours within a dataset.

riots, with negation words in denying tweets (e.g. "not" in the Sydney Siege rumour), and question words used in questioning tweets (e.g. "how do you know" in the Sydney Siege rumour). Notice how the commenting tweets do not express any direct opinion about the rumour.

As mentioned, contrary to the England riots dataset, the PHEME dataset comprises complete conversation threads within the observed window of time. This dataset is therefore more reliable than England riots with respect to containing all tweets pertaining to a rumour, and thus better suited to modeling rumour dynamics over time, as there are no missing tweets between those that are observed. In Figure 2.1 we depict several example rumour profiles from the Ferguson riots dataset. Notice that although there is a rich diversity of rumour profiles, several similarities are evident such as several rumours with very high frequency at their beginning (e.g., from initial reporting or lively discussion), and most rumours having unimodal frequency profiles (e.g., only capturing community attention briefly before fading away).

	position
Sydney Siege: ISIS flags remain on display during the Sydney Siege	
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	support
@u1 sorry - how do you know it's an ISIS flag? Can you actually confirm that?	question
Have you actually confirmed its an ISIS flag or are you talking sh*t	question
@u2 no she can't cos it's actually not	deny
Ottawa shooting: Soldiers are back the same day guarding the war memorial after the Ottawa shooting incident	
These are not timid colours; soldiers back guarding Tomb of Unknown Soldier after today's shooting #StandforCanada –PICTURE–	support
@u1 This photo was taken this morning, before the shooting.	deny
@u1 More on situation at Martin Place in Sydney, AU –LINK–	comment
Charlie Hebdo: The gunmen said: *You tell the media it was al-Qaeda in Yemen*	
Witnesses say the Charlie Hebdo gunmen identified themselves as members of al-Qaida: `http:\/\/t.co\/WSEe7PGIdY` `http:\/\/t.co\/fTzk4xzZIr`	support
@u3 witnesses may have but President never will not will he utter #islam #jihad or #muslim in association with this!	comment
I strongly condemn this terrorist attack as a Turkish Muslim.Terrorists Cannot represent Islam #notinmyname	support

Table 2.7: Tweets pertaining to example PHEME rumours.

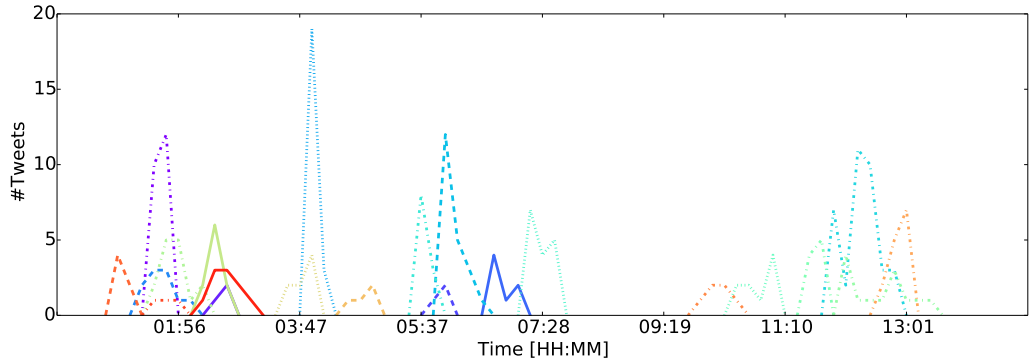


Figure 2.1: Time profiles of several example rumours in the Ferguson data set happening during 13/8/2013.

2.5.3 Other Rumour Datasets

Related work reports experiments on multiple other rumour datasets. Qazvinian et al. (2011) gathered tweets from five long standing rumours and annotated them for stances with respect to their veracity. Thus the characteristics of the data are quite different from the ones listed above (instead of multiple short rumours, Qazvinian et al. (2011) considered a few long rumours). The commenting label has been removed from the dataset before conducting the experiments, and rejecting and questioning stances were collapsed together. The dataset has been used in following works (Hamidian and Diab, 2015, 2016), however it is not publicly available. Zeng et al. (2016b) gathered five rumours around Sydney siege in December 2014. Data has been collected using curated sets of keywords and hashtags, and annotated for support, deny and neutral classes. The neutral class has been removed from the analysis, leaving 2906 supporting and 1469 denying tweets. The dataset is not publicly available. Friggeri et al. (2014) gathered several thousand rumours from Facebook. Rumours were assigned a veracity category based on the Snopes website annotations.[5] The dataset has not been annotated for stance classification, and has not been publicly released. Vosoughi (2015) gathered 938806 tweets coming from: 2013 Boston Marathon bombings, the 2014 Ferguson unrest and the 2014 Ebola epidemic (plus other less prevalent sources), and used the dataset to experiment with rumour detection and rumour veracity classification. The dataset is not publicly available. Ferreira and Vlachos (2016) introduced a dataset of rumours described by on-line news articles. Data is gathered from the website `emergent.info`, which provides rumours collected by journalists. The dataset provides stance label of each article with respect to veracity of a rumour. The stances are: supporting,

[5] `www.snopes.com`

denying and neutral. The dataset also provides veracity of each rumour (if journalists were able to resolve it, otherwise the annotation denotes that a rumour is unsubstantiated). Even though interesting, this dataset is beyond the scope of this thesis, as we focus on rumours discussed in social media rather than in the news articles.

2.6 Conclusions

In this chapter we reviewed the literature on rumours in social media, and discussed two applications: rumour stance classification and rumour popularity prediction. We found that the previous state of research on rumour stance classification fails to consider a number of aspects of the task. First and foremost, its evaluation is not realistic, as the way the cross-validation folds were chosen ignores the temporal ordering of tweets by drawing tweets uniformly across time. Additionally, the cross-validation was conducted uniformly across tweets pooled together from multiple rumours, meaning the same rumour could appear in multiple folds. Moreover, a limited number of stances were considered in the experiments, by either ignoring some of the stances or collapsing different stances together. Also, features such as time and the rumour a tweet pertains to were not used. As for the rumour popularity prediction task, little work has been done in this direction. In this thesis, we aim to define this problem and consider appropriate models. In the next chapter we introduce the machine learning methods that we make use of, when tackling the rumour stance classification and rumour popularity prediction problems.

Chapter 3

Probabilistic Models for Classification and Temporal Modeling

The previous chapter reviewed literature on rumours in social media, and motivated research in two directions: rumour stance classification and rumour popularity prediction. In this chapter we introduce and motivate the machine learning approaches, namely Gaussian processes and point processes, which we use for approaching the aforementioned rumour applications.

3.1 Gaussian Processes

In this section we describe the probabilistic model we use for solving both problems: Gaussian processes (GPs). Not only are they flexible enough to allow for approaching both applications, but they also has multiple properties which lead to it outperforming strong baselines, as we show in the experimental chapters. Our discussion and notation follows that from Rasmussen and Williams (2005). Recall our notation is summarized in the Nomenclature on page xii.

3.1.1 Motivation

Gaussian processes are a Bayesian non-parametric machine learning framework that have been shown to work well for a range of NLP and social media 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). They are most widely used for regression; however, as we are going to show, they are flexible enough to be used for other problems, including classification, Poisson regression and point process modeling.

Gaussian processes exhibit many useful properties which make them appealing. As a non-parametric model, we don't need to specify a fixed parametric form of the relationship between the outputs and the inputs (in our applications it is not clear what parametric families could be appropriate, as we can see on an example of rumour frequencies in Figure 2.1). This probabilistic kernelised framework avoids the need for expensive cross-validation for hyperparameter selection,[1] instead providing a way of directly optimizing the hyperparameters by maximizing the marginal likelihood of the data. See Section 3.1.5 for the discussion of how the marginal likelihood can be used for optimizing kernel hyperparameters in the GPs framework. GPs average over many predictive distributions instead of taking a single prediction, which helps avoid overfitting. Also, GPs provide information about uncertainty about the predictions, which allows a user to decide whether a prediction is reliable. Even though we do not use the uncertainty information in this work, we demonstrate that mean predictions from GPs often work better than the baselines, and in future work the uncertainty information could be used as an additional source of information.

3.1.2 Model

We are interested in modeling outputs $\mathbf{y} = \{y_1, \ldots, y_N\}$ over inputs $X = \{\mathbf{x_1}, \ldots, \mathbf{x_N}\}$, where the output y_i corresponds to the input $\mathbf{x_i}$. We can consider different kinds of outputs y_i, e.g. in the case of classification y_i is a categorical scalar value (e.g. does a tweet $\mathbf{x_i}$ support a rumour or not?), in the case of regression y_i is a real scalar value (e.g. a stock price corresponding to a Twitter discussion represented as $\mathbf{x_i}$), and in the case of Poisson regression y_i is a non-negative integer (e.g. how many re-tweets of a tweet $\mathbf{x_i}$ are going to occur?). Similarly, different types of inputs $\mathbf{x_i}$ may be considered (e.g. vectors representing textual information and scalar values representing time).

A Gaussian process specifies a distribution over functions $f(\mathbf{x}) \sim \mathcal{GP}\left(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')\right)$ defined over inputs $\mathbf{x}$, where $m(\mathbf{x})$ is the mean function specifying the expected value of function f at argument $\mathbf{x}$, $m(\mathbf{x}) = \mathbb{E}[f(\mathbf{x})]$, and $k(\mathbf{x}, \mathbf{x}')$ is the kernel function, specifying the covariance between a pair of arguments, $k(\mathbf{x}, \mathbf{x}') = \mathbb{E}[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}') - m(\mathbf{x}'))]$. A GP prior states that the joint distribution of the function outputs $\{f_1 = f(\mathbf{x_1}), \ldots, f_N = f(\mathbf{x_N})\}$ corresponding to the finite set of inputs $\{\mathbf{x_1}, \ldots, \mathbf{x_N}\}$ is a multi-variate Normal

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

distribution,

$$\mathbf{f} \sim \mathcal{N}\left(\mu{=}m(X), \Sigma = K(X,X)\right), \tag{3.1}$$

where

$$m(X) = \begin{bmatrix} m(\mathbf{x_1}) \\ \dots \\ m(\mathbf{x_N}) \end{bmatrix} \tag{3.2}$$

denotes the mean function evaluated at the inputs, and

$$K(X,X) = \begin{bmatrix} k(\mathbf{x_1},\mathbf{x_1}) & \dots & k(\mathbf{x_N},\mathbf{x_1}) \\ \dots & \dots & \dots \\ k(\mathbf{x_N},\mathbf{x_1}) & \dots & k(\mathbf{x_N},\mathbf{x_N}) \end{bmatrix} \tag{3.3}$$

denotes *the Gram matrix*, i.e., the matrix of kernel evaluations between every pair of inputs. Consequently, the joint distribution of both the latent function values $\mathbf{f}$ at the training inputs X, and the latent function values $\mathbf{f}^* = \{f_1^*, \dots, f_M^*\}$ at the test inputs $X^* = \{\mathbf{x_1^*}, \dots, \mathbf{x_M^*}\}$ is

$$\begin{bmatrix} \mathbf{f} \\ \mathbf{f}^* \end{bmatrix} \sim \mathcal{N}\left(\mu{=}\begin{bmatrix} m(X) \\ m(X^*) \end{bmatrix}, \Sigma{=}\begin{bmatrix} K(X,X) & K(X,X^*) \\ K(X,X^*)^\top & K(X^*,X^*) \end{bmatrix}\right), \tag{3.4}$$

where $K(X,X^*)$ denotes the matrix of kernel values between the training data points (corresponding to rows) and the test data points (corresponding to columns), $K(X^*,X^*)$ denotes the matrix of kernel values between the test data points and $K(X,X)$ denotes the matrix of kernel values between the training data points.

A Gaussian process can be thought of as specifying a distribution $p(\mathbf{f}|X)$ over multi-dimensional objects $\mathbf{f} = \{f(\mathbf{x_1}), \dots, f(\mathbf{x_N})\}$ conditioned on inputs $X = \{\mathbf{x_1}, \dots, \mathbf{x_N}\}$. We call the distribution $p(\mathbf{f}|X)$ the Gaussian process prior. The GP prior is used together with a likelihood of outputs $\mathbf{y}$ conditioned on the latent function values $\mathbf{f}$, $p(\mathbf{y}|\mathbf{f})$, to explain the outputs $\mathbf{y}$ conditioned on the inputs X (different likelihoods may be useful depending on the type of the outputs, as we discuss later in Section 3.1.3). In Figure 3.1 we illustrate how $\mathbf{x}$, $\mathbf{f}$ and $\mathbf{y}$ relate to one another within the GP framework.

Mean function The mean function conveys information about what values a latent function should take in expectation. One could encode their prior knowledge about the reasonable values a function could take. In absence of such knowledge, typically the mean function is set to $m(\mathbf{x}) = \mathbf{0}$ (Bishop, 2006; Preotiuc-Pietro, 2014; Beck, 2017), which is the approach we take in this thesis.

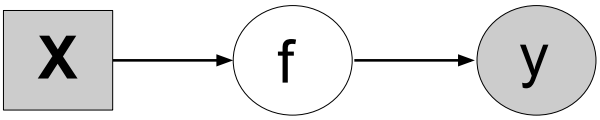


Figure 3.1: A graphical representation of the relation between the input $\mathbf{x}$, the latent function value f, and the output y in the Gaussian process framework. Both the input $\mathbf{x}$ and the output y are observed (denoted by the colour grey), whereas the function value f is latent (denoted by the colour white). Random variables modeled by the GP are denoted by circles, and the input (which is not modeled) is denoted by the square. $p(f|\mathbf{x})$ is modeled using a GP prior, whereas $p(y|f)$ is modeled using a likelihood function.

Kernel function The kernel function encodes the prior information by specifying how the outputs covary as a function of the inputs. As shown in Equation (3.1) a kernel specifies the covariance matrix, which in turn controls how the different dimensions in the multivariate Normal distribution are correlated. Depending on what input representation is used and what assumptions about the dependencies between function outputs at different inputs are made, different kernel functions may be appropriate. We discuss kernel functions in Section 3.1.5.

Likelihood and posterior The Gaussian process prior $p(\mathbf{f}|X)$ combined with the likelihood $p(\mathbf{y}|\mathbf{f})$, i.e., the probability of the outputs conditioned on the latent function values $\mathbf{f}$, gives rise to the posterior distribution of the function values, $p(\mathbf{f}|\mathbf{y}, X)$. Notice that the likelihood is independent of the inputs X, $p(\mathbf{y}|\mathbf{f}, X) = p(\mathbf{y}|\mathbf{f})$. The likelihood explains how the latent function values $\mathbf{f}$ lead to the observed output values $\mathbf{y}$ by encoding an assumption about how the observations are generated from the modeled latent function (in the case of regression, $p(\mathbf{y}|\mathbf{f})$ is often assumed to be a Gaussian distribution; we discuss different likelihoods and their role in Section 3.1.3). The GP posterior over the training inputs can be found by combining the likelihood and the prior using Bayes rule,

$$p(\mathbf{f}|\mathbf{y}, X) = \frac{p(\mathbf{y}|\mathbf{f})p(\mathbf{f}|X)}{p(\mathbf{y}|X)}.$$

Depending on the assumed likelihood, the posterior may or may not be found in a closed form. In Section 3.1.3 we discuss different choices of likelihoods and their impact on the form of the posterior distribution $p(\mathbf{f}|\mathbf{y}, X)$.

The posterior over the latent function values at train inputs can be used to compute the

predictive distribution of the latent function value at a test input,

$$p(f^*|X, \mathbf{y}, \mathbf{x}^*) = \int p(f^*|X, \mathbf{x}^*, \mathbf{f}) p(\mathbf{f}|\mathbf{y}, X) d\mathbf{f}, \tag{3.5}$$

where $f^* = f(\mathbf{x}^*)$. The latent function value at a test input $\mathbf{x}^*$ can then be used for finding the predictive distribution over the test output y^*,

$$p(y^*|X, \mathbf{y}, \mathbf{x}^*) = \int p(y^*|f^*) p(f^*|X, \mathbf{y}, \mathbf{x}^*) df^*, \tag{3.6}$$

where $p(y^*|f^*)$ is the likelihood of the test output y^* conditioned on the latent function value f^* modeled over the test input $\mathbf{x}^*$. Gaussian processes can be applied to a range of problems, which may require employing different likelihoods. In this thesis we apply GPs to four problem types, namely regression, classification, Poisson regression and point process modeling. Depending on the type of a problem, the distributions from Equations (3.5) and (3.6) can take different forms. In the case when the prior and the likelihood are conjugate (as in the case of regression) both can be found in a closed form. Otherwise, one needs to resort to approximation techniques for solving the integrals in Equations (3.5) and (3.6). The predictive distribution over the output y^* from Equation (3.6) involves a one-dimensional integral over the latent function value f^*, and so in general simple numerical techniques are sufficient (Rasmussen and Williams, 2005), such as numerical quadrature or Monte Carlo sampling. The integral from Equation (3.5) is over multiple dimensions of the latent function values $\mathbf{f}$, and so is more challenging. In Section 3.1.4 we describe methods that can be used to approximate the posterior distribution $p(\mathbf{f}|\mathbf{y}, X)$ as a Normal distribution, leading to tractable computations.

Below we explain how GPs can be applied to regression, classification and Poisson regression, whereas the application of GPs to point process modeling is described in Section 3.2.

3.1.3 Outputs

Depending on what output is being modeled, and consequently what likelihood one chooses, inference may lead to a closed form solution or require approximations. Here we review three types of outputs and how GPs may be applied to them: regression, classification and Poisson regression. For each output type we specify an appropriate form of the likelihood of observations given the latent function values $p(\mathbf{y}|\mathbf{f})$ (however, different likelihoods may be applicable for the same output type, e.g. the t-Student distribution can be used instead of the Normal distribution for regression outputs, leading to more robust predictions (Rasmussen

and Williams, 2005)), and discuss the inference.

Regression In single-output regression problems a continuous scalar value is being modeled over the input space. Example regression problems are predicting the future stock price of a company based on indicators describing past stock movement (Bitvai and Cohn, 2015b), or predicting continuous emotion scores from text (Beck et al., 2014). In the case of regression, the likelihood is a Gaussian, $p(\mathbf{y}|\mathbf{f}) = \mathcal{N}(\mathbf{f}, \rho I)$, where ρ denotes the variance of the Gaussian likelihood, and can be learnt together with kernel hyperparameters by maximizing the evidence $p(\mathbf{y}|X)$ (for more details on optimizing hyperparameters via evidence maximization see Section 3.1.5).[2] The Gaussian distribution is a conjugate prior for this likelihood function, meaning that the resulting posterior distribution is also Gaussian (Rogers and Girolami, 2012). Therefore, the predictive distribution from Equation (3.6) can be expressed in a closed form as a Gaussian $p(y^*|X, \mathbf{y}, \mathbf{x}^*) = \mathcal{N}(y^*|\mu^*, \sigma^*)$ with the mean given by

$$\mu^* = K(\mathbf{x}^*, X)(K(X, X) + \rho I)^{-1}\mathbf{y}, \tag{3.7}$$

and the variance expressed by

$$\sigma^* = k(\mathbf{x}^*, \mathbf{x}^*) - K(\mathbf{x}^*, X)(K(X, X) + \rho I)^{-1}K(\mathbf{x}^*, X)^\top. \tag{3.8}$$

Classification In single-output classification problems a categorical variable is modeled over the input space. Example classification problems are predicting occupational class of a user based on their Twitter posts (Preotiuc-Pietro et al., 2015), predicting stance of a tweet with respect to a rumour it is discussing (Qazvinian et al., 2011), and classifying gender based on an image of a face (Jia et al., 2016). Here we consider GPs for binary classification, the special case of classification where only two classes are considered. One way of modeling multi-class classification is by learning multiple one vs. all binary classifiers.[3]

In binary classification the latent function is mapped by the squashing function $\Phi(f)$ (e.g. probit or logit) into the range $[0, 1]$, such that the resulting value can be interpreted as the probability of the positive class, $p(y = 1|\mathbf{x})$. The likelihood of N outputs $\mathbf{y} = \{y_1, \ldots, y_N\}$ given the corresponding N latent function values $\mathbf{f} = \{f_1, \ldots, f_N\}$ is equal to

$$p(\mathbf{y}|\mathbf{f}) = \prod_{n=1}^{N} \Phi(f_n)^{y_n} \left(1 - \Phi(f_n)\right)^{1-y_n}, \tag{3.9}$$

[2] Notice the assumption of homoscedastic noise, meaning that the noise is independent of $\mathbf{f}$. In principle, one could introduce dependence of noise on $\mathbf{f}$.

[3] Another approach could be to directly model the multiple classes with the soft-max likelihood function.

the product of Bernoulli likelihoods of classes $y_1, \ldots, y_N$. The posterior distribution over the latent function value from Equation (3.5) is intractable due to non-conjugacy of the Normal distribution and the Bernoulli likelihood.

Poisson regression Poisson regression is a problem where count outputs are predicted. Example applications of Poisson regression are predicting the number of tweets in a fixed time interval or predicting the number of casualties of an earthquake event. For Poisson regression, the likelihood of outputs given the latent function values is

$$p(\mathbf{y}|\mathbf{f}) = \prod_{n=1}^{N} \text{Poisson}\left(y_n|\exp(f_n)\right), \tag{3.10}$$

where the Poisson distribution is given by

$$\text{Poisson}(y|\lambda) = \frac{\lambda^y \exp(-\lambda)}{y!}. \tag{3.11}$$

The exponential in the likelihood in Equation (3.10) is used in order to enforce a non-negative parameter value. As in the case of classification, the posterior distribution from Equation (3.5) is intractable due to the non-conjugacy of the Normal distribution and the Poisson likelihood. Next, we describe approximation techniques which can be used for obtaining the distribution $p(f^*|X, \mathbf{y}, \mathbf{x}^*)$.

3.1.4 Approximate Inference

The integral from Equation (3.5) is intractable in the classification and Poisson regression settings. One method for dealing with this problem is approximating the posterior distribution $p(\mathbf{f}|\mathbf{y}, X)$ as a Gaussian, this way leading to a tractable solution of Equation (3.5).[4] In this thesis we make use of two techniques: Laplace approximation and Expectation Propagation, both of which yield a Gaussian posterior, making the computations tractable and leading to an analytical solution to the integral Equation (3.5).

Laplace approximation approximates the GP posterior $p(\mathbf{f}|\mathbf{y}, X)$ by a Gaussian distribution $q(\mathbf{f}|\mathbf{y}, X)$ based on the first and the second derivative of the logarithm of the unnormalized posterior (Rasmussen and Williams, 2005). This is obtained by a Taylor expansion of the logarithm of unnormalized posterior $\log p(\mathbf{f}|X, \mathbf{y})$ around its mode, obtaining the

[4] Alternative solutions exist, such as solving the integrals via sampling (Nickisch and Rasmussen, 2008).

approximation:

$$q(\mathbf{f}|\mathbf{y}, X) = \mathcal{N}(\mathbf{f}|\hat{\mathbf{f}}, A^{-1}), \tag{3.12}$$

where

$$\hat{\mathbf{f}} = \arg\max_{\mathbf{f}} \log p(\mathbf{f}|\mathbf{y}, X), \tag{3.13}$$

$$A = -\nabla\nabla \log p(\mathbf{f}|\mathbf{y}, X)|_{\mathbf{f}=\hat{\mathbf{f}}}. \tag{3.14}$$

A is the Hessian of the negative log posterior at the mode $\hat{\mathbf{f}}$. The optimization problem of finding $\hat{\mathbf{f}}$ is concave as long as the likelihood is log concave (Rasmussen and Williams, 2005) (which is the case for both probit and Poisson likelihoods). The mode can be found using Newton's method, and an inverse of the covariance matrix $K(X, X)$ is required, which is typically done using Cholesky decomposition in time $O(N^3)$ (details of the algorithm can be found in Rasmussen and Williams (2005)). Once $q(\mathbf{f}|\mathbf{y}, X)$ is found, the posterior distribution over latent function values at test inputs can be found in closed form.

We use Laplace approximation for point process modeling (we consider a point process with the Gaussian process prior in Section 3.2.4), as it is computationally cheaper than the Expectation Propagation method (described in the next paragraph).

Expectation Propagation (EP) (Minka and Lafferty, 2002) is an approximation technique, in which the posterior is approximated by a fully factorised distribution. Recall that we are approximating the posterior,

$$p(\mathbf{f}|\mathbf{y}, X) \propto p(\mathbf{f}|X) \prod_{n=1}^{N} p(y_n|f_n). \tag{3.15}$$

EP approximates it as:

$$q(\mathbf{f}|X, \{\widetilde{Z_1}, \ldots, \widetilde{Z_N}\}, \{\widetilde{\mu_1}, \ldots, \widetilde{\mu_N}\}, \{\widetilde{\sigma_1}^2, \ldots, \widetilde{\sigma_N}^2\}) = p(\mathbf{f}|X) \prod_{n=1}^{N} t_n(f_n|\widetilde{Z_n}, \widetilde{\mu_n}, \widetilde{\sigma_n}^2), \tag{3.16}$$

where

$$t_n(f_n|\widetilde{Z_n}, \widetilde{\mu_n}, \widetilde{\sigma_n}^2) = \widetilde{Z_n}\mathcal{N}(f_n|\widetilde{\mu_n}, \widetilde{\sigma_n}^2) \tag{3.17}$$

and $\widetilde{Z_n}, \widetilde{\mu_n}, \widetilde{\sigma_n}^2$ are the parameters of the local approximation of the likelihood $p(y_n|f_n)$. Each likelihood term $p(y_n|f_n)$ (for classification we use the probit likelihood) is approximated as an unnormalized Gaussian ($\mathcal{N}$ is used to denote the normalized Gaussian distribution, and multiplication by $\widetilde{Z_n}$ renders it unnormalized) over the latent function value f_n.

Henceforth, we will omit the approximation parameters from conditioning, representing the approximated posterior as $q(\mathbf{f}|X)$.

The parameters $\{\widetilde{Z_n}, \widetilde{\mu_n}, \widetilde{\sigma_n}^2\}_{n=1}^N$ need to be chosen so that the approximated posterior $q(\mathbf{f}|X)$ resembles the true posterior $p(\mathbf{f}|\mathbf{y}, X)$ This is achieved by iteratively refining the approximation parameters, where at iteration i of the algorithm parameters $\widetilde{Z_j}$, $\widetilde{\mu_j}$ and $\widetilde{\sigma_j}^2$ are updated (while parameters corresponding to other data are fixed; notice there can be multiple passes over the data, thus we use different indices for the iteration i and for the updated parameters j) by following the steps:

1. The cavity distribution $q_{-j}(f_j)$ is found, which is a combination of the prior and the approximated likelihood corresponding to parameters not optimized at this step,

$$q_{-j}(f_j) = \int p(\mathbf{f}|X) \prod_{n=1}^{N} \mathbb{I}(n \neq j) t_n(f_n|\widetilde{Z_n}, \widetilde{\mu_n}, \widetilde{\sigma_n}^2) df_n. \tag{3.18}$$

2. An unnormalized Gaussian marginal $\hat{t}(f_j)$ is found by minimizing Kullback-Leibler divergence between $\hat{t}(f_j)$ and $p(y_j|f_j)q_{-j}(f_j)$; this can be achieved by matching the first and second moments between the two distributions.

3. The local approximation $t_j(f_j|\widetilde{Z_j}, \widetilde{\mu_j}, \widetilde{\sigma_j}^2)$ (and consequently parameters $\widetilde{Z_j}$, $\widetilde{\mu_j}$ and $\widetilde{\sigma_j}^2$) is found by dividing $\hat{t}(f_j)$ by the cavity distribution $q_{-j}(f_j)$. This can be done in a closed form, and results in a Gaussian.

The number of iterations required to reach convergence can be several times more than the number of observations N, since changing each local approximation affects the global approximation, thus affecting other local approximations. However, there is no guarantee of convergence. Each full pass over the variables can be implemented in time $O(N^3)$ (Rasmussen and Williams, 2005).

We use EP for classification, as EP has been shown to work very well for this kind of problems (Nickisch and Rasmussen, 2008), and moreover we deal with relatively small datasets in the case of classification (smaller than in the case of point process modeling), thus are able to afford a more complex approximation than Laplace.

3.1.5 Kernels

The central object defining a GP is a kernel function, which specifies how the outputs covary as a function of the inputs. For a GP to form a valid distribution, the covariance matrix, which is the Gram matrix $K(X, X)$, needs to be positive semi-definite with respect to an

arbitrary sequence of inputs $X = \{\mathbf{x_1}, \ldots, \mathbf{x_n}\}$,

$$\forall_{\mathbf{c}\in\mathbb{R}^n} : \mathbf{c}^\top K(X, X)\mathbf{c} \geq 0. \tag{3.19}$$

One simple way of obtaining correct kernels without the need of proving the positive semi-definiteness property condition is by combining existing kernels via operations preserving this condition, such as summation or multiplication. In this section we describe a few popular kernels satisfying the positive semi-definiteness property which we use as building blocks for our models.

Radial Basis Functions Kernel One example kernel that we make use of is the Radial Basis Functions (RBF) kernel given by the formula:

$$k_{RBF}(\mathbf{x}, \mathbf{x}') = \sigma \exp\left(-(\mathbf{x} - \mathbf{x}')^\top \Sigma(\mathbf{x} - \mathbf{x}')\right), \tag{3.20}$$

where $\sigma > 0$ is a hyper-parameter controlling the output scale, and Σ controls how kernel values vary across different inputs. Often, a simpler form of the RBF kernel is used, where $\Sigma = \kappa I$ (here, κ is a scalar value). Then, the RBF kernel takes the form

$$k_{RBF}(\mathbf{x}, \mathbf{x}') = \sigma \exp\left(-\kappa\|\mathbf{x} - \mathbf{x}'\|^2\right). \tag{3.21}$$

The hyperparameter κ is called the length scale, and determines the rate at which the kernel diminishes with distance between the inputs.

RBF is a stationary kernel, which means that it depends only on a difference between the inputs. The hyperparameter κ controls the smoothness of the function. Large values of κ make the kernel values sensitive to variation in inputs, which means the inputs may co-vary less, favouring closer fit to the data. Small κ makes the kernel less sensitive to changes in input, which forces the function to be more smooth. The influence of the κ hyperparameter over the shapes of the modeled functions is depicted in Figure 3.2. Notice how $\kappa = 100$ causes the mean function to pass through every point, and make very uninformed predictions about values of the posterior in regions in between the data observatons. This corresponds to functions having trajectories of high variability. $\kappa = 1$ causes the mean function to largely ignore the data points, just learning the slightly decreasing overall trend of the function values. Notice how in this case the samples from the posterior distribution over the functions are very similar. $\kappa = 10$ makes for a smooth function prediction while learning the sinusoidal response. Notice how the further the input from the data observations, the closer the function value is to the mean $\mu(x)$ of the Gaussian process, which is

0. This is particularly apparent for $\kappa = 10$ and $\kappa = 100$, and it could also be observed for $\kappa = 1$ if the plotted range of arguments was significantly wider.

Bias Kernel The bias kernel is given by the formula:

$$k_{Bias}(\mathbf{x}, \mathbf{x}') = b. \tag{3.22}$$

It contains a single hyperparameter $b > 0$, controlling the scale of the output. The bias kernel models the background similarity between the inputs regardless of their values. It is useful in combination with other kernels for modeling the background similarities between outputs regardless of the inputs.

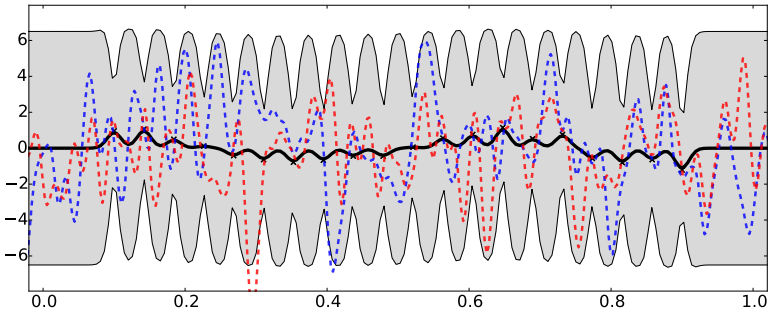
Linear Kernel Another popular kernel function is the linear kernel,

$$k_{LIN}(\mathbf{x}, \mathbf{x}') = \gamma \mathbf{x}^\top \mathbf{x}'. \tag{3.23}$$

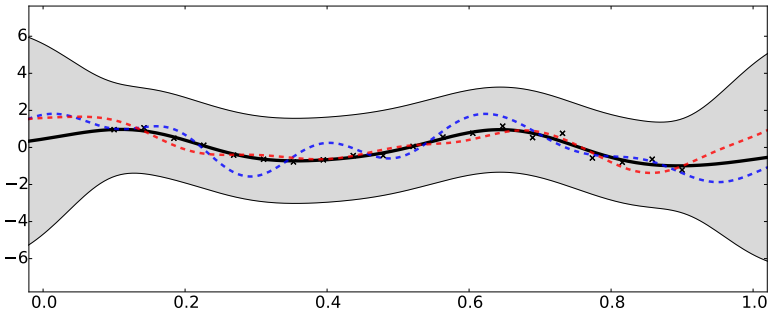
It contains a single hyperparameter $\gamma > 0$, which controls the scale of the outputs. Contrary to the RBF kernel, the linear kernel is non-stationary, as it does not depend only on the differences between pairs of inputs. Notice how this may lead to unbounded kernel values for large inputs. Nevertheless, for high dimensional inputs the linear kernel is a popular choice, as projecting the feature space to higher dimensions might not bring much benefit under such scenarios.

Hyperparameter optimization Kernels often contain hyperparameters controlling the shape of functions modeled by a Gaussian process, such as γ in the linear kernel from Equation (3.23), or the matrix of inter-task correlations B in the ICM kernel introduced later in Section 3.1.6 (there might also be other hyperparameters, such as the variance ρ of the Normal likelihood in the case of regression). The hyperparameter values have a large impact on what the posterior over function values is. For example, in Figure 3.2 we show how different lengthscale hyperparameter values in the RBF kernel lead to overfitting (Figure 3.2a), underfitting (Figure 3.2c), or a relatively good fit to the data (3.2b), illustrating how important an appropriate choice of hyperparameter values is.

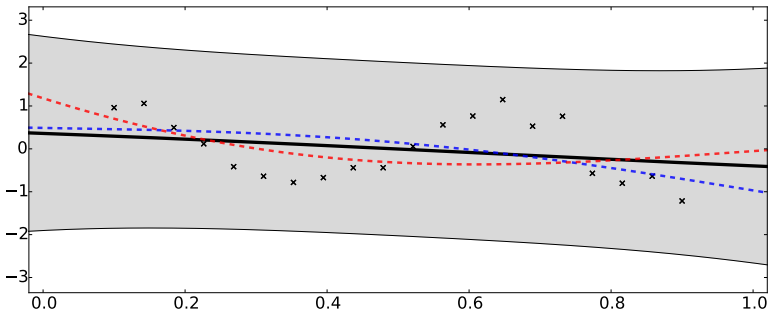
One approach to hyperparameter selection uses a heldout dataset, usually called the *validation set*. It can be used for finding the hyperparameters by iterating over different sets of values and selecting the set which maximizes the performance on the validation set (Bishop, 2006). The downside of this approach is high computational cost. For each set of values the model needs to be re-trained. Moreover, the number of sets of values to



(a) $\kappa = 100$



(b) $\kappa = 10$



(c) $\kappa = 1$

Figure 3.2: Posterior distributions from Gaussian processes with RBF kernels controlled by different hyperparameter values κ. In each subfigure, crosses denote data point observations, a solid line denotes the mean of the posterior distribution $p(\mathbf{f}|\mathbf{y}, X)$, the grey area denotes the 95% confidence interval around the mean prediction, $\mu \pm 2\sigma$, and the dashed lines denote samples from the posterior distribution. The data was generated as follows. For 20 input values x_i equally distributed over the $[0.1, 0.9]$ interval, we generated response values from a Gaussian $Y \sim \mathcal{N}(\sin(4 \times \pi \times x_i), 0.3)$. Then, we found values of GP posteriors parameterized by RBF kernels with varying hyperparameter values $\kappa = 1, 10, 100$. For all GPs, the hyperparameter σ^2 was fixed to 10.

be checked grows exponentially with the number of hyperparameters, which can quickly become prohibitive. Alternative approaches using a validation set try to limit the number of sets of values that are being inspected, with examples of random hyperparameter search (Bergstra and Bengio, 2012) and Bayesian optimization (Shahriari et al., 2015).

In the case of Gaussian processes, a common alternative to using a held-out validation set is choosing hyperparameters by maximizing the marginal likelihood of the training data $p(\mathbf{y}|X)$, also called the evidence (Rasmussen and Williams, 2005). The evidence is given by the equation

$$p(\mathbf{y}|X) = \int p(\mathbf{y}|\mathbf{f}, X)p(\mathbf{f}|X)d\mathbf{f}\,. \tag{3.24}$$

In the case of regression the evidence can be computed in a closed form. The logarithm of the evidence in the case of regression takes form:

$$\log p(\mathbf{y}|X) = -\frac{\mathbf{y}^\top \left(K(X,X)^{-1} + \rho\mathbf{I}\right)\mathbf{y}}{2} - \frac{\log(|K(X,X)^{-1} + \rho\mathbf{I}|)}{2} - \frac{N\log(2\pi)}{2}. \tag{3.25}$$

The different terms in the log evidence are interpretable in the following way: the first term rewards models closely fitting to the data, the second term rewards simple models, and the role of the third term is normalization. The time complexity of finding the evidence is $O(N^3)$ (where N is the number of examples in the data set), since an inverse of the covariance matrix is required. In the case of a non-Gaussian likelihood, Laplace or EP approximation can be used, which, as discussed in Section 3.1.4, require $O(N^3)$ time per algorithm's iteration.

A gradient based approach can be used for maximizing the evidence with respect to the hyperparameters, therefore the hyperparameter training takes $O(N^3)$ times the number of steps of the gradient search.[5] The optimization requires finding derivatives of the evidence with respect to the hyperparameter values, which, by the chain rule, breaks down to finding derivatives of a kernel with respect to its hyperparameters.

There are other approaches to learning hyperparameters of complex kernels, such as the framework of multi-kernel learning, which is often based on learning the weights in a linear combination of component kernels (Gnen and Alpaydin, 2011).

[5]Notice that this ignores the kernel function evaluation between every pair of data points, the complexity of which is $O(k * N^2)$, where k is the complexity of the kernel evaluation between a single pair of inputs. Usually this is assumed to be dominated by $O(N^3)$, however in principle it does not need to be the case.

3.1.6 Multi-task Learning with Gaussian Processes

Let us consider a multi-task learning problem with M tasks, where each task consists of an input-output pair. The idea of multi-task learning learning is to predict an output for an input by exploiting correlations existing across different tasks. Examples include modeling multiple emotions expressed in text over different scales (Beck et al., 2014), and modeling multiple stock price movements over time (Bitvai and Cohn, 2015a). In such applications instead of modeling each task independently it may be beneficial to model the tasks jointly and exploit correlations existing between them. One popular approach to modeling multiple tasks with GPs is based on the Intrinsic Coregionalization Model (ICM) (Bonilla et al., 2008; Álvarez et al., 2012). It is a method which has been successfully applied to a range of NLP tasks (Cohn and Specia, 2013; Beck et al., 2014).

In ICM we define a kernel over the joint space of both the inputs $\mathbf{x} \in U$ and the tasks $m_1, \ldots, m_M$, thus, instead of considering kernels of the form $k(\mathbf{x}, \mathbf{x}')$ (which was the case before), we consider kernels $k\left((\mathbf{x}, m), (\mathbf{x}', m')\right)$. The ICM kernel takes the form:

$$k_{\text{ICM}}\left((\mathbf{x}, m), (\mathbf{x}', m')\right) = k_U(\mathbf{x}, \mathbf{x}')k_M(m, m') = k_U(\mathbf{x}, \mathbf{x}')B_{m,m'}, \tag{3.26}$$

where B is a matrix (called the coregionalization matrix) indexed by pairs of tasks, m and m' denote the tasks corresponding to the inputs $\mathbf{x}$ and $\mathbf{x}'$, k_U is a kernel comparing inputs $\mathbf{x}$ and $\mathbf{x}'$. The intuition is that pairs of tasks m, m' exhibiting similar characteristics should correspond to high correlation values $B_{m,m'}$ in the coregionalization matrix. In Figure 3.3 we depict an illustrative coregionalization matrix B modeling similarities between three different tasks corresponding to emotions: excitement, happiness and sadness. One might expect that happiness and excitement are correlated more than happiness and sadness, as illustrated in Figure 3.3.

The coregionalization matrix B is a hyperparameter, and so it is learnt jointly with other hyperparameters by maximizing the evidence, as described in Section 3.1.5. Hyperparameter optimization is expected to drive cells from matrix B corresponding to similar tasks to higher values than cells corresponding to dissimilar tasks.

For kernel k_{ICM} to be valid, matrix B needs to be symmetric and positive semi-definite. This can be ensured by formulating the matrix B as $B = LL^\top + \boldsymbol{\kappa} I$, which we follow in our experiments in Chapters 4 and 5. The rank of the matrix L controls the complexity of task correlations, and $\boldsymbol{\kappa}$ encodes the information about the task independence.

Figure 3.3: An illustrative example of a coregionalization matrix B. Inputs $\mathbf{x}$ are modeled across multiple emotions: excitement, happiness and sadness. The darker the cell in the matrix, the higher the correlation between a pair of emotions corresponding to the cell. For real world experiments on multi-task learning from multiple emotions see Beck et al. (2014).

3.1.7 Gaussian Processes for NLP and Social Media

Gaussian processes (GPs) have been applied to various applications related to language modeling. Cohn and Specia (2013) used GPs for modeling quality of sentence level data annotation under multiple annotators. The authors showed how a GP with a multi-task learning kernel achieves superior results over baselines. Shah et al. (2013) employed GPs for Quality Estimation of Machine Translation. They leveraged hyperparameter learning in the GP framework for feature selection, by interpreting the learnt hyperparameters as measuring the importance of corresponding features. Bitvai and Cohn (2015b) modeled peer-to-peer lending over text documents using GPs. The authors developed specific kernels and showed how multiple sources of data can be combined, both structured and unstructured. Gaussian processes were used for emotion analysis in Beck et al. (2014), where a multi-task learning kernel was used to learn correlations between different types of emotions, allowing for better results and more interpretation than in single-task learning.

Gaussian processes have also been used for modeling Twitter data. Preotiuc-Pietro and Cohn (2013) modeled hash tag frequency time-series, directly employing GPs with a Gaussian likelihood over frequency occurrences. This ignores a central characteristic of the problem, namely that frequencies can only take non-negative integer values. Moreover,

the authors discretize time, which is another limitation of their study. In Section 3.2.4 we explain how Gaussian processes can be used in the point process framework, avoiding both shortcomings. GPs were also used to run studies on Twitter users. Lampos et al. (2014) demonstrated how user impact can be successfully predicted. Preotiuc-Pietro et al. (2015) showed how GPs can be used for classification of user occupational class in Twitter. In both cases these used a fairly straightforward application of GP regression or classification.

3.2 Point processes

In this section we introduce point processes (PPs), a probabilistic framework for modeling points over a space. Point processes have been succesfully used to solve many problems involving tracking phenomena, e.g. disease mapping (Brix and Diggle, 2001) and conflict mapping (Zammit-Mangion et al., 2012). In general, point processes model points over a space by an intensity function $\lambda(t)$, such that high values of the intensity function correspond to high density of points, and low values of the intensity function correspond to low density of points.

We use two prominent models from the point process framework. First, the Log-Gaussian Cox Process (Møller and Syversveen, 1998), which models the intensity function $\lambda(t)$ using a Gaussian process. It is a type of Poisson process, i.e., occurrences of points in disjoint subregions are independent. The second point process we consider is the Hawkes process (Hawkes and Oakes, 1974), which models points over time via an intensity function which conveys an assumption of mutual excitation, i.e., occurrences of points increase the likelihood of points occurring soon afterwards.

In Figure 3.4 we depict how the two processes fit within the point process framework. In the remaining part of this section we describe point processes and how they can be used in practice for modeling social media data.

3.2.1 Motivation

Predictive tasks considering rumours in social media are typically based on extracting descriptors from rumours and then trying to predict some quantities of interest using a traditional machine learning approach, be it classification of political disinformation (Ratkiewicz et al., 2011), clustering of web sources (Nel et al., 2010) or classification of credibility of events (Castillo et al., 2011). The aforementioned approaches typically require summarizing statistics of event dynamics, which can be challenging and prone to losing information. Instead, when modeling the post times from rumours we resort to point processes, a prob-

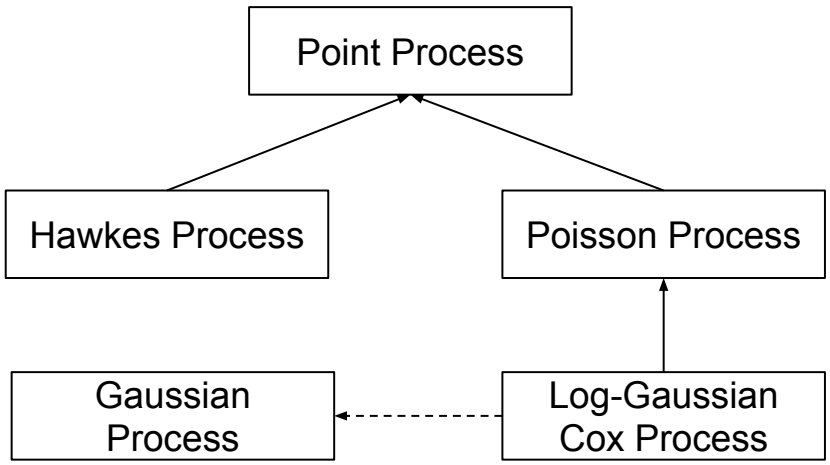


Figure 3.4: Relations between stochastic processes considered in this thesis. Solid lines correspond to the *is type of* relation, and the dashed line corresponds to the *models its intensity function as* relation.

abilistic framework for modeling events over time. Point processes allow to model event dynamics without the need of extracting descriptors from sets of timestamps of events, instead providing an elegant way of dealing with temporal data. Moreover, PPs provide ways of answering questions like *How many points will occur in a given interval of time?* or *When will the next point occur?*, which can be challenging to tackle with other approaches. We are going to consider these types of questions when modeling rumour popularity in Chapters 5 and 6.

3.2.2 Preliminaries

In this section we introduce the fundamental quantities describing a PP (the intensity function, the probability density function, the survival function), and derive the likelihood of a set of events occurring over an interval of time. Let us denote the number of events that occurred right before time t as $N(t)$, and let $dN(t)$ denote the number of events in the interval $[t, t + \delta t]$. The *conditional* intensity function is defined as the instantaneous probability of one event occurrence in the interval $[t, t + \delta t)$, conditioned on the history of previous event occurrences,

$$\lambda\left(t|\mathbf{H}_{t^-}\right) = \lim_{\delta t \to 0} \frac{P\left(dN(t) = 1|\mathbf{H}_{t^-}\right)}{\delta t} = \lim_{\delta t \to 0} \frac{P\left(t \leq T < t + \delta t|T \geq t, \mathbf{H}_{t^-}\right)}{\delta t}, \quad (3.27)$$

where $\mathbf{H}_{t^-}$ is the event history up to time t (excluding t) and T is a random variable denoting the time of the next event occurrence. The earliest next event occurrence is t_0, typically

determined by the arrival of the most recent event from $\mathbf{H}_{t^-}$ (if there are no preceding events, we assume $t_0 = 0$). Note that for $t < t_0$ we get $\lambda\left(t|\mathbf{H}_{t^-}\right) = 0$. The conditional intensity function can be rewritten as

$$\lambda(t|\mathbf{H}_{t^-}) = \underbrace{\lim_{\delta t \to 0} \frac{P(t \leq T < t + \delta t|\mathbf{H}_{t^-})}{\delta t}}_{p(t|\mathbf{H}_{t^-})} \underbrace{\frac{1}{P(T \geq t|\mathbf{H}_{t^-})}}_{1/S(t|\mathbf{H}_{t^-})}, \tag{3.28}$$

where $p(t|\mathbf{H}_{t^-})$ is the probability density function (pdf) of the random variable T and $S(t|\mathbf{H}_{t^-})$ is the survival function, which models the likelihood of the next event happening only after time t (the name *survival function* comes from the study of death events, where $S(t|\mathbf{H}_{t^-})$ models the *survival* until time t).

Now we are going to represent $S(t|\mathbf{H}_{t^-})$ and $p(t|\mathbf{H}_{t^-})$ as functions of $\lambda(t|\mathbf{H}_{t^-})$. Let us denote the cumulative density function of the random variable T as $F(t|\mathbf{H}_{t^-})$. Since $S(t|\mathbf{H}_{t^-}) = 1 - F(t|\mathbf{H}_{t^-})$, we get

$$dS(t|\mathbf{H}_{t^-}) = -dF(t|\mathbf{H}_{t^-}) = -p(t|\mathbf{H}_{t^-})dt. \tag{3.29}$$

From Equations (3.28) and (3.29), we get

$$\lambda(t|\mathbf{H}_{t^-})dt = \frac{p(t|\mathbf{H}_{t^-})dt}{S(t|\mathbf{H}_{t^-})} = -\frac{dS(t|\mathbf{H}_{t^-})}{S(t|\mathbf{H}_{t^-})}. \tag{3.30}$$

Solving for $S(t|\mathbf{H}_{t^-})$:

$$S(t|\mathbf{H}_{t^-}) = \exp\left(-\int_{t_0}^{t} \lambda(s|\mathbf{H}_{t^-})ds\right), \tag{3.31}$$

where t_0 is the earliest time an event can arrive (either the time of the most recent event from $\mathbf{H}_{t^-}$, or 0 if there are no events in $\mathbf{H}_{t^-}$). This implies

$$p(t|\mathbf{H}_{t^-}) = -S'(t|\mathbf{H}_{t^-}) = \lambda(t|\mathbf{H}_{t^-}) \exp\left(-\int_{t_0}^{t} \lambda(s|\mathbf{H}_{t^-})ds\right). \tag{3.32}$$

Figure 3.5 shows the cumulative density function $F(t|\mathbf{H}_{t^-})$ and the probability density function $p(t|\mathbf{H}_{t^-})$ of the inter-arrival time until the next event occurrence for a linear intensity function $\lambda(t|\mathbf{H}_{t^-}) = t$. The probability distribution initially increases and then falls, reaching its maximum at around 1, with the corresponding cumulative distribution monotonically increasing.

The likelihood of N events occurring at times $t_1 < \ldots < t_N$ over an interval $[0, T]$ can

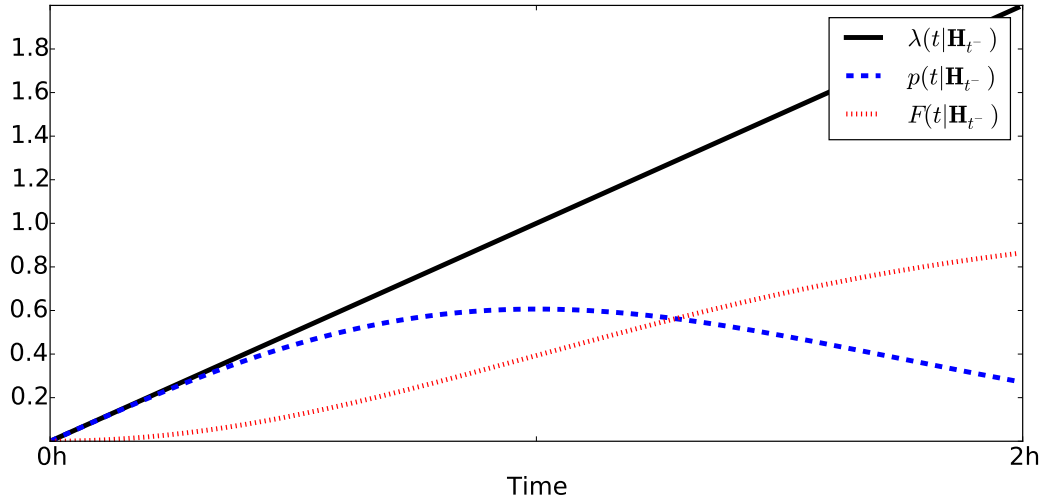


Figure 3.5: Intensity function values over time for $\lambda(t|\mathbf{H}_{t^-}) = t$ (denoted by the solid black line), and the corresponding: instantaneous likelihood of the first event occurrence $p(t|\mathbf{H}_{t^-}) = t\exp(-\frac{t^2}{2})$ (denoted by the dashed blue line) and cumulative distribution function of the first event occurrence $F(t|\mathbf{H}_{t^-}) = 1 - \exp(-\frac{t^2}{2})$ (denoted by the dotted red line). Notice the intensity function (and, consequently, the other quantities too) is independent from the history.

be formulated as

$$L\left(t_1, \ldots, t_N\right) = p\left(t_1|\mathbf{H}_{t_1^-}\right) \times p\left(t_2|\mathbf{H}_{t_2^-}\right) \times \ldots \times p\left(t_N|\mathbf{H}_{t_N^-}\right) \times S\left(T|\mathbf{H}_T\right). \quad (3.33)$$

The last term $S(T|\mathbf{H}_T)$ represents the likelihood that there is no event in the interval $(t_n, T]$. After plugging the formulas for the pdf from Equation (3.32) and the survival function from Equation (3.31) into Equation (3.33), we obtain the following form of the joint likelihood of the sequence of events under a point process:

$$L(t_1, \ldots, t_N) = \left(\prod_{n=1}^{N} \lambda(t_n|\mathbf{H}_{t_n^-})\right) \left(\underbrace{\exp\left(-\int_0^T \lambda(s|\mathbf{H}_{s^-})ds\right)}_{P(E_T)}\right). \quad (3.34)$$

In Equation (3.34), the first multiplicand is the product of intensity function values at the observed event timestamps and the second multiplicand is the exponentiated integral of the intensity function over the observation window. Henceforth, we are going to represent the second part as $P(E_T)$, assuming the intensity function parametrizing it is known from the

context.

Predicting the Next Arrival Time

The probability density function from Equation (3.32) can be used for sampling timestamps of the next event occurrence. For ease of sampling, we approximate the integral in the formula for pdf,

$$p(t|\mathbf{H}_{t^-}) \approx \lambda(t|\mathbf{H}_{t^-}) \exp(-t\lambda(\frac{t}{2}|\mathbf{H}_{\frac{t}{2}^-})), \tag{3.35}$$

where $\frac{t}{2}$ is the middle point of the interval $[0, t]$, and $\lambda(\frac{t}{2}|\mathbf{H}_{\frac{t}{2}^-})$ is the intensity function value in the middle point of that interval.

There exist multiple sampling schemes that can be facilitated for obtaining the arrival time of the next event. One example is the importance sampling (Gelman et al., 2003), where a proposal distribution is used to sample from and then reweighted according to Equation (3.35). Assuming the previous event occurred at time s, we obtain the sampled arrival time of the next tweet as outlined in Algorithm 3.1. Another sampling method is the *Ogata's thinning algorithm* (Ogata, 2006), which instead takes a thinning approach. In the *Ogata's thinning algorithm* timestamps are sampled from a homogeneous Poisson process with the intensity function bounding the simulated point process' intensity function at every point of time (a homogenous Poisson process is a point process with a constant intensity function). Timestamps are then rejected if they are happening too soon according to the intensity of the PP. In the case of the Hawkes process (introduced later in Section 3.2.5) the intensity function is decreasing over time, which facilitates application of the *Ogata's thinning algorithm*. In the case of other PPs a sampling approach could be utilized for finding the maximum. The *Ogata's thinning algorithm* is outlined in Algorithm 3.2. Either of the two sampling algorithms can be repeated until the end of the interval of interest for finding a sequence of times of event occurrences.

3.2.3 Poisson Processes

A Poisson process is a point process with the assumption that point occurrences in disjoint subspaces are independent. It is characterized by the intensity function independent from the history, $\lambda(t|\mathbf{H}_{t^-}) = \lambda(t) > 0$. A homogeneous Poisson process (HPP) assumes the intensity to be constant with respect to time, i.e., $\lambda(t) = \lambda$, whereas an inhomogeneous Poisson process (IPP) can model points occurring at a variable rate by considering the intensity to be a function of time, *i.e.* $\lambda(t)$. In a Poisson process the number of tweets y

Algorithm 3.1 Importance sampling for predicting the arrival time of the next event. Arrival times are sampled from a proposal distribution which is relatively easy to sample from, and afterwards they are averaged by weighting their importance according to the pdf of the true distribution.

Input: probability density function $p(t|\mathbf{H}_{t^-})$, proposal distribution $q(t)$, number of samples N
for $i = 1$ **to** N **do**
  Sample $u_i \sim q(t)$.
  $w_i = \frac{p(u_i|\mathbf{H}_{u_i^-})}{q(u_i)}$
end for
Predict the expected next arrival time as
$\bar{u} = \sum_{i=1}^{N} u_i \frac{w_i}{\sum_{j=1}^{N} w_j}$
Return: $\bar{u}$

Algorithm 3.2 Ogata's thinning algorithm for predicting the arrival time of the next event within the interval of time $[0, T]$. Candidate arrival times are repeatedly sampled from a Homogenous Poisson Process (HPP; see Section 3.2.3 for the definition of an HPP) with an intensity β being an upper bound to the intensity $\lambda(t|\mathbf{H}_{t^-})$ of the point process until the conditions for acceptance are satisfied.

Input: intensity function $\lambda(t|\mathbf{H}_{t^-})$, time T
$\beta = \max\{\lambda(t|\mathbf{H}_{t^-})\}_{t\in[0,T]}$
$t = 0$
while $t \leq T$ **do**
  Sample $s \sim$ HPP(β)
  Sample $u \sim U(0, 1)$
  if $(t + s > T)$ OR $(u > \frac{\lambda(t|\mathbf{H}_{t^-})}{\beta})$ **then**
    $t = t + s$
  else
    Return: t
  end if
end while
Return: None

occurring in an interval $[s, e]$ is Poisson distributed with the rate parameter equal $\int_s^e \lambda(t)dt$,

$$\begin{aligned} P\left(y|\lambda(t), [s,e]\right) &= \text{Poisson}\left(y|\int_s^e \lambda(t)dt\right) \\ &= \frac{\left(\int_s^e \lambda(t)dt\right)^y \exp\left(-\int_s^e \lambda(t)dt\right)}{y!}. \end{aligned} \tag{3.36}$$

Figure 3.6 shows three example intensity functions of Poisson processes and events sampled from their distributions. Two intensity functions are non-constant (thus correspond to IPP). Notice how high intensity function values correspond to high number of generated events. The last plotted intensity function corresponds to HPP, and the pairs of neighbouring sampled timestamps are roughly equidistant.

3.2.4 Log-Gaussian Cox Processes

The doubly stochastic Poisson process, also called the Cox process, is a type of Poisson process in which the intensity function $\lambda(t)$ is modelled via a stochastic process. This way there are two levels of stochasticity when modeling points coming from a Cox process: the intensity function is drawn from a stochastic process, and then the points are drawn from a stochastic process controlled by the sampled intensity function. We consider the log-Gaussian Cox process (LGCP) (Møller and Syversveen, 1998), which assumes the intensity function $\lambda(t)$ is modelled using the latent function $f(t)$ sampled from a Gaussian process (Rasmussen and Williams, 2005), such that $\lambda(t) = \exp\left(f(t)\right)$ (the exponent ensures positivity). Combining the Gaussian process assumption with Equation (3.32), the likelihood that a single point occurs at time t in the interval $[s, t]$ given the latent function $f(t)$ is

$$p(t|f) = \exp\left(f(t)\right) \exp\left(-\int_s^t \exp\left(f(t)\right) dt\right). \tag{3.37}$$

Then, the likelihood of points $E = \{t_1, \ldots, t_N\}$ in the time interval $[0, T]$ given the latent function f can be obtained as

$$L(E|f) = \exp\left(-\int_0^T \exp\left(f(t)\right) dt + \sum_{n=1}^{N} f(t_n)\right). \tag{3.38}$$

The likelihood (3.38) is commonly approximated by assuming constant intensities in subregions of T (Møller and Syversveen, 1998; Vanhatalo et al., 2013) to overcome computational difficulties arising due to integration. Following this, the likelihood of the arrival

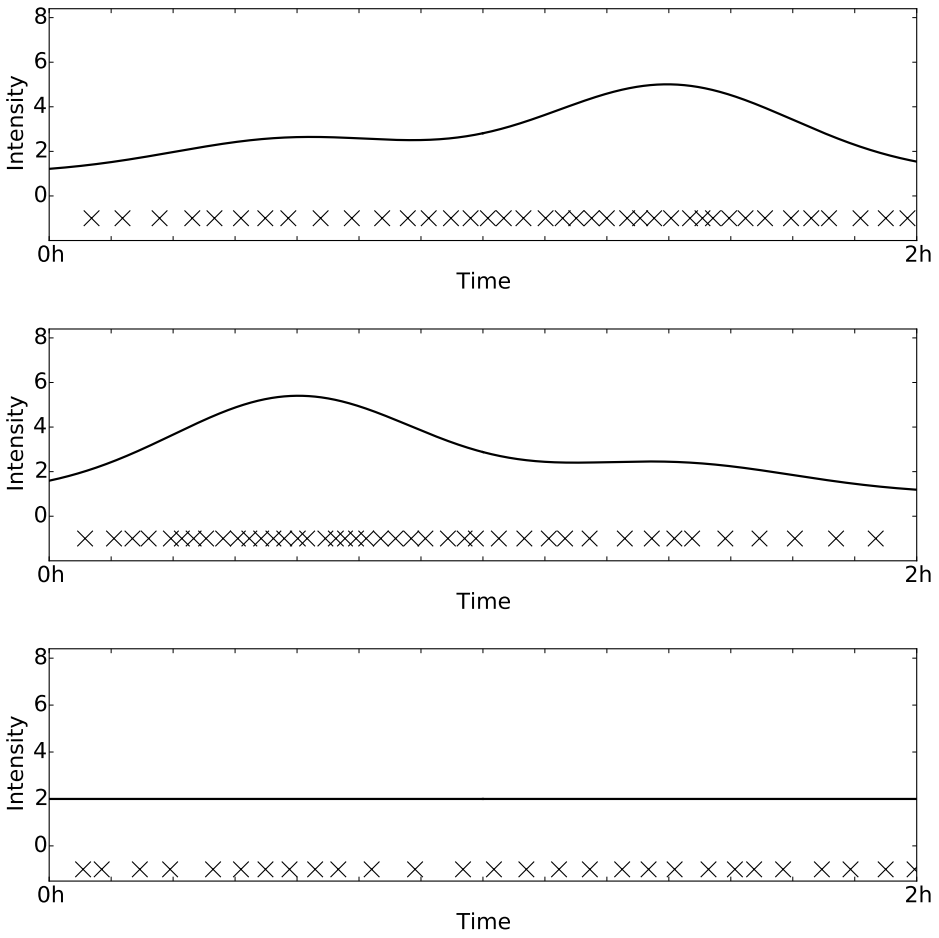


Figure 3.6: Time varying intensity functions for two inhomogeneous and one homogeneous Poisson process, and the corresponding samples of tweet arrivals. The corresponding intensity functions are (starting from the top): $\lambda_1(t) = 8 \times \mathcal{N}(4, 2) + 20 \times \mathcal{N}(10, 2) + 1$, $\lambda_2(t) = 22 \times \mathcal{N}(4, 2) + 7 \times \mathcal{N}(10, 2) + 1$, $\lambda_3(t) = 2$. Here, $\mathcal{N}(\mu, \sigma^2)$ corresponds to a single random variable drawn from a Normal distribution with mean μ and variance σ^2. Note that more points are sampled from regions where an intensity function takes higher values. Timestamps sampled using the Ogata's thinning algorithm (see Algorithm 3.2).

times E is approximated as

$$L(E|f) = \prod_{s=1}^{S} \text{Poisson}\left(y_s \mid l_s \exp\left(f(\dot{t}_s)\right)\right). \tag{3.39}$$

Here, the time interval $[0, T]$ is divided into S intervals, $\dot{t}_s$ is the middle point of interval s, l_s is its length, and y_s is the number of events from E falling into interval s.

Inference

The distribution of the posterior $p(f(t)|E)$ at an arbitrary timestamp t is calculated based on a Gaussian process prior and the Poisson likelihood. It is intractable and approximation techniques are required. There exist various methods to deal with calculating the posterior, with Laplace approximation and Expectation Propagation being two popular approximations (Rasmussen and Williams, 2005) (we describe both in Section 3.1.4). In the experiments we make use of Laplace approximation, which leads to the posterior being approximated by a Gaussian distribution based on the first 2 moments. Then, the predictive distribution over function values at a timestamp t^* is obtained using the approximated posterior. This predictive distribution is then used to obtain the intensity function value at a timestamp of interest t^*,

$$\lambda(t^*|E) = \int \exp\left(f(t^*)\right) p\left(f(t^*)|E\right) df\,. \tag{3.40}$$

The intensity function can then be used for answering various queries about the unobserved intervals of time, e.g. how many events are expected to occur in the future 1 hour.

Related Work on Log-Gaussian Cox Processes

The Log-Gaussian Cox process (LGCP) has been applied to applications involving modeling temporal phenomena, e.g. disease mapping (Brix and Diggle, 2001) and conflict mapping (Zammit-Mangion et al., 2012). LGCP has not been widely applied to modeling the spread of information through social networks (Linderman and Adams, 2014), which might be due to the independence assumption (in social networks one may want to explicitly model posts influencing one another). Linderman and Adams (2014) applied LGCP to modeling the background intensity of the Hawkes process model (see Section 3.2.5 for an introduction to Hawkes processes). Simma and Jordan (2010) developed a Poisson Cox process model for capturing propagation of multiple cascades across a social network, although the authors do not use a Gaussian process prior.

3.2.5 Hawkes processes

A point process which explicitly models mutually exciting phenomena across multiple dimensions is the multi-variate Hawkes process (Hawkes and Oakes, 1974). It does not make an independence assumption like the Poisson process does. Instead, the Hawkes process defines an underlying intensity function explicitly modeling influences from the past events. In other words, the intensity function of the Hawkes process models the self-exciting nature by adding up influences from past events. In this section we give an introduction to the Hawkes process defined over the joint space of: time, $|U|$ users and $|R|$ rumours. We follow the model definition from Yang and Zha (2013) (originally, the authors considered memes, however we adhere to our rumour use-case).

Let us consider a sequence of tweets $\{(t_n, i_n, m_n, \boldsymbol{w}_n)\}_1^N$, where each tweet is a quadruple composed of:

1. t_n, the time of occurrence of the nth tweet,
2. i_n, the user that posted the nth tweet, represented as an integer ranging from 1 to $|U|$,
3. m_n, the rumour the nth tweet pertains to, represented as an integer ranging from 1 to $|R|$,
4. $\boldsymbol{w}_n$, the text of the nth tweet (over vocabulary of size V, encoded using a vector representation).

Recall our notation is summarized in the Nomenclature on page xii.

In the multi-variate Hawkes process, an intensity function $\lambda_{i,m}(t|\mathbf{H}_{t^-})$ is defined for each user i and rumour m pair, and its value at time t depends on the subset of the history $\mathbf{H}_{t^-}$ of previous events pertaining to rumour m,

$$\lambda_{i,m}(t|\mathbf{H}_{t^-}) = \mu_i \gamma_m + \sum_{\ell=1}^{N} \mathbb{I}(m_\ell = m)\mathbb{I}(t > t_l)\alpha_{i_\ell,i}\kappa(t - t_\ell), \tag{3.41}$$

where μ_i is the base intensity for user i and γ_m is the base intensity for rumour m. The matrix $\boldsymbol{\alpha}$ of size $|U| \times |U|$ (denoted by a small letter to adhere with the notation from previous work (Yang and Zha, 2013)) models the degrees of influence between pairs of users posting tweets. The influence the users have on one another is not necessarily symmetric, and so the $\boldsymbol{\alpha}$ matrix is asymmetric, and can be interpreted as the underlying influence network.

The second term in Equation (3.41) represents the influence from the tweets that happened prior to the time of interest. The influence from each tweet decays over time and is

modeled using a decay kernel $\kappa(t - t_\ell)$, typically the exponential kernel,

$$\kappa(t - t_\ell) = \omega \exp(-\omega(t - t_\ell)), \tag{3.42}$$

where $\omega > 0$.

Figure 3.7 shows three samples from a Hawkes process (HP) with a single user tweeting about a single rumour over an interval $[0, 2h]$. We can see that as each consecutive tweet occurs, the intensity of the HP increases, rendering the occurrence of events soon afterwards more likely. Also, when no events occur, the intensity decays exponentially to the base intensity, making arrivals of new events less likely. Overall, notice how the three samples from the same Hawkes process differ, showing that a Hawkes process with the same set of hyperparameters may explain various trajectories of tweet arrivals.

Likelihood

In this section we discuss the likelihood of a set of tweets coming from multiple rumours under the multi-variate Hawkes process. Following Yang and Zha (2013) we also incorporate a linguistic component to the model. In Chapter 7 we show how inference and optimization can be performed under the Hawkes process model for a sequence classification application.

The language component is modeled as a multinomial distribution conditioned on the rumour that a tweet pertains to. This way, the likelihood of text $\boldsymbol{w}_n$ from the nth post and discussing rumour m_n is (encoding all text vectors in a matrix W, such that an element v in row n is denoted as W_{nv})

$$p(\boldsymbol{w}_n | m_n) = \prod_{v=1}^{V} \beta_{m_n,v}^{W_{nv}}, \tag{3.43}$$

where β is a parameter specifying the language models, V is the vocabulary size, $\boldsymbol{w}_n$ is a vector counting words in the nth post, and W_{nv} is the vth dimension of the vector $\boldsymbol{w}_n$.

Combining the likelihood of text content with the likelihood of tweet occurrences under the Hawkes process, we obtain the following form of the complete likelihood: (Yang and Zha, 2013)

$$L(\boldsymbol{t}, \boldsymbol{i}, \boldsymbol{m}, W | \boldsymbol{\beta}, \boldsymbol{\mu}, \boldsymbol{\gamma}, \boldsymbol{\alpha}, \omega) = \left(\prod_{n=1}^{N} p(\boldsymbol{w}_n | m_n) \right) \left(\prod_{l=1}^{N} p(t_l, i_l | m_l, \mathbf{H}_{t_l^-}) \times P(E_T) \right). \tag{3.44}$$

Recall the meaning of $P(E_T)$ from Equation (3.34). It is convenient to consider the log

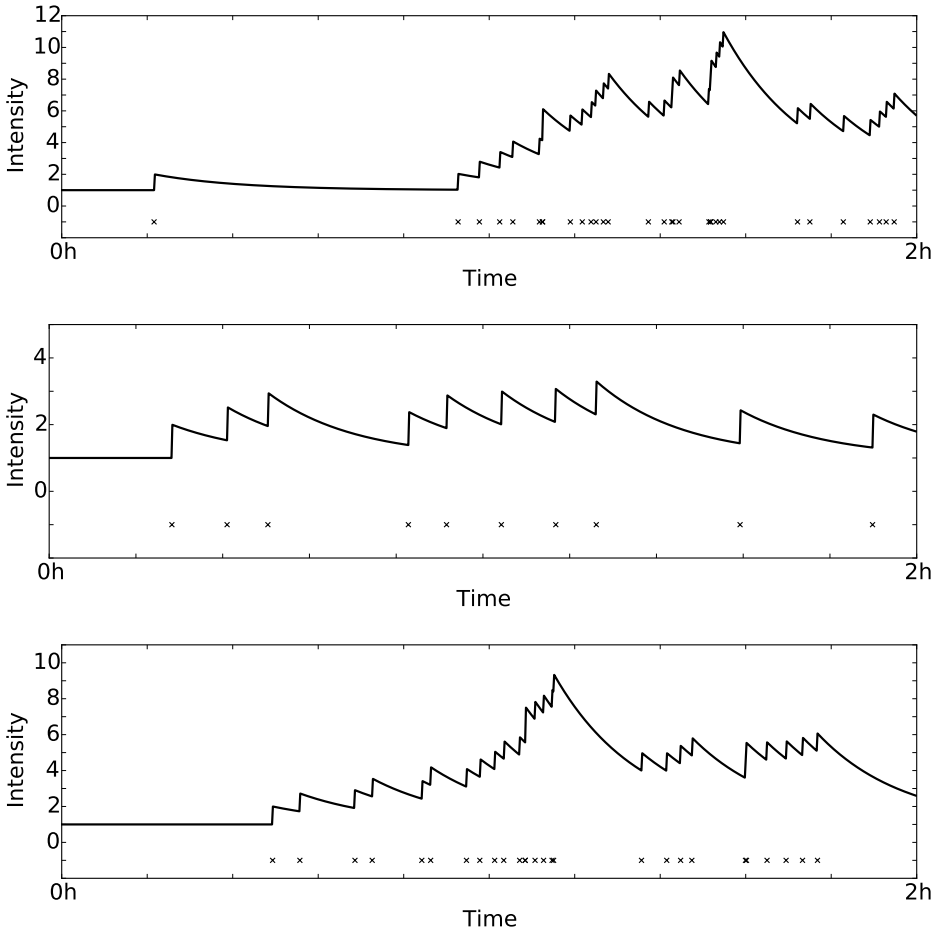


Figure 3.7: Samples of tweets drawn from a uni-variate Hawkes process (denoted by the crosses at the bottom of each subfigure) and the corresponding intensity function values over time. The samples have been obtained using the Ogata's thinning algorithm (see Section 3.2.2). The parameters of the Hawkes process are: $\mu = \gamma = 1$ and $\omega = 1$.

likelihood,

$$\ell(\boldsymbol{t}, \boldsymbol{i}, \boldsymbol{m}, W|\boldsymbol{\beta}, \boldsymbol{\mu}, \boldsymbol{\gamma}, \boldsymbol{\alpha}, \omega) = \sum_{n=1}^{N}\sum_{v=1}^{V} W_{nv} \log \beta_{m_n,v} + \sum_{n=1}^{N} \log \lambda_{i_n,m_n}(t_n|\mathbf{H}_{t_n^-}) - \sum_{i=1}^{|U|}\sum_{m=1}^{|R|}\int_0^T \lambda_{i,m}(s|\mathbf{H}_{s^-})ds. \tag{3.45}$$

It can be shown (the derivation of this fact is included in Appendix A) that the integral term from Equation (3.45) can be represented in a closed form,

$$\sum_{i=1}^{|U|}\sum_{m=1}^{|R|}\int_0^T \lambda_{i,m}(s|\mathbf{H}_{s^-})ds = T\sum_{i=1}^{|U|}\sum_{m=1}^{|R|}\mu_i\gamma_m + \sum_{i=1}^{|U|}\sum_{\ell=1}^{N}\alpha_{\alpha_{i_\ell,i}}K(T-t_\ell), \tag{3.46}$$

where $K(T-t_k) = 1-\exp(-\omega(T-t_k))$ arises from the integration of $\kappa(t-t_k)$ (see Appendix A for details). This leads to the following form of the log-likelihood (plugging Equations (3.41) and (3.46) into Equation (3.45)):

$$\begin{aligned}\ell(\boldsymbol{t}, \boldsymbol{i}, \boldsymbol{m}, W|\boldsymbol{\beta}, \boldsymbol{\mu}, \boldsymbol{\gamma}, \boldsymbol{\alpha}, \omega) = &\sum_{n=1}^{N}\sum_{v=1}^{V} W_{nv} \log \beta_{m_n,v} \\ &+ \sum_{n=1}^{N} \log \lambda_{i_n,m_n}(t_n|\mathbf{H}_{t_n^-}) \\ &- T\sum_{i=1}^{|U|}\sum_{m=1}^{|R|}\mu_i\gamma_m - \sum_{i=1}^{|U|}\sum_{\ell=1}^{N}\alpha_{\alpha_{i_\ell,i}}K(T-t_\ell)\end{aligned} \tag{3.47}$$

The parameters of the Hawkes process can be optimized by maximizing the log-likelihood of the data from Equation (3.47). In Chapter 7 we will consider different ways of learning the parameters.

Related Work on Hawkes Processes

Multiple works have modeled posts' occurrences in social media using Hawkes processes (HP). Yang and Zha (2013) applied HP to inferring the underlying network of connections based on the timestamps of observed events and the text content. The limitations of dealing with a fixed kernel were addressed by Zhou et al. (2013b) through learning the kernel function from data, and by Wang et al. (2016) who introduced a link function wrapping the sum of influences from past events. Joint modeling of information spread and text has been considered by He et al. (2015), who introduced a joint model of topics and network

inference from information propagation. Du et al. (2015) developed a Dirichlet-Hawkes process model, which models clustering of events across Hawkes processes via a Dirichlet process. Also, prior knowledge about the users and connections from a social network has been incorporated into the Hawkes process model (Zhou et al., 2013a; Kobayashi and Lambiotte, 2016; Srijith et al., 2017). Here, instead of performing network inference or clustering of tweets, we propose a novel application of an HP to a sequence classification task, demonstrating how the assumptions made by the model lead to outperforming baselines (see Chapter 7).

3.3 Evaluation Metrics

In this section we consider evaluation metrics for regression, Poisson regression and classification problems. We denote the vector of outputs from the model as $\hat{\mathbf{y}}$, and the vector of ground truth outputs as $\mathbf{y}$, assuming lengths of both to be equal to N.

Regression In regression problems continuous outputs are predicted. A popular error metric is mean squares error (MSE),

$$\text{MSE}(\hat{\mathbf{y}}, \mathbf{y}) = \frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2, \tag{3.48}$$

for a vector of N prediction values $\hat{\mathbf{y}}$ and a vector of ground truth values $\mathbf{y}$. Notice the correspondence between the MSE and the log-likelihood under the Normal distribution. The log-likelihood of a sequence of observations $\hat{\mathbf{y}}$ under a multi-variate normal distribution with mean $\mu = \mathbf{y}$ and covariance $\Sigma = \sigma^2 I$ (where I stands for the identity matrix) is given by

$$LL_{\text{Normal}}\left(\hat{\mathbf{y}}|\hat{\mathbf{y}} \sim \mathcal{N}\left(\mu = \mathbf{y}, \Sigma = \sigma^2 I\right)\right) = -\frac{N}{2}\ln(2\pi) - \frac{N}{2}\ln(\sigma^2) - \frac{1}{2\sigma^2}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2. \tag{3.49}$$

Setting model parameters to MLE estimates under a Normal likelihood centered at model predictions and with fixed variance (shown in Equation (3.49)) is equivalent to setting the parameters by minimizing the MSE loss function (shown in Equation (3.48)).

Poisson Regression

The evaluation of Poisson (count) regression problems can be conducted as for regression, using mean squared error. However, as we mentioned, MSE corresponds to evaluating likelihood of the predictions under a Normal distribution. For evaluating count predictions it may be more appropriate to use a distribution allowing for only integer values, such as a Poisson distribution. Alternatively, the predictive distribution from a model can be used for finding the likelihood of the true count. In Chapters 5 and 6 we use both MSE and predictive distributions from models (which typically are Poisson) for evaluating Poisson regression problems.

Multi-class Classification Accuracy and F1 scores are popular metrics for evaluation of classification problems, which we make use of in our work on rumour stance classification. Both of the measures rely on precision and recall,

$$\text{Precision}_k = \frac{\text{tp}_k}{\text{tp}_k + \text{fp}_k}, \tag{3.50}$$

$$\text{Recall}_k = \frac{\text{tp}_k}{\text{tp}_k + \text{fn}_k}, \tag{3.51}$$

where tp_k (true positives for class k) refer to the number of instances correctly classified into class k, fp_k (false positives for class k) is the number of instances incorrectly classified into class k, and fn_k (false negatives for class k) is the number of instances that actually belong to class k but were not classified as such. Aggregate precision and recall for a c-class classification problem can be calculated either via microaveraging:

$$\text{Precision}_{\text{micro}} = \frac{\sum_{k=1}^{c} \text{tp}_k}{\sum_{k=1}^{c} \text{tp}_k + \sum_{k=1}^{c} \text{fp}_k}, \tag{3.52}$$

$$\text{Recall}_{\text{micro}} = \frac{\sum_{k=1}^{c} \text{tp}_k}{\sum_{k=1}^{c} \text{tp}_k + \sum_{k=1}^{c} \text{fn}_k}, \tag{3.53}$$

or macroaveraging:

$$\text{Precision}_{\text{macro}} = \frac{\sum_{k=1}^{c} \text{Precision}_k}{c}, \tag{3.54}$$

$$\text{Recall}_{\text{macro}} = \frac{\sum_{k=1}^{c} \text{Recall}_k}{c}. \tag{3.55}$$

Notice that microaveraged precision is equal to microaveraged recall. This is due to the fact that the number of false positives in the multi-class classification settings is equal to

the number of false negatives.[6] After computing either microaveraged or macroveraged precision and recall, the final F1 score is computed as the harmonic mean,

$$\text{F1} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}. \tag{3.56}$$

3.4 Conclusions

In this chapter we have introduced the machine learning frameworks we extensively use in the thesis. First, we described Gaussian processes, a Bayesian framework allowing for approaching a wide range of applications, including regression, classification and Poisson regression. It has multiple appealing characteristics, such as explicit modeling of uncertainty, and efficient hyperparameter optimization framework. We make use of Gaussian processes for building models of rumours in Chapters 4 to 6. Next, we moved to point processes, the framework for modeling sets of points occurring over some space (in our case it is defined over the space of time), which can be used to answer a wide range of questions, such as: *How many points will occur in a given interval of time?* or *When will the next point occur?* We described two point process models. Firstly, the log-Gaussian Cox process, which assumes that point occurrences in disjoint subspaces are independent, conditioned on the intensity function drawn from an exponentiated Gaussian process model. Secondly, the Hawkes process, which makes the assumption that occurrences of points increase the likelihood of points happening soon afterwards. We use point processes for modeling temporal dynamics of rumours in Chapters 5 to 7. In the following chapter we develop a Gaussian process model for rumour stance classification.

[6] If a particular instance is wrongly classified, then it contributes 1 to both the number of false positives (because it does not belong to the class it was classified to) and to the number of false negatives (because it was wrongly classified as not belonging to the true class). If a particularly instance is correctly classified, this contributes 0 to both false positives and false negatives.

Chapter 4

Rumour Stance Classification

In the preceding chapters we reviewed literature on rumours in social media, and introduced the machine learning frameworks we make use of in the experiments. In this chapter, we set out to address the first aim specified in Chapter 1: **predicting stance of tweets regarding rumours**. In particular, we work towards answering the first two research questions:

1. **What would be a realistic and fair evaluation framework for rumour stance classification?**

   which we address in Section 4.2. We motivate new evaluation settings, which satisfy desirable properties.

2. **Can information about stances of tweets from one rumour be useful for predicting stances of tweets from another rumour?**

   which we address in Section 4.5. We model rumour stances across multiple rumours using a Gaussian process with a multi-task learning kernel. We find that this method outperforms the Gaussian process model with a single-task kernel, and is competitive with other single-task learning approaches.

Parts of this chapter have been published as

> Lukasik, M., Cohn, T., and Bontcheva, K. (2015a). Classifying tweet level judgements of rumours in social media. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

4.1 Introduction

There is an increasing need to interpret and act upon rumours spreading quickly through social media during breaking news, where new reports are released piecemeal and often

have an unverified status at the time of posting. Previous research has emphasized the damage that the diffusion of rumours can cause in society, and that corrections issued by news organisations or the police may not necessarily achieve the desired effect sufficiently quickly (Lewandowsky et al., 2012; Procter et al., 2013a). Being able to quickly analyze rumours is therefore crucial in these scenarios.

Determining the stance of social media posts automatically has been attracting increasing interest in the scientific community in recent years, as this is a useful first step towards more in-depth rumour analysis. Rumour stance classification has been used for monitoring the community reactions, as well as in supporting downstream applications: it has been used to flag rumours which are particularly doubtful (Derczynski et al., 2015; Liu et al., 2015), and to classify events into rumour and non-rumour categories (Zhao et al., 2015b).

Work on automatic rumour stance classification, however, has shortcomings, with some works assuming an unrealistic evaluation scenario where all tweets are treated as coming from a single rumour, and evaluation is conducted on the past tweets while trained on the future (e.g. Qazvinian et al. (2011)). Moreover, previous approaches were modeling tweets from different rumours as a single rumour, i.e., no attempt has been made to model varying characteristics of tweets coming from different rumours (Qazvinian et al., 2011; Liu et al., 2015; Zhao et al., 2015b; Zeng et al., 2016b). In comparison to the previous work on rumour stance classification, in this chapter we report insights about this application in the following aspects:

1. We move away from evaluation based on a simple cross-validation ignoring the time of tweet occurrences. Instead, we perform stance classification on *unseen rumours*, given a training set of already annotated rumours on different topics. Additionally, we run experiments with a small number of initial tweets from the target rumour being available for the classifier during training, and evaluating it on the future tweets.

2. We report novel work using a *rumour identity* feature (i.e. a feature allowing to discriminate between tweets coming from different rumours), and show it is an important source of information. We base our model on a hypothesis that rumours exhibit similar characteristics (Procter et al., 2013b), however may differ. When using external rumours for making predictions about a target rumour, our multi-task learning approach uses the similarities between rumours, while filtering out rumour-specific characteristics.

3. Work described in this chapter was the first to consider a three category problem: supporting, denying and questioning, as published in Lukasik et al. (2015a). Previous

work considered this task in binary classification settings dealing with a more coarse-grained setting (Qazvinian et al., 2011).

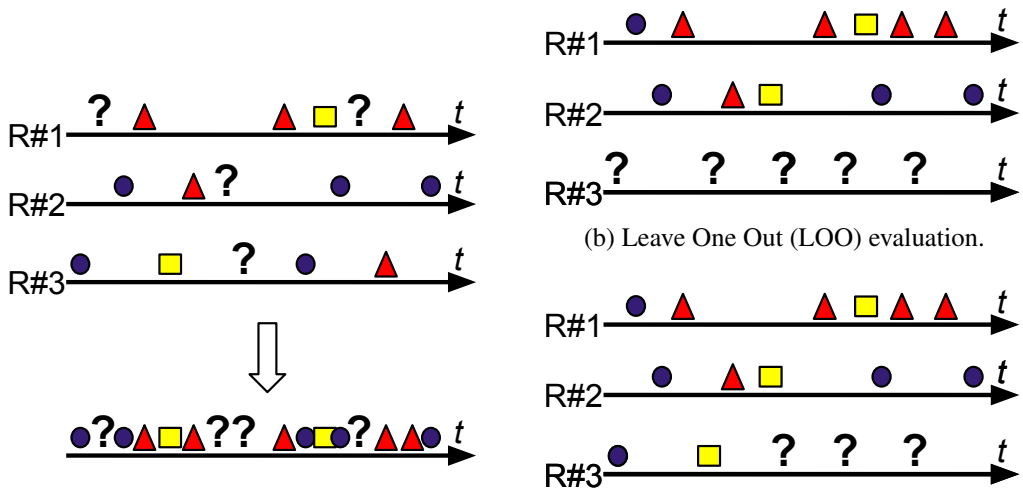
4.2 Problem Definition

We introduced the rumour stance classification task in Section 2.2. Here, we define it more formally, and discuss the evaluation scheme. Let $R = \{R_1, \ldots, R_{|R|}\}$ be a set of rumours, each of which consists of posts (tweets) discussing it, $\forall_{m=1,\ldots,|R|}\ R_m = \{\mathbf{p}_m^1, \ldots, \mathbf{p}_m^{|R_m|}\}$. $P = \cup_{m=1,\ldots,|R|} R_m$ is the complete set of tweets from all rumours. Each tweet is classified as supporting, denying or questioning with respect to its rumour: $\forall_{\mathbf{p}\in P}\ y(\mathbf{p}) \in Y = \{\text{supporting, denying, questioning}\}$.[1] We defined the rumour stances in Section 2.2 (notice that here, similarly to previous work, we do not include the commenting stance; see Chapter 7 for experiments including all stances).

Previous work evaluated the rumour stance classification task using cross-validation. In this approach the set of all tweets P is randomly split among K folds (Qazvinian et al. (2011) used $K = 5$), and iteratively each fold is used as a test set, and the remaining $K - 1$ folds serve as a training set. In Figure 4.1a we show an illustration of one fold in this setting, with question marks denoting tweets from the test set and other symbols denoting labels from the training set. Notice how training tweets occur after the test tweets within the same rumour, a scenario which does not occur in real world settings where journalists are interested in obtaining stances expressed in the most recent tweets. Ultimately, the separation of rumours and time dependencies were ignored in evaluation of previous work. Here, we deal with the task differently, arguing that the evaluation from previous work does not correspond to a real world scenario. In applications one should be able to classify new, emerging rumours, which can differ from what the classifier has observed in the training set.

We formulate the problem in settings which better reflect the real world scenario. First, we consider the Leave One Out (LOO) setting, in which for each rumour $R_m \in R$ we construct the test set equal to R_m and the training set equal to $P \setminus R_m$. This is the most challenging scenario, where the test set contains an entirely unseen rumour. We depict it in Figure 4.1b. Thus, we apply the method to each rumour separately. Ultimately, we consider the rumour stance classification problem as a form of transfer learning and seek to classify unseen rumours by training the classifier on previously annotated rumours. We argue that this makes for a more realistic classification scenario towards implementing a real-world rumour-tracking system. This scenario, which we introduced as published in Lukasik et al.

[1] Recall our notation is summarized in the Nomenclature on page xii.



(a) Cross validation.

(b) Leave One Out (LOO) evaluation.

(c) Leave Part Out (LPO) evaluation.

Figure 4.1: Illustration of different evaluation techniques for rumour stance classification. Different symbols correspond to tweets from one of the rumours which occurred at a specific point of time. Question marks denote the tweets that need to be classified in the test phase, other symbols denote observed classes of tweets in the training set (blue circles denote supporting tweets, yellow squares denote questioning tweets, red triangles denote rejecting tweets). The evaluation from previous work ignores rumour identities and time dependencies between tweets (Figure 4.1a), conflating all rumours into one (shown at the bottom line). In our approach, we focus on predicting labels for tweets from a left-out rumour strictly into the future (Figures 4.1b and 4.1c).

(2015a), has been adopted by the community in the follow up work on the task (Zeng et al., 2016b; Zubiaga et al., 2016a).

The second setting is Leave Part Out (LPO). Here, a number of initial tweets from the target rumour R_m is added to the training set $\{\mathbf{p}_m^1, \ldots, \mathbf{p}_m^k\}$, as depicted in Figure 4.1c. This scenario becomes applicable typically soon after a rumour breaks out and journalists have started monitoring and analysing the related tweet stream. In our experiments, we consider $k \in \{10, 20, 30, 40, 50\}$.

Notice that in these settings future tweets can still be present in the training set as long as they come from reference (non-test) rumours, and as such are not strictly realistic. The riot events we consider are short-lived, with rumours of short lifespans (see Figure 2.1 for depiction of lifespans of multiple rumours from the Ferguson riots dataset). This results in rumours overlapping in time, and so keeping only non-overlapping past rumours would result in very little reference data being kept for training. Therefore, here we keep reference rumours regardless of when they occurred (an approach adopted in the follow-up work in the community (Zeng et al., 2016b; Zubiaga et al., 2016a)).

The tweet-level stance classification problem here assumes that tweets from the training set are already categorized into what rumours they discuss. This information can be acquired either via manual annotation as part of initial analysis by journalists, as is the case with our dataset, or automatically, e.g. using pattern-based rumour detection (Zhao et al., 2015b). Our method is then used to classify the stance expressed in each new tweet from the test set.

4.3 Model

Below we describe the model we use for our experimentation, the Gaussian processes for classification (GPC) model. We also list the baseline classifiers that we use for comparison.

Gaussian processes for Classification

In Section 2.2 we reviewed methods that have been used in previous work on rumour stance classification, including Random Forests and Logistic Regression. Here we consider a Gaussian process model as our main approach. It is non-parametric, which means that the complexity of the model grows with the number of available training examples (recall from Section 3.1 that the predictions are made based on kernel function evaluations between the test inputs and all the training inputs). This contrasts with the parametric models, which are described by a fixed set of parameters. We described and motivated Gaussian processes

in more detail in Section 3.1. Here, we describe the adjustments for rumour stance classification in the presence of multiple rumours.

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, following previous work (Preotiuc-Pietro et al., 2015). We tune hyperparameters by maximizing evidence of the model $p(\mathbf{y}|X)$, as explained in Section 3.1.5. Moreover, Gaussian processes for classification require resorting to approximation techniques. Here, we make use of Expectation Propagation, an approximation technique we described in Section 3.1.4.

Transfer Learning In the Leave-Part-Out (LPO) setting initial labelled tweets from the target rumour are observed in addition to labelled tweets from other rumours. We propose to weight 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) (Bonilla et al., 2008), which we described in Section 3.1.5. ICM parametrizes the kernel by a matrix which represents the extent of covariance between pairs of tasks. The complete kernel takes form of

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

where B is a square coregionalisation matrix, m and m' denote the tasks of the two inputs and k_{data} is a kernel for comparing inputs $\mathbf{x}$ and $\mathbf{x}'$ (here, linear). Thus, the similarity function between the two tweets is a product of inter-rumour similarity ($B_{m,m'}$) and a tweet similarity independent from the rumour identities (k_{data}). This allows transfer learning by weighting the importance of annotated tweets from the reference training rumours based on how similar the characterstics of the reference rumours are to that of a test rumour. We parametrize the coregionalisation matrix $B = \boldsymbol{\kappa} I + \boldsymbol{v}\boldsymbol{v}^T$, where $\boldsymbol{v} \in \mathbb{R}$ specifies the correlation between tasks and the vector κ controls the extent of task independence.

Model Settings We consider GPs in three settings, varying in what data the model is trained on and what kernel it uses. [2]

GP ONLY TARGET considers only target rumour data for training, and thus only uses the single task learning kernel. Notice that this model setting can not be considered in the LOO problem setting, as it would not have access to any training data.

[2] In the experiments we use the GPy implementation of Gaussian processes (The GPy authors, 2015).

GP considers both the target rumour data as well as the reference rumours data (i.e. other than the target rumour), however only uses the single task learning kernel.

GP-ICM considers both the target rumour data as well as the reference rumours data (i.e. other than the target rumour), and employs the ICM kernel for learning the similarities between different rumours.

Baselines We compare our model against baselines:

MAJORITY VOTE classifier based on the training label distribution.

MAXENT Logistic Regression was the first method employed for rumour stance classification (Qazvinian et al., 2011). We use ℓ_1 regularisation with the cost coefficient selected from the list: [0.00001, 0.0001, 0.001, 0.01, 0.1, 1, 10, 100]. The cost coefficient was found using grid search employing 3-fold cross-validation over the training set, where two folds were used for training and one for evaluation of the proposal coefficient.

SVM Support Vector Machines with the cost coefficient selected via nested cross-validation from the list of values: [0.00001, 0.0001, 0.001, 0.01, 0.1, 1, 10, 100]. The cost coefficient was found using grid search employing 3-fold cross-validation over the training set.

RF Zeng et al. (2016b) found Random Forests to be the best approach in their experiments with rumour stance classification. The authors report the value of only one hyperparameter value, namely they set the number of trees to 30, although they do not state whether it is chosen via hyperparameter optimization. A Random Forests classifier is controlled by a number of hyperparameters, which we select via grid search over the cross product between the considered hyperparameter values (employing 3-fold cross-validation over the training set). The hyperparameters that we consider are: the splitting criterion measuring the quality of a split (optimized for from the list [Gini impurity, entropy]), the number of trees (optimized for from the list [10, 50, 100, 150, 200]), the minimum number of samples in a node to perform a split (optimized for from the list [2, 5, 10]). We use bootstrap samples when choosing data for each tree.

We use Scikit-learn implementations of the baseline classifiers (Pedregosa et al., 2011).

4.4 Experiment Settings

This section details the features and evaluation metrics used in our experiments on tweet level stance classification.

Features For experiments, we use the England riots and the PHEME datasets, described in detail in Section 2.5. In the case of either of the datasets, we conducted a series of preprocessing steps in order to address data sparsity. All words were converted to lowercase; stopwords have been removed;[3] all emoticons were replaced by words;[4] and stemming was performed. In addition, multiple occurrences of a character were replaced with a double occurrence (Agarwal et al., 2011), to correct for 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 preprocessing the text data, we either consider bag of words (BOW) word representation, or replace all words with their Brown cluster ids (Brown). Brown clustering is a hard hierarchical clustering method (Liang, 2005). 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 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. More details on the Brown clusters that we used as well as the words that are part of each cluster are available online (Owoputi et al., 2013).[5] Since we find Brown clusters to perform better than BOW (as shown in Table 4.1), unless explicitly noted we will mean Brown clusters feature representation when reporting experimental results.

Evaluation Metrics Preceeding work on rumour stance classification considered a binary classification setting, and employed accuracy and F1 evaluation metrics (Qazvinian et al., 2011). Here, we deal with the multi-class classification scenario, thus these metrics are not directly applicable. Moreover, as shown in the statistics about distributions of stances in our datasets (see Table 2.1 for information about the England Riots dataset, and Table 2.6 for information about the PHEME dataset) the classes are imbalanced, with varying proportions of stances in each rumour. In order to gain insights into model performance we employ multiple evaluation metrics: micro-averaged and macro-averaged F1 scores, described in Section 3.1.3. After computing the F1 score for each fold (corresponding to a rumour), we compute the averaged score across folds.

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

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

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

4.5 Results

In this section we report results of experiments on rumour stance classification. We analyze the performance of different methods, and draw conclusions about the task. We report Micro-F1 and Macro-F1 metrics as described in Section 3.3, which provide insights about two aspects of model performance: how well it classifies tweets overall (i.e. minimizing the absolute number of errors), and how well it balances the errors for different stances.

Tables 4.1 and 4.2 report the combined results in both the LOO and LPO settings. Consecutive columns correspond to an increasing number of tweets from the target rumour available during training (column 0 corresponds to the LOO setting, and other columns correspond to the LPO setting). In Table 4.1 we show results for the GP based models using different text representations. Notice that in the case of both England riots and PHEME datasets, Brown clusters make for a more robust text representation. Brown clusters always yield better results on the PHEME dataset according to both Micro-F1 and Macro-F1 scores. Moreover, on the England riots dataset, Brown clusters always lead to a better Macro-F1 score, and to competitive Micro-F1 scores. Thus, in the following analysis we report baselines using the more promising text representation employing Brown clusters. We discuss the relative performance of different GP settings in the following sections.

Experiments on the England Riots dataset In Table 4.2a we report micro-averaged and macro-averaged F1 scores of methods' performance on the England riots dataset as the number of tweets from the target rumour used for training increases (this information is graphically illustrated in Figures 4.2a and 4.2b). Notice how performance of *GP Only Target* is significantly lower than that of *GP* and *GP-ICM*, showing the importance of using additional data from reference rumours. We can notice that the performance of most of the methods improves as the proportion of annotated training examples from the target rumour increases. This phenomenon is especially noticeable for the GP-ICM method. Notice that when no annotation from the target rumour is used, its performance is poor in terms of micro-averaged F1 score. However, it is able to make very effective use of the annotation. Its performance keeps improving as the number of training instances approaches 50, and overtakes the baselines after 20 annotated examples. This shows GP-ICM is able to make use of the labelled instances from the target rumour, which the baselines struggle with. Note that 50 tweets represent, on average, less than 7% of the whole rumour, with the rest of the rumour unobserved during training. Moreover, notice how regardless of the number of labelled instances, GP-ICM yields good results in terms of macro-averaged F1 score. This shows that GP-ICM balances between the errors made for each stance better than other

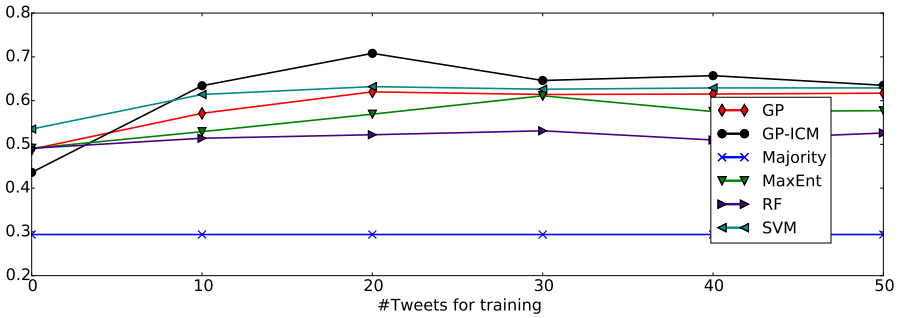
		0	10	20	30	40	50
Macro-F1	GP Only Target Brown	N/A	0.346	0.366	0.366	0.382	0.416
	GP Only Target BOW	N/A	0.314	0.346	0.379	0.388	0.402
	GP Brown	**0.489**	0.571	0.620	0.614	0.615	0.617
	GP BOW	0.452	0.572	0.603	0.593	0.588	0.616
	GP-ICM Brown	0.436	**0.634**	**0.708**	**0.646**	**0.657**	**0.635**
	GP-ICM BOW	0.394	0.510	0.585	0.544	0.574	0.562
Micro-F1	GP Only Target Brown	N/A	0.787	0.722	0.733	0.735	0.769
	GP Only Target BOW	N/A	0.781	0.710	0.760	0.751	0.775
	GP Brown	0.614	0.737	0.765	0.761	0.762	0.763
	GP BOW	**0.640**	0.817	0.825	0.818	0.811	**0.833**
	GP-ICM Brown	0.540	0.812	**0.855**	**0.829**	**0.833**	0.828
	GP-ICM BOW	0.476	**0.821**	0.806	0.809	0.811	0.822

(a) England riots

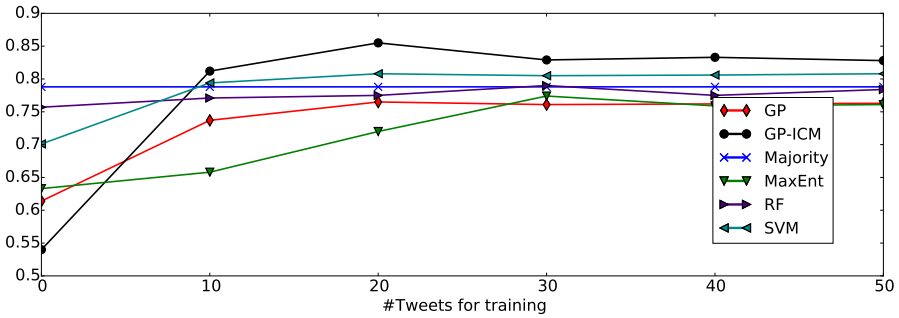
		0	10	20	30	40	50
Macro-F1	GP Only Target Brown	N/A	0.434	0.489	0.494	0.514	0.515
	GP Only Target BOW	N/A	0.356	0.382	0.399	0.415	0.439
	GP Brown	0.548	**0.555**	0.569	0.566	0.567	0.575
	GP BOW	0.465	0.472	0.475	0.477	0.471	0.481
	GP-ICM Brown	**0.557**	**0.555**	**0.592**	**0.575**	**0.594**	**0.598**
	GP-ICM BOW	0.453	0.465	0.455	0.439	0.466	0.471
Micro-F1	GP Only Target Brown	N/A	0.546	0.577	0.612	0.606	0.613
	GP Only Target BOW	N/A	0.591	0.548	0.554	0.546	0.558
	GP Brown	0.631	**0.636**	0.644	0.644	0.645	0.650
	GP BOW	0.551	0.569	0.572	0.575	0.572	0.579
	GP-ICM Brown	**0.655**	0.635	**0.652**	**0.655**	**0.668**	**0.675**
	GP-ICM BOW	0.561	0.579	0.587	0.578	0.580	0.577

(b) PHEME

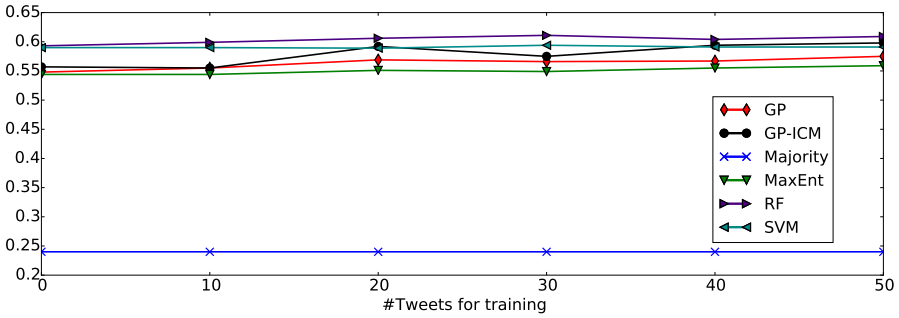
Table 4.1: Micro-F1 and Macro-F1 scores for GP based methods under different settings using different word representation methods (Brown clusters and BOW; denoted by rows) and different proportions of the initial tweets annotated from the target rumour/event on the England riots and the PHEME datasets (denoted by columns).



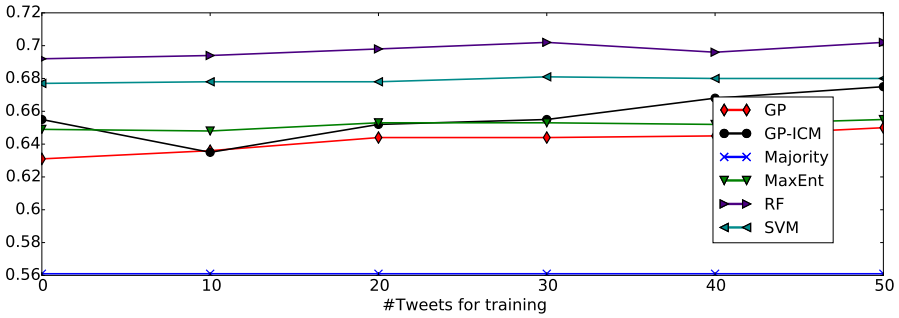
(a) Macro-F1 on the England riots



(b) Micro-F1 on the England riots



(c) Macro-F1 on the PHEME



(d) Micro-F1 on the PHEME

Figure 4.2: Macro-F1 and Micro-F1 scores for different methods over the number of tweets from the target rumour used for training on the England riots and the PHEME datasets. The test set is fixed to all but the first 50 tweets of the target rumour, making the results comparable across the varying training size.

		0	10	20	30	40	50
Macro-F1	Majority	0.294	0.294	0.294	0.294	0.294	0.294
	GP Only Target	N/A	0.346	0.366	0.366	0.382	0.416
	GP	0.489	0.571	0.620	0.614	0.615	0.617
	GP-ICM	0.436	**0.634**	**0.708**	**0.646**	**0.657**	**0.635**
	MaxEnt	0.491	0.529	0.569	0.611	0.575	0.577
	SVM	**0.535**	0.614	0.632	0.626	0.629	0.629
	RF	0.491	0.514	0.522	0.531	0.510	0.526
Micro-F1	Majority	**0.788**	0.788	0.788	0.788	0.788	0.788
	GP Only Target	N/A	0.787	0.722	0.733	0.735	0.769
	GP	0.614	0.737	0.765	0.761	0.762	0.763
	GP-ICM	0.540	**0.812**	**0.855**	**0.829**	**0.833**	**0.828**
	MaxEnt	0.633	0.658	0.720	0.774	0.759	0.761
	SVM	0.701	0.794	0.808	0.805	0.806	0.808
	RF	0.757	0.771	0.775	0.790	0.775	0.784

(a) England riots

		0	10	20	30	40	50
Macro-F1	Majority	0.240	0.240	0.240	0.240	0.240	0.240
	GP Only Target	N/A	0.434	0.489	0.494	0.514	0.515
	GP	0.548	0.555	0.569	0.566	0.567	0.575
	GP-ICM	0.557	0.555	0.592	0.575	0.594	0.598
	MaxEnt	0.544	0.544	0.551	0.549	0.555	0.559
	SVM	0.590	0.590	0.589	0.594	0.591	0.591
	RF	**0.593**	**0.599**	**0.606**	**0.611**	**0.604**	**0.609**
Micro-F1	Majority	0.561	0.561	0.561	0.561	0.561	0.561
	GP Only Target	N/A	0.546	0.577	0.612	0.606	0.613
	GP	0.631	0.636	0.644	0.644	0.645	0.650
	GP-ICM	0.655	0.635	0.652	0.655	0.668	0.675
	MaxEnt	0.649	0.648	0.653	0.653	0.652	0.655
	SVM	0.677	0.678	0.678	0.681	0.680	0.680
	RF	**0.692**	**0.694**	**0.698**	**0.702**	**0.696**	**0.702**

(b) PHEME

Table 4.2: Micro-F1 and Macro-F1 scores for different methods and different proportions of the initial tweets annotated from the target rumour/event on the England riots and the PHEME datasets. All methods use Brown clusters word representations. The results are also pictorially shown in Figure 4.2.

models. Lastly, we notice that SVM achieves competitive results that are above the rest of the baselines, outperforming GP on both metrics. Notice that only GP-ICM and SVM are able to consistently beat the Majority classifier once 20 tweets from the training rumour are available for training.

Experiments on the PHEME dataset In Table 4.2b we report micro-averaged and macro-averaged F1 scores of methods' performance on the PHEME dataset as the number of tweets from the target rumour used for training increases (this information is graphically illustrated in Figures 4.2c and 4.2d). Notice the results are overall lower than in the case of the England Riots dataset. The reason for this is two fold. First, in the PHEME dataset we deal with a more challenging setting, where at each fold of the evaluation we leave out an event out (where an event is composed of rumours), and train on other events. Instead, in the England riots case we were leaving one rumour out within the same event. Secondly, the PHEME dataset is largely composed of tweets that are replying to others (Zubiaga et al., 2016c), which makes them shorter. As shown in Table 2.1 and Table 2.5 rumours from the PHEME dataset are much shorter than rumours from the England riots dataset, and hence more challenging to get meaningful features from. Despite these difficulties we are interested in exploring if similar trends hold across classifiers.

One difference from the England Riots results is that, in this case, the classifiers are not benefiting as much from incorporating increasing number of annotated tweets from the target rumour. This is likely due to the heterogeneity of the events from the PHEME dataset. Namely, within each event there are multiple rumours, and as the number of initial tweets is annotated, we are gaining insight into a diverse set of rumours as they start to unfold. All of these rumours are different, and they are not necessarily covering all rumours from the target event. By contrast, each left out set of tweets in the England Riots pertains to a single rumour, and so annotating its initial tweets is useful, giving insights into characteristics about that rumour.

We observe that Random Forests turn out to be the best approach according to both metrics. Interestingly, the second best method is different depending on which metric we consider. According to Macro-F1, GP-ICM and SVM are both competitive, with GP-ICM being slightly better with larger supervision. However, under Micro-F1, SVM is clearly the second best approach. Similarly to the England Riots results, the performance of *GP Only Target* is significantly lower than that of *GP* and *GP-ICM*, showing that reference rumour annotations is crucial for the GPs to achieve competitive results (we omit *GP Only Target* from the graphs for better visualization of differences between the best performing models). Moreover, *GP-ICM* outperforms *GP*, which shows that the multi-task learning kernel brings

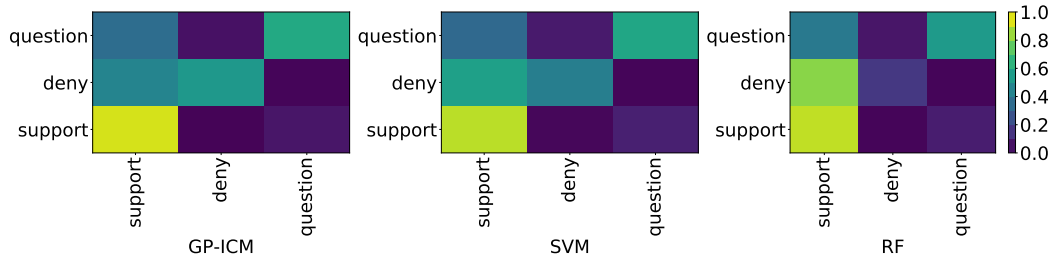


Figure 4.3: Cross-classification rates for competitive methods on the England riots dataset. A cell i, j denotes what percentage of times the ground truth stance i is being classified as stance j. The statistics are also reported in Table 4.3.

improvements within the same model.

Analysis of the Best Performing Methods Next, we analyze the results of the best-performing classifiers (GP-ICM, SVM and RF) by looking at the per-class performance. Table 4.3 reports per-class F1 scores for the three best performing classifiers for the England riots dataset and the PHEME dataset in LPO settings where 20 tweets from a target rumour are available during training. The table also reports statistics on the misclassifications that the approaches made (the cross-stance classifications are also depicted graphically for the case of England riots in Figure 4.3).

Notice that in the case of the England riots, GP-ICM is a clear winner. It is the only approach which manages to retrieve more than 50% of denies, which is one of the two under-represented stances (denies are around 6 times less frequent than supports, whereas questions are around 10 times less frequent than supports, as reported in Table 2.1). Interestingly, the questioning stance is easier to correctly classify than the denying stance across the methods, even though it is even less frequent.

In the case of the PHEME rumours, GP-ICM is again the best classifier in terms of retrieving the denies. This is an interesting property, as denying is the most challenging stance (challenging in the sense of yielding the worst results across all methods). However, for both supporting and questioning stances RF and SVM make better predictions.

The problem of misclassifying denies is due to the datasets' imbalance, which is a common problem in the rumour stance distribution, as previous studies have shown that users rarely deny or question rumours, but instead largely support rumours regardless of whether they are true or false (Zubiaga et al., 2016c).

Discussion We experimented with two rumour datasets, and adapted the introduced evaluation schemes differently to each of them. In the case of the first dataset we were making

England Riots							
Class	Classifier	Performance			Cross-classification rates		
		P	R	F1	S	D	Q
supporting (S)	GP-ICM	**0.893**	**0.935**	**0.914**	0.935	0.008	0.057
	RF	0.833	0.903	0.867	0.903	0.014	0.082
	SVM	0.873	0.896	0.884	0.896	0.019	0.086
denying (D)	GP-ICM	**0.882**	**0.535**	**0.666**	0.452	0.535	0.013
	RF	0.584	0.162	0.254	0.823	0.162	0.015
	SVM	0.742	0.423	0.539	0.563	0.423	0.014
questioning (Q)	GP-ICM	**0.496**	**0.602**	**0.544**	0.352	0.045	0.602
	RF	0.380	0.540	0.446	0.403	0.057	0.540
	SVM	0.394	0.593	0.473	0.339	0.068	0.593
PHEME							
Class	Classifier	Performance			Cross-classification rates		
		P	R	F1	S	D	Q
supporting (S)	GP-ICM	**0.748**	0.767	0.757	0.767	0.133	0.101
	RF	0.714	**0.899**	**0.796**	0.899	0.053	0.047
	SVM	0.706	0.865	0.777	0.865	0.071	0.064
denying (D)	GP-ICM	0.442	**0.385**	**0.412**	0.428	0.385	0.187
	RF	**0.570**	0.277	0.373	0.594	0.277	0.129
	SVM	0.511	0.259	0.344	0.604	0.259	0.137
questioning (Q)	GP-ICM	0.588	**0.625**	0.606	0.233	0.141	0.625
	RF	**0.708**	0.601	**0.650**	0.329	0.071	0.601
	SVM	0.676	0.618	0.646	0.318	0.064	0.618

Table 4.3: Per-class precision, recall and F1 scores for the best-performing classifiers (GP-ICM, SVM and RF) on the England Riots and the PHEME datasets with 20 tweets from a target rumour available during training. Cross-classification rates denote how often the target stance (denoted by the row) is being misclassified as another stance (denoted by the column). Figure 4.3 graphically illustrates the cross-classification rates for the top approaches on the England riots dataset.

predictions for single held out rumours from the England riots event from 2011, thus dealing with the setting where all the rumours are revolving around the same background event. In the case of the second dataset, we were making predictions for each of the five different events, having access to the remaining four, making for a significantly more challenging setup. In these different settings, we made various observations regarding the relative performance of different approaches. We observed that while a GP trained only on the target data is not achieving competitive results, the GP using reference examples (the scenario of all other baselines) achieves better performance, and its multi-task learning variant GP-ICM leads to additional improvements leading to outperforming the baselines in the case of the England riots dataset. Moreover, in the case of both datasets, GP-ICM manages to perform relatively well in classifying the denying stance, which turns out to be the most challenging. Another appealing aspect of GP-ICM in the England riots dataset is that it performs well despite having very few annotations from the target rumour, making better use of such training data than the baselines. However, we notice that when annotation comes from external events (which is the case for the PHEME dataset experiments), Random Forests and SVMs are competitive with the GP-ICM approach. This poses a question of whether multi-task learning variants of Random Forests or SVMs (Evgeniou and Pontil, 2004) could bring further improvements, as we found the multi-task learning Gaussian process model (GP-ICM) to consistently outperform the single-task learning GP across all settings.

4.6 Conclusions

This chapter investigated the problem of classifying stances expressed in tweets about rumours. We investigated realistic scenarios of evaluating the problem, as we set out in research question 1.1. First, we considered a setting where no training data from the target rumours is available (LOO). Without access to annotated examples of the target rumour the learning problem proved to be challenging. We showed that in the supervised domain adaptation setting (LPO), annotating even a small number of tweets helps to achieve better results. Moreover, we demonstrated the benefits of a multi-task learning approach by showing how GPs with a multi-task learning kernel consistently outperforms GPs with a single-task learning kernel. This shows how the reference rumours can provide valuable information for making predictions about the target rumour, as we set out to investigate in research question 1.2. The next two chapters consider the second rumour application of this thesis, rumour popularity prediction. We come back to the rumour stance classification application in Chapter 7, where we investigate whether the ideas from the rumour popularity modeling can improve results on this application.

Chapter 5

Modeling Temporal Dynamics of Rumours

In the preceding chapter we considered the problem of rumour stance classification. In this chapter, we move to a different application concerning rumours in social media. Namely, we set out to address the second aim specified in Chapter 1: **predicting rumour popularity**. In particular, we work towards answering the first three research questions:

1. **How can the rumour popularity prediction problem be formulated?**

   which we address in Section 5.2. We define the task as predicting the number of tweets about the rumour during unobserved intervals of time from the rumour lifespan.

2. **Can information about tweet arrivals from one rumour bring useful information for predicting the popularity of another rumour?**

   which we address in Sections 5.3 and 5.5. We model rumour popularity across multiple rumours using a log-Gaussian Cox process with a multi-task learning kernel, the Intrinsic Coregionalization Model. We find that this approach yields better predictions than non-multi-task learning approaches.

3. **Does the text content of tweets convey useful information for the rumour popularity prediction task?**

   which we address in Sections 5.3 and 5.5. We explore a multi-task learning approach where similarities between rumours are parameterized by text content from tweets aggregated over the rumour. We find that employing text allows for achieving superior results compared to baselines.

Parts of this chapter have been published as

Lukasik, M., Cohn, T., and Bontcheva, K. (2015b). Point process modelling of rumour dynamics in social media. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (ACL-IJCNLP)*, pages 518–523.

and

Lukasik, M., Srijith, P. K., Cohn, T., and Bontcheva, K. (2015c). Modeling tweet arrival times using log-Gaussian Cox processes. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 250–255.

5.1 Introduction

The ability to model rumour dynamics helps with identifying rumours, which if not debunked early, are likely to spread very fast. One such example is the false rumour of rioters breaking into McDonald's during the 2011 England riots. An effective early warning system of this kind is of interest to government bodies and news outlets, who struggle with monitoring and verifying social media posts during emergencies and social unrests. The challenge comes from the observation that different rumours exhibit different trajectories, as in the case of the Ferguson rumours depicted in Figure 2.1. Online discussion of some rumours quickly fades, whereas for other it takes a lot longer to diminish. Two characteristics can help determine if a rumour will continue to be discussed. One is the dynamics of post occurrences, e.g. if the frequency profile decays quickly, chances are it would not attract further attention. A second factor is text from the posts themselves, where phrases such as *not true*, *unconfirmed*, or *debunk* help users judge veracity and thus limit rumour spread (Zhao et al., 2015b).

This chapter considers the problem of modelling temporal frequency profiles of rumours by taking into account both the temporal and textual information. Since posts occur at continuous timestamps, and their density is typically a smooth function of time, we base our model on point processes, which have been shown to model well such data in epidemiology and conflict mapping (Brix and Diggle, 2001; Zammit-Mangion et al., 2012). This framework models count data in a continuous time through the underlying intensity of a Poisson distribution. The posterior distribution can then be used for several inference problems, e.g. to query the expected count of posts, or to find the probability of a count of posts occurring during an arbitrary time interval. We model frequency profiles using a log-Gaussian Cox process (Møller and Syversveen, 1998), a point process where the log-intensity of the Poisson distribution is modelled via a Gaussian process (see Chapter 3 for details about

Gaussian processes).

Modeling the frequency profile of a rumour based on posts is challenging, since many rumours consist of only a small number of posts and exhibit complex patterns. To overcome this difficulty, we propose multi-task learning approaches, where patterns are correlated across multiple rumours. In this way statistics over a larger training set are shared, enabling more reliable predictions for distant time periods, in which no posts from the target rumour have been observed. We considered one multi-task learning approach in Chapter 4 in the context of rumour stance classification. Here, apart from employing the ICM multi-task learning model, which we have considered for rumour stance classification, we also demonstrate how text from observed posts can be used to weight similarities between rumours. This way, we incorporate textual information about rumours into the rumour popularity prediction task.

We also introduce the problem of predicting the times of tweets in the unobserved time intervals. It turns out that this problem is closely related to modelling the frequency profiles. We evaluate the models using Twitter rumours from the 2014 Ferguson unrest, and demonstrate that they provide good prediction of both rumour popularity and inter-arrival time, beating baselines, e.g. homogeneous Poisson Process and Gaussian process regression.

This chapter makes the following contributions:

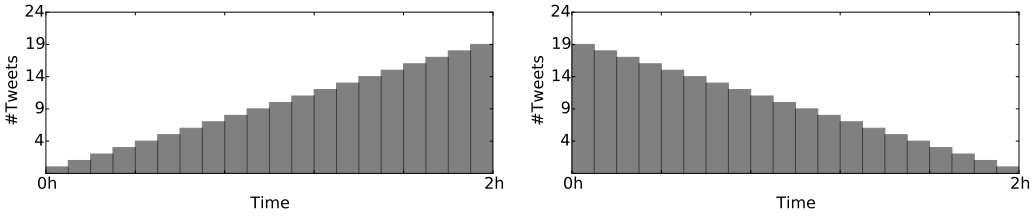
1. Introduces the problem of modelling rumour popularity, and presents a method based on a log-Gaussian Cox process;

2. Incorporates multi-task learning to generalize across disparate rumours;

3. Demonstrates how incorporating text into multi-task learning improves results.

5.2 Problem definition

First we define the rumour popularity prediction problem. One approach is to frame it as binary classification, where popular rumours and non-popular rumours are assigned to separate categories. However, in order to do that, one needs to define what a popular rumour is. Does one define a popular rumour as one that is going to attract a large number of tweets in the future, i.e., above some threshold? If so, what threshold should be chosen? Moreover, one might prefer to define a popular rumour based on how the number of tweets varies over time, rather than focusing on just a single aggregate statistic. For example, in Figure 5.1 two example rumour trajectories are shown. The total number of tweets around both is the same, however for one the number of tweets overall increases (5.1a), whereas about the



(a) A rumour with an increasing number of tweets per unit of time.

(b) A rumour with a decreasing number of tweets per unit of time.

Figure 5.1: Counts of tweets in consecutive 6 minute time intervals over two hours for two illustrative rumours. Number of tweets about one rumour increases (a), and about another decreases (b). The total number of tweets is equal across the two hours for both rumours.

other decreases (5.1b). The definition of rumour popularity based on the total number of tweets ignores this characteristics, which might be important for a journalist assessing the rumour popularity.

Instead, we decide to leave the interpretation of what a popular rumour is to journalists, and provide them with predictions which can be interpreted according to what one might be interested in. Namely, we are going to output predictions of how the number of tweets is going to change over time in the future, without assigning any label to this spread.

Let us consider a time interval $[0, l]$ of length l=2 hours, a set of $|R|$ rumours $R = \{R_m\}_{m=1}^{|R|}$, where rumour R_m consists of a set of $|R_m|$ posts $R_m = \{\mathbf{p}_m^n\}_{n=1}^{|R_m|}$. Posts are tuples $\mathbf{p}_m^n = (\mathbf{w}_m^n, t_m^n)$, where $\mathbf{w}_m^n$ is text (in our case a bag of Brown clusters text representation, see Section 5.4 for more details) and t_m^n is a timestamp describing when the post $\mathbf{p}_m^n$ happened, measured in time elapsed since the first post on rumour R_m (this way the post that happened first happened at time 0).[1] Posts occur at different timestamps, yielding varying density of posts over time, which we are interested in estimating. To evaluate the predictions for a given rumour R_m we hold out posts from a set of 10 non-overlapping 6-minute time intervals $T_{te} = \{[s_m^k, e_m^k]\}_{k=1}^{10}$ (where s_m^k and e_m^k are respectively start and end points of interval k for rumour m) and estimate performance at predicting counts in them by the trained model.

We consider the problem in supervised settings, where posts on this rumour outside of these intervals form the training set:

$$R_m^{\mathrm{O}} = \{\mathbf{p}_m^n : t_m^n \notin \bigcup_{k=1}^{10} [s_m^k, e_m^k]\}. \tag{5.1}$$

[1] Recall our notation is summarized in the Nomenclature on page xii.

We also consider a domain adaptation setting, where additionally posts from other rumours are observed: $P \setminus R_m$, yielding training set $R_m^{\text{O}} \cup (P \setminus R_m)$ (where P is the set of tweets from all rumours). More specifically, we are going to consider two instantiations of this problem formulation. The first is *interpolation*, where the test intervals are not ordered in any particular way. This corresponds to a situation, e.g., when a journalist analyses a rumour during short spot checks, but wants to know the prevalence of the rumour at other times, thus limiting the need for constant attention. The second formulation is that of *extrapolation*, where all observed posts occur before the test intervals. This corresponds to a scenario where the user seeks to predict the future profile of the rumour, e.g., to identify rumours that will attract further attention or wither away.

Notice the similarity to the problem formulation of rumour stance classification from section 4.2. The extrapolation setting is analogous to the LPO setting from the rumour stance classification experiments, where a number of initial tweets from a rumour are observed. The interpolation setting does not correspond to any evaluation scenario from Chapter 4, as here we allow for observation of tweets which happen after (some of) the unobserved intervals.

We also introduce the problem of predicting the exact time of posts in the future unobserved time interval, which is studied as *inter-arrival time prediction* (note we only consider this in the extrapolation setting). In this problem we observe posts $\{\mathbf{p}_m^n\}_{n=1}^{|R_m^O|}$ and query the model for a complete set of times $\{t_m^n\}_{n=|R_m^O|+1}^{|R_m|}$ of posts about rumour R_m in the future time interval (in our case for the future one hour). This can be viewed as a more fine-grained version of the previous problems, where we are not only interested in the counts in intervals, but rather in exact times of occurrences.

5.3 Model

We employ a log-Gaussian Cox process, a point process coupled with a GP prior, which has been described in section 3.2.4. LGCP formulates the model as $\lambda(t) = \exp(f(t))$, such that the function $f(t)$ is a sample from a Gaussian process. This way, a non-parametric prior is put on the hazard function, allowing for the expressivity of the model to grow together with the dataset. Below we describe the adjustments for the problem of rumour dynamics modeling. Namely, in order to exploit similarities across rumours we propose multi-task approaches, where each rumour represents a task. We consider two approaches, first based on the Intrinsic Coregionalization model (Bonilla et al., 2008; Álvarez et al., 2012), and the second based on text parametrization of rumour similarities. In either case, all hyperparameters are optimized by maximizing the marginal likelihood of the data $L(R_m^O|\theta)$, where θ

denotes the set of hyperparameter values, as explained in Section 3.1.5.

Multi-task learning via the Intrinsic Coregionalization Model We employ a multiple output GP based on the Intrinsic Coregionalization Model (ICM) which we described in Section 3.1.6 and employed in Chapter 4 for modeling rumour stances. ICM parametrizes the kernel by a matrix representing similarities between pairs of tasks. We expect it to find correlations between rumours exhibiting similar temporal patterns. The kernel takes the form

$$k_{\text{ICM}}((t,m),(t',m')) = k_{time}(t,t')B_{m,m'},$$

where B is a square coregionalization matrix (rank 1, i.e. $B = \boldsymbol{\kappa} I + \boldsymbol{v}\boldsymbol{v}^T$), m and m' denote the tasks of the two inputs, k_{time} is a kernel for comparing inputs t and t' (here RBF) and $\boldsymbol{\kappa}$ is a vector of values modulating the extent of each task independence.

Incorporating Text In Section 5.1 we argued that text might bring useful information for predicting rumour popularity. The question of how to incorporate text is actually not an easy one. One simple approach could be augmenting the list of arguments of a kernel by a dimension corresponding to text, leading to a kernel of a form $k_{\text{TXT}}((t,m,\text{text}),(t',m',\text{text}'))$. Such a kernel could be potentially evaluated when comparing timestamps from fully observed intervals of rumour lifespans. One way of obtaining text representation for timestamp t from rumour R_m could be collapsing text from posts around rumour R_m from some region of time corresponding to timestamp t. Another approach to finding text representation for time t could be taking text from the closest post from rumour R_m. These approaches are helpful for timestamps coming from intervals of time for which posts have been observed about a rumour of interest. However, in the problem as has been formulated in Section 5.2, we need to make predictions for intervals of time from which no posts have been observed. Therefore, simply expanding a list of arguments by text corresponding to posts around a particular timestamp is not feasible.

Instead, we propose to use text from all observed posts comprising each rumour. This way, we obtain the same text representation for a rumour across all timestamps. This approach ignores information about the timestamp when finding the text representation of a rumour. The kernel takes form

$$k_{\text{TXT}}((t,m),(t',m')) = k_{time}(t,t') \times k_{text}\left(\sum_{\mathbf{p}_m^n \in R_m^O} \mathbf{w}_m^n, \sum_{\mathbf{p}_{m'}^l \in R_{m'}^O} \mathbf{w}_{m'}^l\right).$$

We compare text via a linear kernel with additive underlying base similarity, expressed by

$k_{text}(\mathbf{w}, \mathbf{w}') = b + c\mathbf{w}^T\mathbf{w}'$. We also consider $k_{\text{ICM+TXT}} = k_{\text{ICM}} + k_{\text{TXT}}$, the combination of the two multi-task learning approaches. This corresponds to the kernel fusion scheme (Wu et al., 2005), where different views of time series are combined in a weighted average (kernel variances correspond to the weights).

Modelling inter-arrival time Predicting inter-arrival time between tweets around rumours could be applied for ranking rumours according to their activity. This in turn can serve for narrowing the interest only to the rumours for which tweets are predicted to have fresh posts the soonest. We are interested in predicting the next arrival time of a tweet given the time at which the previous tweet happened. As explained in Section 3.2.2, this can be achieved by sampling the inter-arrival time of occurrence of the next tweet using Equation (3.35). We use the importance sampling scheme (Gelman et al., 2003) as shown in Algorithm 3.1, where an exponential distribution is used as the proposal density (we set the scale parameter of this exponential distribution $\beta = 2$ in order to generate points close to 0). We run this algorithm sequentially until the end of the interval of interest, for which a user wants to find times of post occurrences.

5.4 Experiment Settings

Data We use the Ferguson rumour data set (Zubiaga et al., 2015a), as the one with the largest number of tweets per rumour from the PHEME rumour events as shown in table 2.5. It consists of tweets collected in August and September 2014 during the Ferguson unrest and contains both source tweets and the conversational threads around these (where available). Since some rumours have few posts, we consider only those with at least 15 posts in the first hour as rumours of particular interest. This results in 114 rumours consisting of a total of 4098 tweets. For more details about the dataset, see Section 2.5.2.

In our experiments, we consider the first two hours of each rumour lifespan, which we split into 20 evenly spaced intervals. This way, our dataset consists in total of 2280 intervals. In experiments we iterate over rumours using a form of folded cross-validation, where in each iteration we exclude some (but not all) time intervals for a single target rumour. The excluded time intervals form the test set: either by selecting half at random (interpolation); or by taking only the second half for testing (extrapolation). We consider the problem of predicting the time of tweets in the extrapolation settings, where the posts from the first hour are observed and we predict the tweets in the second hour. See section 5.2 for more details and motivation behind the evaluation.

We use the preprocessing for obtaining text representation of tweets as reported in Section 4.4 for the experiments with rumour stance classification. In particular, we replace words with their Brown cluster ids, using 1000 clusters acquired on a large scale Twitter corpus (Owoputi et al., 2013). This approach was shown to outperform BOW feature representation (see Section 4.5).

Evaluation metrics We use mean squared error (MSE) averaged across rumours to measure the difference between the true counts and predicted counts in the test intervals. The higher this metric, the further the model is from the actual counts. We also provide standard deviations, showing how much the MSE varies across rumours. Since the probabilistic models return distributions over the possible outputs, we also evaluate them via the log-likelihood (LL) of the true counts under the returned distributions. This metric indicates how much probability mass a model puts on the correct prediction, with high values indicating a good prediction. We also provide standard deviations, showing how much the LL varies across rumours.

For the inter-arrival time prediction we use a metric based on root mean squared error for evaluating inter-arrival time prediction, which we call penalized root mean squared error (PRMSE). Let the arrival times predicted by a model be $(\hat{t}_1, \ldots, \hat{t}_M)$ and let the actual arrival times be $(t_1, \ldots, t_N)$. The number of arrival times predicted by the model (M) and the actual number of arrival times (N) could differ, which in turn should be penalized by the metric. We propose an evaluation metric which takes this into account by adding the RMSE between the aligned first min(M, N) times and the squared differences between the unaligned times and the end of the observation window T. The metric takes the form:

$$PRMSE(\boldsymbol{t}, \hat{\boldsymbol{t}}) = \sqrt{\frac{1}{\min(M,N)} \sum_{i=1}^{\min(M,N)} (\hat{t}_i - t_i)^2 + \sum_{i=N+1}^{M} (T - \hat{t}_i)^2 + \sum_{i=M+1}^{N} (T - t_i)^2}. \tag{5.2}$$

Note that if we normalized the penalization terms by the number of the components the metric would not penalize for the excessive or insufficient number of points predicted by the model. Therefore, we leave it not normalized. We also consider the aligned root mean squared error (ARMSE), which is takes the form:

$$ARMSE(\boldsymbol{t}, \hat{\boldsymbol{t}}) = \sqrt{\frac{1}{\min(M,N)} \sum_{i=1}^{\min(M,N)} (\hat{t}_i - t_i)^2}. \tag{5.3}$$

Baselines We use the following baselines for frequency prediction:

HPP The first is the Homogenous Poisson Process (HPP) trained on the training set of the rumour. We select its intensity λ using the maximum likelihood estimate, i.e., the mean frequency of posts in the training intervals.

GP The second baseline is Gaussian process (GP) regression introduced for modeling popularity of hashtags in Twitter by Preotiuc-Pietro and Cohn (2013). The approach is based on discretising time into bins, and conducting regression over counts. The model uses Gaussian likelihood and thus ignores the count nature of the data, as it yields support for non-integer values. The Gaussian distribution is known to approximate Poisson likelihood (which is appropriate for count data) for large counts. However, the number of posts occurring in queried intervals in our case is not high, thus a Gaussian approximation for the point occurrence probability is not appropriate.

Various kernels were considered in the experiments, most notably periodic kernels. Periodic kernels were appropriate for their task, as various hashtags exhibit periodic behaviour (e.g. #TGIF re-occurs on Fridays). In our case it is not apparent that rumours exhibit periodic characteristics, as can be seen in Figure 2.1. We restrict our focus to the RBF kernel and leave inspection of other types of kernels such as periodic ones for both GP and LGCP models for future work.

INTERPOLATE Another baseline we use is tailored for the interpolation setting (*Interpolate*), and uses simple interpolation by averaging over the frequencies of the closest left and right intervals, or the frequency of the closest interval for test intervals on a boundary.

0 We also consider a baseline which is to always predict 0 posts in all intervals.

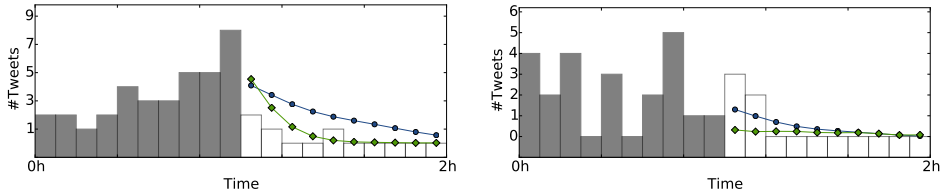
GPLIN For the inter-arrival time prediction we employ a GP baseline with a linear kernel (GPLIN), where the time until the next event is modelled over the input being the time of the recent post occurrence. We also considered an RBF kernel, but a model trained with this kernel tends to predict the inter-arrival times close to zero as time increases, yielding a huge number of points at prediction, often not stopping in feasible time. As we show in the experiments section, this problem is not fully solved for the linear kernel.

5.5 Experiments

In this section we report the experiments with the rumour popularity prediction problem on the Ferguson riots rumour dataset.

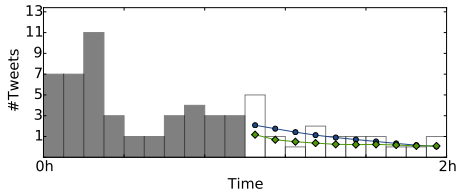
	Extrapolation		Interpolation	
	MSE	LL	MSE	LL
HPP	7.14±10.1⋆	-23.5±10.1⋆	7.66±7.55⋆	-25.8±11.0⋆
GP	4.58±11.0⋆	-	6.13±6.57⋆	-
Interpolate	4.90±13.1⋆	-	5.29±6.06⋆	-
0	2.76±7.81⋆	-	7.65±11.0⋆	-
LGCP	3.44±9.99⋆	-15.8±11.6†⋆	6.01±6.29⋆	-21.0±8.77†⋆
LGCP Pooled	2.50±8.62†⋆	-15.1±11.8†⋆	5.05±10.43†⋆	-18.9±11.9†⋆
LGCP ICM	2.46±7.82†⋆	-14.8±11.2†⋆	8.59±19.9⋆	-20.7±9.87†⋆
LGCP TXT	2.32±7.06†	-14.7±9.12†	3.66±5.67†	-16.9±5.91†
LGCP ICM+TXT	2.31±7.80†	-14.6±10.8†	3.92±5.20†	-16.8±5.34†

Table 5.1: MSE between the true counts and the predicted counts (lower is better) and predictive log likelihood of the true counts from the point process models (higher is better) for test intervals over the 114 Ferguson rumours for extrapolation (left) and interpolation (right) settings, showing mean ± std. dev. Baselines are shown above the line, with LGCP models below. Models: HPP, GP, 0 and LGCP only use target rumour data, with other models having access to reference rumours. We omit the LL evaluation for non-probabilistic models and for GP, which returns a predictive distribution over a continuous support, thus not directly comparable against other models. Key: † denotes significantly better than the best baseline (0 in the extrapolation, and Interpolate in the interpolation setting); ⋆ denotes significantly worse than the best model (LGCP ICM+TXT in the extrapolation, and LGCP TXT in the interpolation setting), according to the Wilcoxon signed rank test $p < 0.05$.

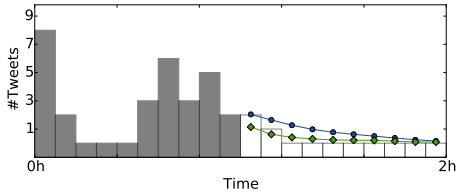


(a) rumour #7

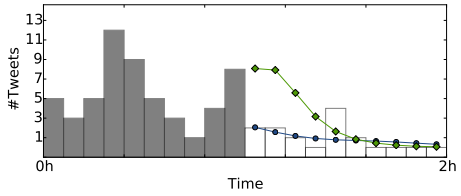
(b) rumour #12



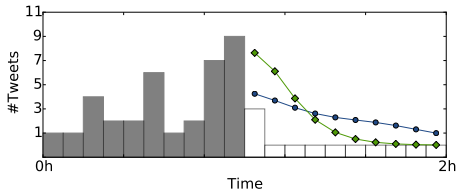
(c) rumour #18



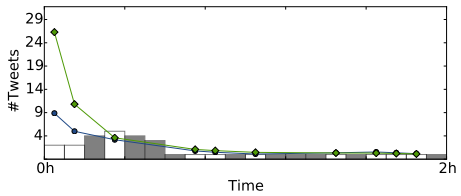
(d) rumour #19



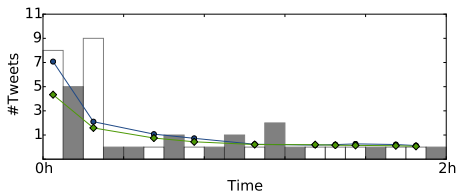
(e) rumour #23



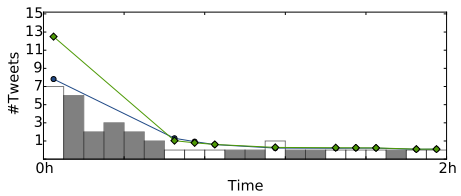
(f) rumour #29



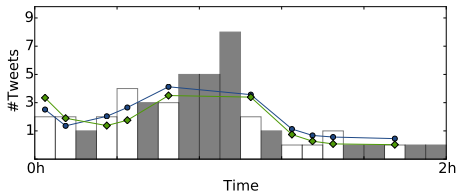
(g) rumour #2



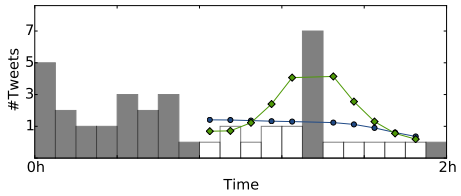
(h) rumour #5



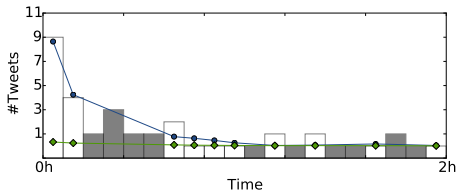
(i) rumour #7



(j) rumour #12



(k) rumour #27



(l) rumour #108

Figure 5.2: Intensity functions for different methods across example Ferguson rumours in the extrapolation (top six figures) and interpolation (bottom six figures) setting. Dark bars denote frequencies of tweets in the train intervals, and white bars denote counts of tweets in the test intervals. Predictions from LGCPTXT are denoted by blue circles, and from LGCPICM by green diamonds.

Extrapolation The left columns of Table 5.1 report the results for the extrapolation experiments, showing the mean and variance of results across the 114 rumours. GP performs relatively well compared to other baselines in terms of MSE, however is worse than LGCP. Notice GP models a predictive distribution with continuous support, and the results on LL are omitted, as likelihoods are not directly comparable to the probility estimates from the LGCP models.[2] Amongst the approaches which only have access to target rumour data, the plain LGCP model significantly outperforms the HPP and GP. This demonstrates that employing a model aware of the integer nature of the data yields a big improvement, even when a point estimate of the prediction is considered (MSE). We note that incorporating information about other rumours is useful, as LGCP Pooled yields a large improvement over LGCP. ICM, TXT and ICM+TXT multi-task learning approaches achieve the best scores, however only ICM+TXT significantly outperforms the LGCP Pooled method. Frequency profiles for ICM and TXT methods in the extrapolation setting are depicted in the top six subfigures of Figure 5.2 in the example of multiple rumours. TXT is better than ICM on multiple examples (rumours #33, #37, #57). However, for some rumours ICM does a better job (rumours #12, #54), and for others it is not directly clear which method is better (rumours #60). Notice that often TXT is better than ICM, which frequently underestimates the counts.

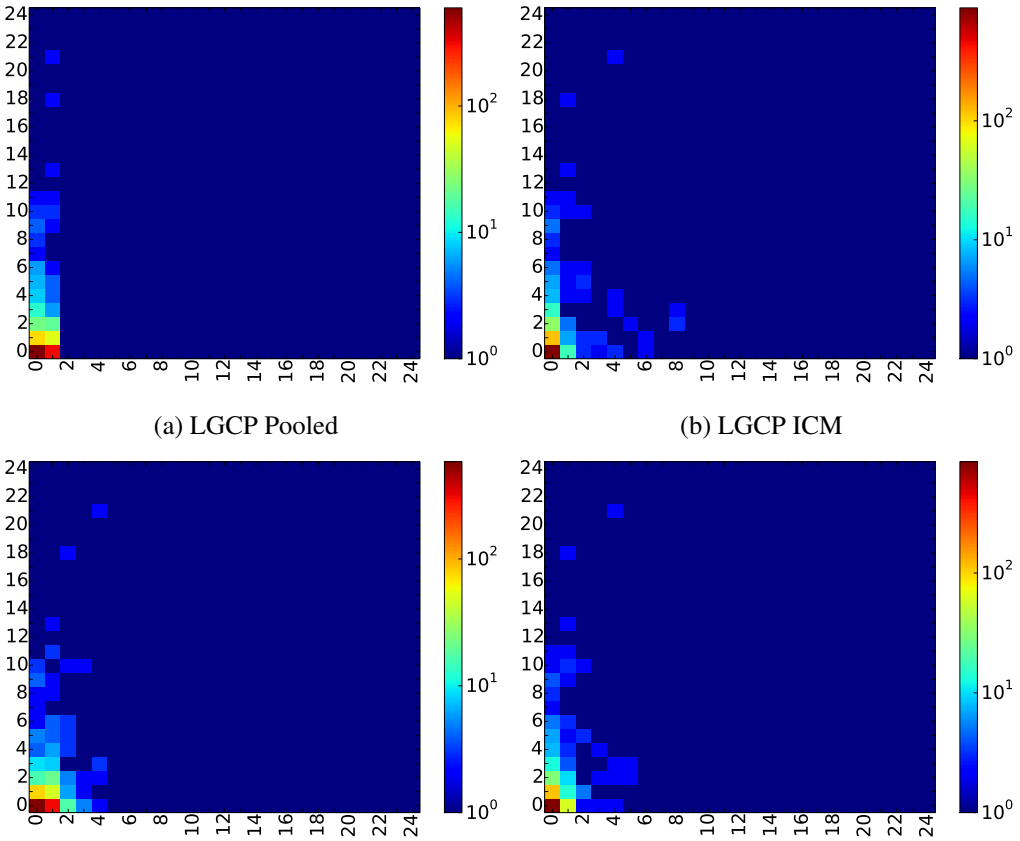
Interpolation As for the interpolation setting, we notice that again HPP and GP are among the worst methods. We can observe a large improvement in results when reference rumours are used for the LGCP model, however, quite surprisingly, learning correlations in the ICM model does not bring improvements, but to the contrary, leads to poor predictions. As for the multi-task methods, we notice that text is particularly useful, with TXT achieving the best results. We illustrate the differences between the TXT and ICM methods in the interpolation setting in the bottom six subfigures of Figure 5.2. First, notice that the task is easier than that of extrapolation, as there are observed intervals interleaving with the intervals of time that are unobserved, and so there is less uncertainty about what values one might expect in the test intervals, as typically the values don't drastically differ between neighbouring intervals (e.g. see rumour #7). Nevertheless, there are intervals for which the number of tweets is very different than in the neighbouring intervals (e.g. for rumour #27 the observed interval of time in the second hour of the rumour lifespan). Overall, notice that the text based multi-task learning approach makes better predictions about the initial peaks (first time bucket) than ICM (e.g. for rumours #5, #108). At the same time, for some

[2] In both interpolation and extrapolation setting the values from the pdf of the predictive distribution are much lower than from other models: respectively -34.6 and -90.1.

rumours ICM wrongly predicts peaks of interest at the beginning (e.g. for rumours #2, #7). This shows that temporal dynamics alone is not enough to correctly discriminate between rumours which are discussed at the beginning and not, and text adds complementary information to this end.

Analysis of the Best Performing Methods Next, we analyze the results of the best-performing approaches in each of the two settings by looking at the per-frequency performance. In Figures 5.3 and 5.4 we plot matrices denoting how often a particular approach classifies a ground truth frequency of tweets (corresponding to a row of the matrix) as a target frequency of posts (corresponding to a column of the matrix). Since a method can predict a non-integer count of tweets, we round each prediction to the nearest integer value. The higher the frequency values on the diagonal of a matrix, the better the performance of the model. Conversely, the higher the intensity values off-diagonal, the worse the model performance. Notice most ground truth counts take low values (0, 1 and 2), which demonstrates this is a largely imbalanced problem in terms of the prediction space. Thus, the bottom left side of each matrix largely contributes to the overall score. In Figure 5.3 we show the cross-classification matrices for LGCP Pooled, LGCP ICM, LGCP TXT, LGCP ICM+TXT methods in the extrapolation setting. Intensities for all methods are concentrated at the left side of the matrix, meaning that methods predict lower counts of tweets than they should. This is particularly noticable for LGCP Pooled, for which there is a very limited number of predicted counts a model makes: it only predicts either 0 or 1 posts for every interval. This is due to the method treating all tweets as coming from a single rumour (i.e. not weighing similarities between rumours in any way), thus learning the smoothed average counts in different intervals, which are usually low. However, since low frequency is predominant in the ground truth counts, this is already a very competitive approach. Notice that compared to TXT, ICM has a tendency to make predictions which are too large (columns corresponding to 4, 5, 6 and 8). TXT makes better calibrated decisions, not overestimating the counts as often. Looking at the differences between TXT and ICM+TXT, it is not clear which is better, which concurs with their similar results reported in Table 5.1.

In Figure 5.4 we show the cross-classification matrices for Interpolate, LGCP Pooled, LGCP TXT, LGCP ICM+TXT in the interpolation setting. Notice that overall the range of predicted values is wider for the interpolation setting than for the extrapolation setting, which is due to the low counts of tweets in the second hour in the data. Notice that again LGCP Pooled exhibits the most limited range of predicted values due to treating all tweets as if coming from the same rumour. Comparing Interpolate and LGCP Pooled methods, Interpolate shows a better overall spread over the matrix, whereas LGCP Pooled does not predict



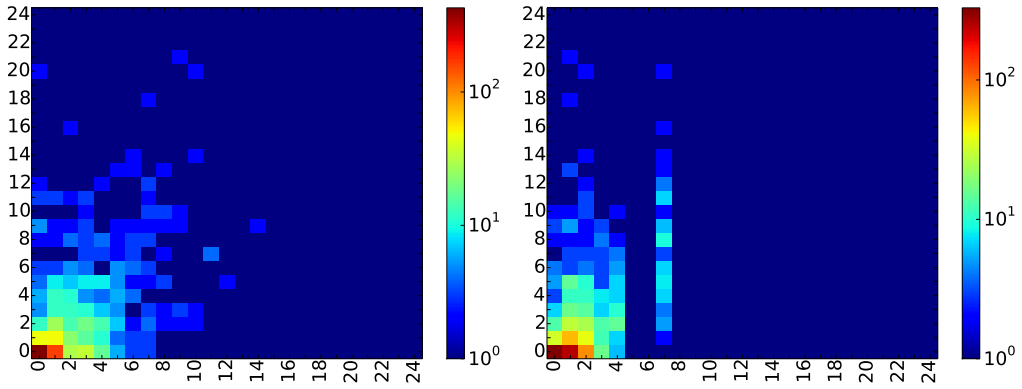
(a) LGCP Pooled

(b) LGCP ICM

(c) LGCP TXT

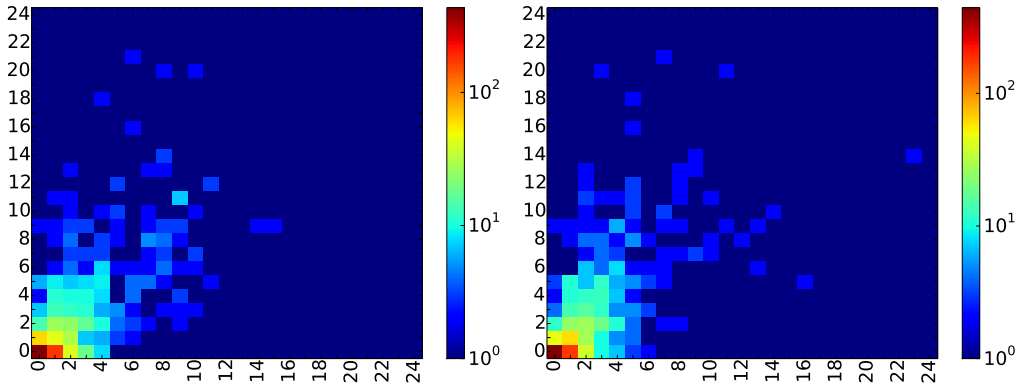
(d) LGCP ICM+TXT

Figure 5.3: Confusion matrices for four selected methods in the extrapolation setting. Each row corresponds to the true count of tweets in an interval, and each column corresponds to the predicted number of tweets. A cell i, j denotes how many times the ground truth count of tweets i is being classified as target count of tweets j.



(a) Interpolate

(b) LGCP Pooled



(c) LGCP TXT

(d) LGCP ICM+TXT

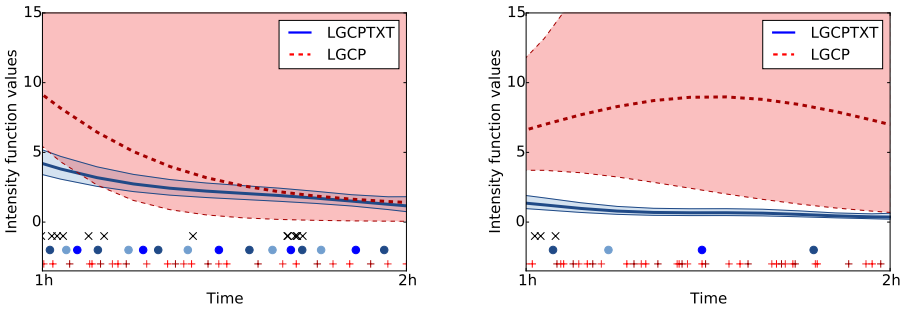
Figure 5.4: Confusion matrices for four selected methods in the interpolation setting. Each row corresponds to the true count of tweets in an interval, and each column corresponds to the predicted number of tweets. A cell i, j denotes how many times the ground truth count of tweets i is being classified as target count of tweets j.

method	ARMSE	PRMSE
GPLIN	20.60±22.01⋆	1279.78±903.90⋆
HPP	21.85±22.82⋆	431.4±96.5⋆
LGCP	13.31±14.28	261.26±92.97⋆
LGCPTXT	15.52±18.79	154.05±115.70

Table 5.2: ARMSE and PRMSE between the true event times and the predicted event times expressed in minutes (lower is better) over the 114 Ferguson rumours, showing mean ± std. dev. Key ⋆ denotes significantly worse than LGCPTXT method according to the Wilcoxon signed rank test ($p < 0.05$). In the case of ARMSE, LGCP is not significantly better than LGCP TXT according to Wilcoxon test.

higher frequencies than 7. Nevertheless, the two methods exhibit similar performance, as shown in Table 5.1. This might be due to LGCP Pooled compensating by making better predictions for low counts (i.e., see a higher intensity at cell $(1, 1)$ for LGCP Pooled (corresponding to count 61) than for Interpolate (corresponding to count 48)). LGCP ICM+TXT shows high over-predictions, whereas TXT suffers from this problem to a lesser extent.

Interarrival time prediction Table 5.2 reports the results of predicting arrival times of tweets in the second hour of the rumour lifecycle. We report both ARMSE (which only considers aligned sub-sequences from the ground truth and model output), and PRMSE (which does penalize for insufficient or excessive number of tweets predicted). In terms of ARMSE, LGCP is the best method, performing better than LGCPTXT (though not statistically significantly) and outperforming other approaches. Figure 5.5b depicts an example rumour, where LGCP greatly overestimates the number of points in the interval of interest. Here, the three points from the ground truth (denoted by black crosses) and the initial three points predicted by the LGCP model (denoted by red pluses), happen to lie very close, yielding a low ARMSE error. However, LGCP predicts a large number of arrivals in this interval, whereas LGCPTXT predicts only four points (denoted by blue dots). ARMSE fails to capture this difference between the models. A metric capturing this unwanted phenomenon is PRMSE, which does penalize LGCP in this example. Overall, according to PRMSE, LGCPTXT is the most successful method which significantly outperforms all other according to the Wilcoxon signed rank test. Figure 5.5a depicts the behavior of LGCP and LGCPTXT on rumour 39 with a larger number of points from the ground truth. Again, LGCPTXT predicts fewer arrivals than LGCP. Lastly, notice that GPLIN performs very poorly according to PRMSE, which is due to this method predicting decreasing inter-arrival times, leading to



(a) rumour #39 (b) rumour #60

Figure 5.5: Intensity functions and corresponding predicted arrival times for different methods across example Ferguson rumours. Arrival times predicted by LGCP are denoted by red pluses, LGCPTXT by blue dots, and ground truth by black crosses. Light regions denote uncertainty of predictions from the underlying Gaussian processes, which however are not used for predictions. Nevertheless, they show that the underlying GP has narrower uncertainty bounds for LGCPTXT.

large numbers of tweets being predicted.

5.6 Conclusions

This chapter introduced the problem of modelling frequency profiles of rumours in social media. We demonstrated that joint modelling of collective data over multiple rumours using multi-task learning resulted in more accurate models that are able to recognise and predict commonly occurring temporal patterns. We showed how text data from social media posts added important information about similarities between different rumours, which led to better predictions. We also introduced the problem of predicting the exact times of occurrences of future tweets. We demonstrated how the best LGCP-based model for frequency prediction is also successful for this task, significantly outperforming the baselines.

The most successful approaches from this chapter used multi-task learning approaches with kernels comparing pairs of rumours. In the next chapter we further explore this approach, seeking better ways of capturing rumour similarities.

Chapter 6

Convolution Kernels for Modeling Temporal Dynamics of Rumours

In the previous chapter we started addressing the second research aim of this thesis: **predicting rumour popularity**. We formulated a model of rumour popularity using a kernelized point process model, a log-Gaussian Cox process (LGCP), and considered multiple formulations of LGCP with different kernel functions. The most successful approaches were incorporating data from other rumours while modeling similarities between them. In this chapter, we continue addressing the same research aim, and work towards answering the research question:

Does information about how text usage in tweets changes over time convey useful information for the rumour popularity prediction task?

In particular, we show how rumour similarities can be measured in a way that captures how text evolves over time in different rumours, and show how such an approach outperforms alternative methods.

Parts of this chapter have been published as

> Lukasik, M. and Cohn, T. (2016). Convolution kernels for discriminative learning from streaming text. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence*.

6.1 Introduction

Modeling of time series is a foundational problem in the machine learning literature, which typically attempts to model the dynamics of a signal into the future based on a history of observations. The previous chapter considered the problem of modeling popularity of

rumours represented by sequences of posts over continuous time. Point processes were used to model the spread of multiple rumours over time via kernels defined over both time and rumours, with kernels being decomposed into a product of a kernel over time and a kernel over rumour space. Both the temporal dynamics of the rumour spread and the text content of the posts contain important cues for determining the future spread of a rumour. However, we considered both of these sources of information in separation, i.e., we did not look at the differences between text posted about a rumour at different points of time.

In this chapter we address the problem of comparing time series of text over continuous time by adapting the general idea of convolution kernels (Haussler, 1999). Convolution kernels avoid the effort of feature engineering and allow for using a rich data representation. We demonstrate the proposed convolution kernels outperform previous approaches on the rumour popularity prediction tasks, and also demonstrate its applicability to other problems involving discriminative learning from time series of text over continuous time: classification, regression and Poisson regression. Therefore, the convolution kernels are introduced in general settings, not restricted to the rumour popularity prediction problem.

This chapter makes the following contributions:

1. Introduces a method based on convolution kernels for discriminative modeling of time series composed of text over continuous time;

2. Demonstrates the proposed approach on a range of discriminative learning problems: classification, regression, Poisson regression and point process modeling;

3. Demonstrates the efficacy of the method on three synthetic and two real datasets. In particular, we show how it outperforms methods considered in the previous chapter on the task of predicting rumour popularity.

6.2 Related Work on Modeling Sequences of Text over Time

Time series modeling is an important problem with diverse applications, from epidemiological models of population health (Bhaskaran et al., 2013) to biological models of gene expression over time (Bar-Joseph et al., 2012) and even words of text as they appear in sentences using Markov models (Brown et al., 1992). The modeling challenge is to characterise how the time series data changes over time, typically in support of predictions of future changes in the time series or to fill in missing data from the past. The aim of this chapter is to develop a method (in the form of a kernel function) for comparing sequences of posts in continuous time in such a way that predictions of rumour popularity become bet-

ter. Below, we conduct a review of approaches for modeling problems related to learning from sequences of text over time.

Discriminative Learning from Time Series

Time series have been considered for a long time. Predominantly it is considered in settings where previous instances of a time series are used to predict the outputs for future points (Shumway and Stoffer, 2006). However, there is also a branch of work devoted to discriminative learning from time series, where the aim is to perform classification over whole time series data. One considered type of time series is a sequence of symbols, where model based approaches such as Hidden Markov Models (Rabiner, 1989) have been adapted for discriminative learning (Xing et al., 2010). However, in this chapter we consider multivariate time series, which are sequences of numerical vectors (Xing et al., 2010).

Recently for time series feature selection based methods were considered. A popular approach to feature selection is extraction of informative subsequence patterns (shapelets), which has been proposed in conjunction with decision trees (Ye and Keogh, 2011), SVM (Ando and Suzuki, 2014) and has been applied to a range of problems e.g. medicine (Xing et al., 2011). Closely related is extraction of meta features, the user-defined recurring substructures of a time series (Kadous and Sammut, 2005). However, in this chapter we consider a kernelized approach where no explicit feature extraction is conducted.

Another approach to time series classification is based on using distance functions for comparing pairs of time series in conjunction with a distance-based method such as KNN, SVM (Xing et al., 2010). In contrast, we use a Bayesian non-parametric approach based on Gaussian processes. In the case of simple time series, where the only information about each point from a time series is time, popular approaches were euclidean distance for time series of equal lengths and Dynamic Time Warping (DTW) for sequences of unequal lengths (Keogh and Pazzani, 2000). These are not appropriate for our problem, since we consider multi-variate time series where a vector of meta data (text) is assigned to each timestamp (see Section 6.3). Another example is based on the kernel fusion scheme, in which multiple feature extraction methods are considered, and different kernels are applied for comparing the different feature representations. The kernels are then combined together by a weighted sum or some more complex combining function (Wu et al., 2005). We do consider a combination of kernels over text and over time treated as different representations of time series (see Section 6.4), which can be viewed as an instantiation of the kernel fusion scheme. In this case different views of time series are combined in a weighted average, where weights are kernel variances and are optimized by maximizing the evidence in the Gaussian process

framework (see Section 3.1). However, in our main contribution we do not combine kernels over different representations of time series, but conduct an R-convolution (see Section 6.4) between kernels over individual points from them, which is a more powerful method as we show in experimental sections.

Note that these algorithms were not previously applied to multivariate time series of text which is our main contribution. Moreover, they considered only classification settings, whereas we consider more scenarios in our experiments in Section 6.5, introducing regression and Poisson regression problems over the time series inputs.

6.3 Notation and Problem Formulation

In Section 5.2 we introduced the problem of rumour popularity prediction, which is the main motivation for the method developed in this chapter. However, we also demonstrate the usefulness of the proposed method on a range of other problems. Here we introduce the range of problems that our approach can be used for.

Let us consider a set of events $R = \{R_m\}_{m=1}^{|R|}$, each of which consists of a set of posts $R_m = \{\mathbf{p}_m^n\}_{n=1}^{|R_m|}$. Posts are tuples $\mathbf{p}_m^n = (\mathbf{w}_m^n, t_m^n)$, where $\mathbf{w}_m^n$ is a vector text representation and t_m^n is the timestamp of post $\mathbf{p}_m^n$.[1] This way, events are time series over vectors. One can imagine different situations for which the kernels we introduce can be easily adapted, e.g. using different text representations by applying string kernels over the text (Lodhi et al., 2002).

In this work we consider a range of discriminative problems over paired data $D = \{R_m, y_m\}_{m=1}^{|R|}$, where time series are inputs and we model an output variable, most often a numerical value, but may be more complex as in the case of the point processes below, and which can be viewed in more general context beyond the rumour use-case. In particular, we consider the following four general classes of problems:

1. Binary classification, where the task is to model a binary valued function over events, $y_m \in \{0, 1\}$, e.g. labelling events as rumours or non-rumours;

2. Regression, where the task is to model a real valued function over events, $y_m \in \mathbb{R}$, e.g. the price of stocks described by a time series of posts about them;

3. Poisson regression, where the task is to model a non-negative integer valued function over events, $y_m \in \mathbb{N}$, e.g. based on events up to time t, determining the number of posts in a future time interval;

[1] Recall our notation is summarized in the Nomenclature on page xii.

4. Frequency prediction, where the task is to model a non-negative integer valued function over intervals of time, i.e., based on events up to time t, determine the number of posts in arbitrary future time intervals. In this case, y_m could be a complete set of points over the observation window. Intuitively, frequency prediction can be viewed as a finer grained version of Poisson regression. The example frequency prediction problem is rumour popularity prediction as defined in Section 5.2, which is the main motivation of this chapter.

Our kernels are capable of modeling a broader range of problems than those listed above. For example they could be used in a kernel-based clustering method for event detection in a social media stream. In this case, one might group together tweets based on the distance calculated using a convolution kernel over text and time.

6.4 Convolution time series kernels

Convolution kernels are a framework in which kernels between structured objects are specified as a combination of kernel values between substructures (Haussler, 1999). Convolution kernels have been successfully applied to discrete objects, for example graphs (Vishwanathan et al., 2010). A compelling natural language processing example is a tree kernel which operates on syntax trees through recursive decomposition into kernels between subtrees (Collins and Duffy, 2001). Of particular relevance are string kernels, which recursively decompose strings into substrings (Lodhi et al., 2002). These kernels treat inputs as time series over discrete time, whereas the time series we consider here are over continuous time (and possibly other dimensions based on the metadata attached to events). This makes a direct application of such kernels impossible. However, they can serve as an inspiration for constructing convolution kernels between time series, where decomposition is conducted over parts of a time series. We start from a simple kernelised approach to comparing events, before showing how it can be generalized to convolution kernels.

6.4.1 Formulations

A kernel over rumours can be formulated in many different ways. The most obvious is concatenating text from posts together and comparing the resulting bags-of-words, e.g. using a kernel on the vectors. Such a kernel can be expressed via the formula

$$\mathbf{k}_{\Sigma\text{text}}(R_m, R_l) = k_{\text{text}}\left(\sum_{\mathbf{p}_m^i \in R_m} \mathbf{w}_m^i, \sum_{\mathbf{p}_l^j \in R_l} \mathbf{w}_l^j\right), \tag{6.1}$$

where $\mathbf{w}_m^i$ denotes a vector representation of text content of post i (in our case this is bag of Brown cluster ids, see section 6.6 for details about the text representation) from rumour R_m. Note that we denote kernels over rumours with bold font $\mathbf{k}$ and kernels over their component posts using small k.

A very popular kernel choice for text is the linear kernel, which takes the form

$$k_{\text{text}}(\mathbf{w}_1, \mathbf{w}_2) = \gamma \mathbf{w}_1^\top \mathbf{w}_2, \tag{6.2}$$

where γ is a learned scaling hyper-parameter (see Section 3.1.5 for more details about the linear kernel). Combined with equation (6.1), the linear kernel simplifies to

$$\mathbf{k}_{\Sigma\text{text}}(R_m, R_l) = \sum_{\mathbf{p}_m^i \in R_m} \sum_{\mathbf{p}_l^j \in R_l} k_{\text{text}} \left(\mathbf{w}_m^i, \mathbf{w}_l^j \right). \tag{6.3}$$

The kernel in equation (6.3) is expressed as a double summation of kernels between each pair of posts. It can be also viewed as a convolution kernel between the rumours, where the decomposition is made into substructures being posts, and combined via simple summation. In this paper we use a linear kernel for the text, although note that we could easily replace it with another kernel for comparing the text, e.g. a string kernel or an RBF kernel, in which case this ceases to be equivalent to a kernel over the concatenated text.

The first non-trivial convolution kernel we consider is the linear kernel normalized by the rumour sizes,

$$\mathbf{k}_{\text{text}}(R_m, R_l) = \frac{1}{|R_m||R_l|} \sum_{\mathbf{p}_m^i \in R_m} \sum_{\mathbf{p}_l^j \in R_l} k_{\text{text}} \left(\mathbf{w}_m^i, \mathbf{w}_l^j \right). \tag{6.4}$$

We normalize the kernel values so that time series of varying length can be reliably compared without any bias towards longer structures.

A shortcoming of $\mathbf{k}_{\text{text}}$ is that it operates on text only and ignores time. Therefore, we introduce a convolution kernel using time,

$$\mathbf{k}_{\text{time}}(R_m, R_l) = \frac{1}{|R_m||R_l|} \sum_{t_m^i \in R_m} \sum_{t_l^j \in R_l} k_{\text{time}} \left(t_m^i, t_l^j \right). \tag{6.5}$$

This kernel should be useful for comparison of raw time series without additional metadata.

For comparing time values we use an RBF kernel,

$$k_{\text{time}}(t_i, t_j) = \sigma \exp\left(-\frac{(t_i - t_j)^2}{l}\right), \tag{6.6}$$

where $\sigma > 0$ is a hyper-parameter controlling the output scale, while $l > 0$ is the length scale, determining the rate at which the kernel diminishes with distance (see Section 3.1.5 for mote details about the RBF kernel).

We also consider a summation of time and text kernels,

$$\mathbf{k}_{\text{text+time}}(R_m, R_l) = \mathbf{k}_{\text{text}}(R_m, R_l) + \mathbf{k}_{\text{time}}(R_m, R_l). \tag{6.7}$$

Note that kernel text+time can be viewed as an instantiation of the kernel fusion scheme (Wu et al., 2005) where different views of time series are combined in a weighted average, where weights are kernel variances.

The final convolution kernel we consider compares pairs of posts via a product of kernels over text and time,

$$\mathbf{k}_{\text{text}\otimes\text{time}}(R_m, R_l) = \frac{1}{|R_m||R_l|} \sum_{\mathbf{p}_m^i \in R_m} \sum_{\mathbf{p}_l^j \in R_l} k_{\text{text}}\left(\mathbf{w}_m^i, \mathbf{w}_l^j\right) k_{\text{time}}\left(t_m^i, t_l^j\right). \tag{6.8}$$

This way, the kernel captures not only the textual similarities between posts coming from the two time series, but also how close they are in time. It effectively weights influence coming from each pair of posts by how far apart in time they were created. In this way it can focus on differences in the usage of text across time.

6.4.2 Proof of correctness

In this section we prove that the proposed convolution kernels correspond to R-convolutions, which themselves are valid kernels (Haussler, 1999).

Theorem 1. *From Haussler (1999). Let $R_m \in R$ be a composite structure. Let* $\mathbf{Rel}$ *be a relation such that* $\mathbf{Rel}\left(\mathbf{p_m}^1, \ldots, \mathbf{p_m}^Z, R_m\right)$ *is true iff* $\mathbf{p_m}^1, \ldots, \mathbf{p_m}^Z$ *are "parts" of* R_m*. For brevity,* $\mathbf{p_m}^1, \ldots, \mathbf{p_m}^Z$ *are jointly denoted as* $\overrightarrow{\mathbf{p_m}}$*. Let* $\mathbf{Rel}^{-1}(R_m) = \{\overrightarrow{\mathbf{p_m}} : \mathbf{Rel}(\overrightarrow{\mathbf{p_m}}, R_m)\}$*. Let the similarity function* $\boldsymbol{k}(R_m, R_l)$ *be defined as:*

$$\boldsymbol{k}(R_m, R_l) = \sum_{\overrightarrow{\mathbf{p_m}} \in R^{-1}(R_m)} \sum_{\overrightarrow{\mathbf{p_l}} \in R^{-1}(R_l)} \prod_{z=1}^{Z} k_d(\mathbf{p_m}^z, \mathbf{p_l}^z) \tag{6.9}$$

where k_d is a valid kernel for comparison of corresponding parts from $\overrightarrow{\mathbf{p_m}}$ and $\overrightarrow{\mathbf{p_l}}$. The zero-extension of $\mathbf{k}$ to $R \times R^2$ (also called the R-convolution) satisfies the Mercer's condition.[3]

Proof. See (Haussler, 1999). □

Theorem 1 defines R-convolutions over sets of structured objects R, where the inverse of relation $\mathbf{Rel}$: $\mathbf{Rel}^{-1}(R_m)$ generates decompositions of R_m. An example is when R is a set of strings and $\mathbf{Rel}^{-1}(R_m)$ is a set of prefixes of a string R_m. Theorem 1 states that, given R and $\mathbf{Rel}$, a similarity function formulated as in equation (6.9) constitutes a valid kernel.

Theorem 2. *The introduced time series kernels are R-convolutions.*

Proof. We demonstrate how the general form of proposed time series kernels:

$$\mathbf{k}(R_m, R_l) = \sum_{\mathbf{p_m}^i \in R_m} \sum_{\mathbf{p_l}^j \in R_l} k\left(\mathbf{p_m}^i, \mathbf{p_l}^j\right) \tag{6.10}$$

is an R-convolution by defining an equivalent formulation falling into the definition of R-convolution kernels given in Theorem 1. We consider a formulation without multiplication by $\frac{1}{|R_m||R_l|}$ as it is easy to see that this normalization preserves the Mercer's condition.[4]

As defined in the main text, each rumour R_m is a sequence of posts $\{\mathbf{p_m}^i\}_{i=1}^{|R_m|}$. We define a relation $\mathbf{Rel}$: $\mathbf{Rel}(\mathbf{p_m^i}, R_m)$ to be true iff $\mathbf{p_m^i} \in R_m$. With such a defined $\mathbf{Rel}$, kernel

$$\mathbf{k}(R_m, R_l) = \sum_{\mathbf{p_m}^i \in \mathbf{Rel}^{-1}(R_m)} \sum_{\mathbf{p_l}^j \in \mathbf{Rel}^{-1}(R_l)} k(\mathbf{p_m}^i, \mathbf{p_l}^j) \tag{6.11}$$

is an equivalent form of the time series kernel shown in equation (6.10). It corresponds to the form given in assumptions of Theorem 1 (where $Z = 1$), therefore time series kernels are R-convolutions. □

Corollary 1. *Given Theorems 1 and 2, the introduced time series kernels satisfy the Mercer's condition and so are valid kernels.*

[2]The zero-extension of $\mathbf{k}$ to $R \times R$ is considered because the similarity function $\mathbf{k}$ is defined by definition only on $S \times S$, where $S = \{R : \mathbf{Rel}^{-1}(R) \neq \emptyset\}$.

[3]Kernels satisfying Mercer's condition are positive semi-definite, and thus can be used for specifying a valid Gaussian process distribution.

[4]Multiplication by $\frac{1}{|R_m||R_l|}$ corresponds to a tensor product with kernel $\mathbf{k}(R_m, R_l) = \langle \frac{1}{|R_m|}, \frac{1}{|R_l|} \rangle$.

6.5 Experiments on Synthetic Data

In this section we describe experiments with the introduced kernels on synthetic data. We start with two toy classification examples, constructed to illustrate when our proposed convolution kernels can be useful. Then, we move to more extensive synthetic evaluation for classification, regression and Poisson regression outputs.

6.5.1 Toy example for time

First we consider binary classification based on purely temporal time series with no further meta data. Time series from class 0 consist of a number of timestamps drawn from a triangular distribution[5] $T(0, 0.5, 0.5)$ and a number of subsequent timestamps drawn from a triangular distribution $T(0.5, 0.5, 1)$. Class 1 time series are defined similarly, but with the shapes of triangular distributions swapped: timestamps come from $T(0, 0, 0.5)$ and $T(0.5, 1, 1)$. We depict the histograms of timestamps from example time series in Figure 6.1. In this problem, the expected value of a timestamp coming from either class 0 or class 1 is the same, however the overall distributions are different.

We run an experiment to compare the time convolution kernel (from Equation (6.5)) with the baseline kernel $\mathbf{k}_{\Sigma\text{time}}$,

$$\mathbf{k}_{\Sigma\text{time}}(R_m, R_l) = k_{\text{time}}\left(\sum_{\mathbf{p}_m^i \in R_m} t_m^i, \sum_{\mathbf{p}_l^j \in R_l} t_l^j\right). \tag{6.12}$$

For this experiment, we generate 500 timestamps for each class from each of the two triangular distributions. Then, each of the 500 timestamps is uniformly assigned to one of the 50 rumours of the dataset. We evaluate the kernels based on predictive accuracy with 5 fold cross validation on the resulting 100 rumours. We can see from Table 6.1 the results from the experiment. The time kernel is very effective, modelling the problem almost perfectly, whereas the prediction accuracy of the baseline kernel is no better than chance (achieving an accuracy of 0.5). Thus, unsurprisingly, simply looking at the mean over the timestamps is insufficient to distinguish between the two classes.

[5] A triangular distribution is a probability distribution with a density function of a triangular shape. It is parametrized by a minimum value, a mode value and a maximum value of the distribution. Note that if mode equals to either the minimum or the maximum value, the density function is of a right-angled triangle shape.

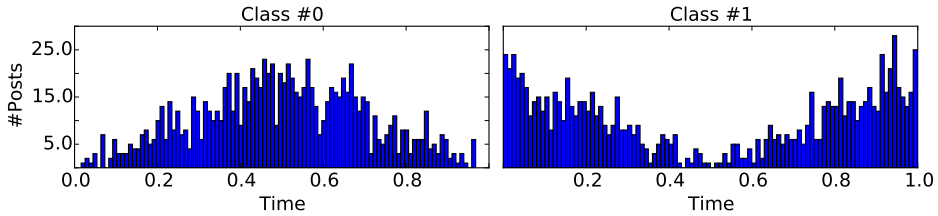


Figure 6.1: Samples from the toy classification example for time. Data shown in the form of histograms of posts over time.

method	mean	std
majority	0.44	0.04
Σtime	0.48	0.05
time	0.99	0.02

Table 6.1: Accuracy across 5CV for the toy experiment on temporal time series with no text meta data.

6.5.2 Toy example for time and text

Next, we move to a synthetic example incorporating additional meta data (text) for which the convolution kernel text∘time should prove useful. As before, we again consider binary classification. The time values for each timestamp are drawn from a Gaussian distribution $t \sim \mathcal{N}(0.5, 0.1)$ (excluding the values drawn from outside the interval $[0, 1]$), regardless of class identity. Text is assigned to each timestamp, and together they constitute a post. Below we describe how we generate text conditioned on time. Our goal is to mimic a scenario, where in one class of rumours text usage changes over time in one specific way (e.g. from the crowd supporting a rumour to denying it), and in the second class of rumours the text usage changes over time differently (from the crowd denying a rumour to supporting it). Moreover, we generate the text in such a way, that marginally the text distributions across the two classes do not differ.

We now give the details of the process generating the example. Each post consists of 10 words over a two word vocabulary $\{w_1, w_2\}$ drawn from a Bernoulli distribution dependent on t. Namely, for rumours coming from class 0 the probability of word w_1 is $|t|$, whereas the probability of word w_2 is $1 - |t|$. The probabilities for class 1 are reversed. We depict example samples coming from such distributions in Figure 6.2. For each class we generate 1000 rumours, each consisting of 10 posts.

In Table 6.2 we report averaged results from 5 fold cross validation. As expected, the

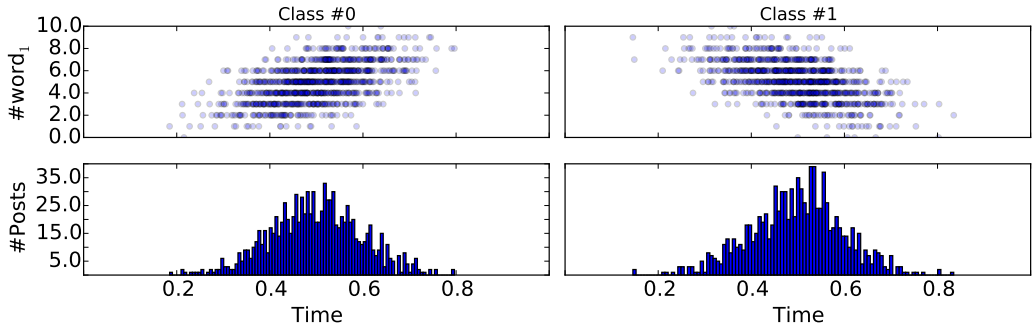


Figure 6.2: Samples from the toy classification example for time and text. The lower row shows histograms of posts over time, while the upper row shows the count #word$_1$ for each post. Note the count of word$_2$ equals $10-$#word$_1$.

method	mean	std
majority	0.44	0.07
text	0.50	0.09
time	0.41	0.06
text$\circ$time	0.98	0.02

Table 6.2: Accuracy across 5CV for the toy experiment on temporal time series with text meta data.

$$
\begin{aligned}
\alpha_0 &\sim U([1, 10]) \\
m &\sim T(1, 1.5, 2) \\
l &\sim \mathcal{N}(0.1, 0.05) \\
b^1, \ldots, b^{|R|} &\sim U\left(\max(1, 2m-2), \min(2, 2m-1)\right) \\
\forall_{i=1:|R|}\, c_1^i &\sim \mathcal{N}(b^i, \sigma_1^2) \\
\forall_{i=1:|R|}\, c_2^i &\sim \mathcal{N}\big(2m - b^i, \sigma_1^2\big) \\
\alpha_1^1, \ldots, \alpha_1^{|R|} &\sim \mathcal{N}(\alpha_0, \sigma_2^2) \\
\alpha_2^1, \ldots, \alpha_2^{|R|} &\sim \mathcal{N}(\alpha_0, \sigma_2^2) \\
\forall_{i=1:n} \forall_{j=1:z_i}\, t_{ij} &\sim \text{Beta}(\alpha_1^i, \alpha_2^i) && \text{post times for rumour } i \\
\forall_{i=1:n} \forall_{j=1:z_i}\, d_{ij} &\sim \text{Geometric}(l) && \text{number of words in post } i, j \\
\forall_{i=1:n} \forall_{j=1:z_i} \forall_{k=1:d_{ij}}\, w_{ijk} &\sim \text{Geometric}(c_1^i + (c_2^i - c_1^i) t_{ij}) && \text{words in post } i, j
\end{aligned}
$$

Table 6.3: Generating process of a single synthetic experiment. Variable z_i is fixed to 20 for regression and Poisson regression, whereas for classification it is randomly drawn as described in the main text.

methods that use marginal information about text and time perform no better than random, since there is no signal in text (or time) alone. Only our convolution kernel is capable of modelling this data, achieving near perfect accuracy.

6.5.3 Complex synthetic experiment

We now introduce a more complex synthetic experimental setting, in contrast to the previous two toy settings. Namely, we generate rumours of different temporal, textual, and joint temporal-textual characteristics. Our aim is to generate data where it is hard to perform well on the task using either time or text only, i.e., where incorporating information of how language usage changes over time is necessary for achieving good results.

In real text data, the distribution is such that there are many different words, where few words are frequent and most words are infrequent. In order to make our synthetic experiments account for this characteristic, words in posts come from a Geometric distribution (where words correspond to positive integers). Moreover, we model the parameter of the Geometric distribution as a function of time, changing the effective frequency of each word.

The generative story Table 6.3 summarises the generative process. First, we draw parameters controlling the "background rumour" and then perturb them for each rumour. These

parameters are perturbed in such a way that rumours are similar, and careful modeling is necessary for capturing differences between them.

First, three parameters influencing the "background rumour" are drawn: parameter α_0 controlling the timestamps, parameter m controlling the language, and parameter l controlling the lengths of posts. Two additional parameters are controlled in the experiment: σ_1^2 and σ_2^2, determining how the rumours differ from the "background rumour" in the distributions of text and time.

After drawing the parameters controlling the "background rumour", parameters specific to each rumour are drawn. For each rumour R_i, parameter b^i is drawn controlling the dynamics of language change around a rumour. The parameter of the Geometric distribution controlling the language usage at time 0 is c_1^i, and the parameter of the Geometric distribution controlling the language usage at time 1 is c_2^i (the parameter of the Geometric distribution controlling the language at any other timestamp is a linear function of time). Both of these parameters are drawn from Gaussian distributions with means which are functions of b^i.

Next, for each rumour two parameters governing the rumour specific distribution of timestamps are obtained by adding Gaussian noise with small variance to α_0. The addition of little noise poses a tough challenge in modelling the data, making it hard to learn the different temporal distributions across rumours.

In the end of the generative process, timestamps are drawn from a Beta distribution $t \sim \text{Beta}(\alpha_1^i, \alpha_2^i)$, the length of each post is drawn from a Geometric distribution controlled by the parameter l (shared across the rumours), and each word in each post is drawn from a Geometric distribution dependent on both the drawn timestamp and the rumour specific parameters c_1^i and c_2^i.

In summary, the described procedure of generating rumours consists of three sources of variation across rumours. One is the marginal difference between timestamps of posts (different parameters for Beta distributions governing time), another is the marginal difference between text in posts (random perturbations of parameters c_1^i and c_2^i controlling the text distributions in posts from each rumour), and the third difference is in the dynamics of variation of text over time (the linear function over time dependent on c_1^i and c_2^i governing the Geometric distribution from which text content is generated).

Output variables

We consider three different problem settings: classification, regression and Poisson regression.

	classification (ACC)	Regression (MSE)	Poisson regression (LL)
mean	-	0.16±0.08	-39.15±8.18
majority	0.44±0.04	-	-
text	0.62±0.16	0.15±0.08	-39.46±9.42
time	0.44±0.04	0.15±0.07	-39.40±8.21
text+time	0.57±0.12	0.14±0.07	-37.01±8.66
text ∘ time	**0.80±0.16**	**0.05±0.03**	**-31.00±5.63**

Table 6.4: Results from the synthetic experiments for a range of methods (mean ± std dev), showing classification accuracy, mean squared error for regression and Poisson log likelihood. We ran 55 experiments of each setting, taking 80% of rumours for training and 20% for testing.

Classification In the classification setting, we draw parameters governing text distribution of posts twice, obtaining c_1^0, c_2^0 (corresponding to label 0, e.g. *the true rumour*) and c_1^1, c_2^1 (corresponding to label 1, e.g. *the false rumour*). We generate 1000 posts for each label and split them uniformly into 50 rumours, ending up with 100 rumours for a single experiment. We evaluate performance using predictive accuracy.

Regression For regression we draw as many pairs of parameters c_1^i, c_2^i as there are rumours (100, each consisting of 20 posts). The response variable for rumour R_i is set to $y_i = c_1^i - c_2^i$ (recall c_1^i and c_2^i control how rapidly the language changes over time). We use mean squared error (MSE) between the predicted responses and the ground truth for evaluation.

Poisson regression As for the regression setting, in Poisson regression we draw as many pairs of parameters c_1^i, c_2^i as there are rumours (100, each consisting of 20 posts). For each rumour the response variable is set to $y_i = \max\left(1, 1 + \left\lfloor 10(c_1^i - c_2^i)\right\rfloor\right)$. We evaluate models by calculating the log likelihood (LL) of observations under Poisson distribution with mean set to the mean value predicted by the model.

Results

In Table 6.4 we report results from the three experiment settings. Note that in all settings, text∘time yields excellent results, outperforming all other methods. In contrast, the simpler

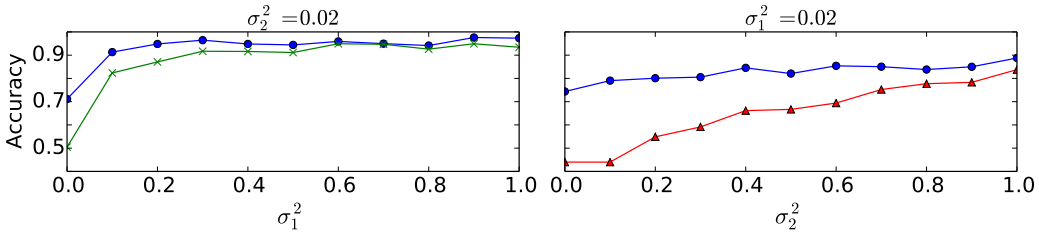


Figure 6.3: Comparison of accuracy between convolution kernels, showing kernels text ∘ time (blue circles), text (green crosses) and time (red triangles). The left plot shows the accuracy as the σ_1^2 varies, while the right plot varies σ_2^2 (the other parameter is fixed). This changes the amount of signal in text and time components alone, respectively.

kernels (text, time, text+time) did not significantly improve over the mean or majority baseline. Note that text+time can be viewed as an instantiation of the kernel fusion scheme (Wu et al., 2005) where different views of time series are combined in a weighted average. The results show that this method is outperformed by the more powerful text∘time kernel. In Figure 6.3 we analyze the influence of parameters σ_1^2 and σ_2^2 from the generative process to performance of the convolution kernels. Notice that as σ_1^2 or σ_2^2 increases (thus introducing bigger marginal differences across rumour distributions in time or text) — due to high variance in appropriate component of the generative process —, the kernels using text or time only become more and more effective relative to other methods.

6.6 Experiments on Rumour Data

We now evaluate our approach on two social media datasets. The two tasks we consider are related to rumour popularity modeling, following Chapter 5.

Data We conduct experiments on two social media datasets constructed from the PHEME rumour datasets described in section 2.5.2. Notice we do not need stance annotations for tweets in the rumour popularity prediction task, and therefore are not restricted to the subsets of PHEME datasets described in Section 2.5.2. The first dataset consists of 114 popular rumours collected in August 2014 during the Ferguson unrest and is composed of 4098 tweets. This dataset has been used for rumour dynamics modeling in the previous Chapter 5. We will show how the introduced convolution kernels outperform the methods that we introduced earlier. We also consider a second dataset consisting of tweets collected October 2014 during Ottawa shootings and in August 2014 during the Ferguson unrest (Zubiaga et al., 2015a). The corpus consists of 288 Ferguson rumours and 470 Ottawa rumours, and is

	MSE	LL
HPP	7.14±10.1⋆	-23.5±10.1⋆
GP regression	4.58±11.0⋆	-
Interpolate	4.90±13.1⋆	-
0	2.76±7.81⋆	-
LGCP	3.44±9.99⋆	-15.8±11.6†⋆
LGCP ICM	2.46±7.82†	-14.8±11.2†
LGCP TXT	2.32±7.06†⋆	-14.7±9.12†⋆
LGCP ICM+TXT	2.31±7.80†	-14.6±10.8†
LGCP text ∘ time	**2.21**±7.09†	**-14.2**±8.6†

Table 6.5: MSE between the true counts and the predicted counts (lower is better) and predictive log likelihood of the true counts from the point process models (higher is better) for test intervals over the 114 Ferguson rumours, showing mean ± std. dev. Key: † denotes significantly better than the 0 baseline; ⋆ denotes significantly worse than LGCP text ∘ time, according to the Wilcoxon signed rank test $p < 0.05$. All except the final result are taken from Chapter 5. We omit the LL evaluation for non-probabilistic models and for GP, which returns a predictive distribution over a continuous support, thus not directly comparable against other models.

composed of 13,002 tweets. For both datasets, in order to reduce feature sparsity, we replace words with their Brown cluster ids, using bag-of-clusters in place of bag-of-words. We used 1000 clusters acquired on a large scale Twitter corpus (Owoputi et al., 2013), following the approach from Chapter 5.

Rumour Popularity Prediction The first task is rumour popularity prediction, as introduced in Chapter 5. Consider a set of rumours $R = \{R_m\}_{m=1}^{|R|}$, in which each rumour is represented by a set of posts $\{\mathbf{p}_m^n\}_{n=1}^{|R_m|}$. The problem is as follows: given the first hour of posts from a rumour R_m, predict the counts of tweets in each of the 6-minutes intervals from the second hour. As before, the predictions are evaluated using two metrics: mean squared error (MSE) over each test interval, and the predictive log-likelihood (LL), used for probabilistic approaches only. Evaluation uses a leave one out method, where the prediction is made for each rumour, using the first hour of tweets for all rumours as well as (depending on whether a single-task or multi-task method is considered) the full two hours of tweets for the remaining 113 rumours.

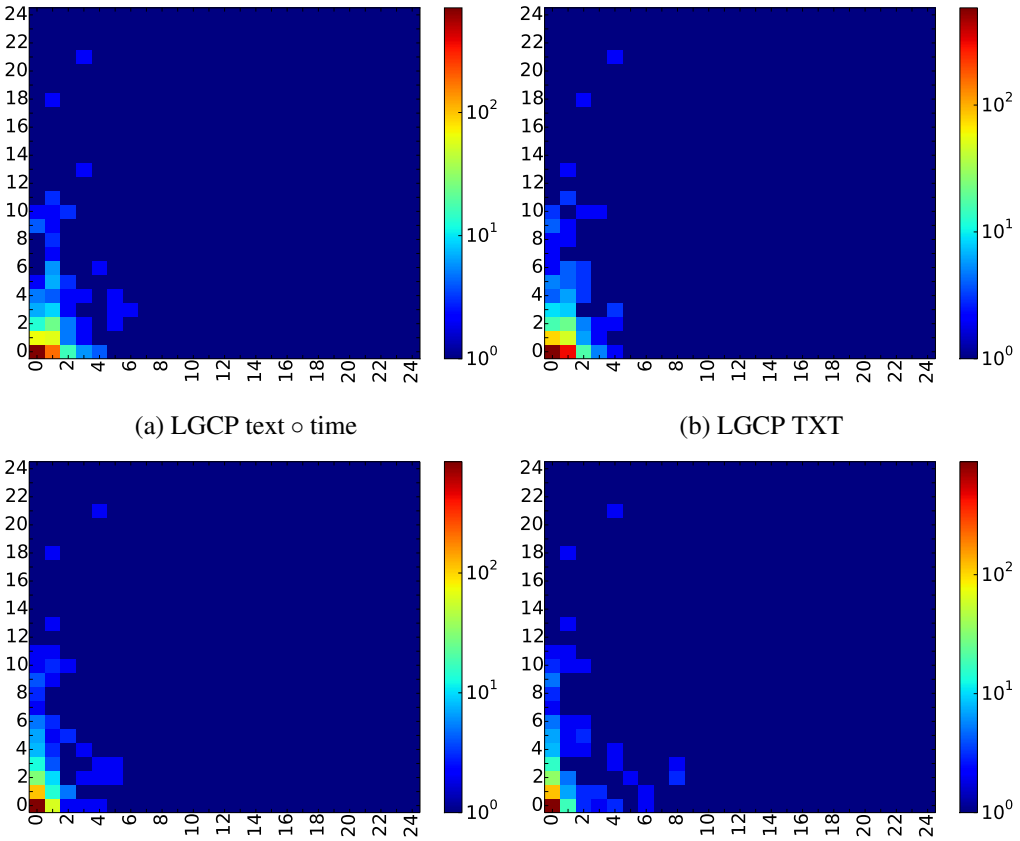
We incorporate baselines from Chapter 5: Homogenous Poisson Process (denoted HPP), Gaussian process regression (GP), Interpolation and a 'predict no tweets' method (0), all of

which are single task methods. The most successful benchmark methods are based on log-Gaussian Cox Processes. The first approach uses an RBF kernel to independently model each target rumour (denoted LGCP). The remaining methods use multi-task learning over the collections of rumours, using a RBF kernel across time intervals combined in a product with a rumour kernel. The rumour kernel uses the rumour text (TXT) and/or explicit learning of a low-rank matrix of rumour correlations (ICM). For more information about these baselines and benchmark systems, we refer the reader to Chapter 5.

We report the results on the 114 Ferguson rumours in Table 6.5, the same dataset as previously reported in Table 5.1. We found the convolution kernel to work best without normalization of rumours by their lengths, and we report results from such an approach. LGCP text $\circ$ time outperforms all methods according to both evaluation metrics. The improvements of LGCP text $\circ$ time over LGCP TXT and LGCP ICM+TXT are however not statistically significant according to the Wilcoxon signed rank test, which shows that there are multiple rumours for which the convolution kernel does not improve the results. Notice how for both metrics the predictive variance is equal to or smaller than the best benchmark results. Thus LGCP text $\circ$ time not only outperforms the previous best method (although not significantly), but also avoids the worst mistakes.

In Figure 6.4 we show the confusion matrices comparing the text $\circ$ time approach as well as three other methods for scoring similarities between pairs of rumours that we considered in Chapter 5. Note that the value $(1, 0)$ is the smallest for text $\circ$ time, showing that compared to other methods text $\circ$ time makes fewer mistakes in misclassifying count 1 as 0. Moreover, both cells $(1, 1)$ and $(2, 2)$ have higher values for text $\circ$ time than for ICM+TXT, indicating that the convolution kernel makes better predictions for the small counts. Also, compared to all other methods, the matrix corresponding to text $\circ$ time exhibits low densities in the middle range of the column corresponding to frequency 0, which means that the convolution kernel does not suffer from predicting 0 counts for non-zero intervals as much as the baselines. However, we also observe unwanted phenomena for the text $\circ$ time, such as the fact that cell $(3, 6)$ is non-zero only for this method, and only ICM kernel yielding a bigger overprediction for intervals of count 3. Another peculiarity of text $\circ$ time is a higher error made for the highest ground truth frequency: text $\circ$ time misclassifies 21 as 3, whereas ICM+TXT misclassifies 21 as 4, thus making a slightly lower error. Nevertheless, overall we can see better patterns in predictions made by text $\circ$ time than any other kernel.

Frequency profiles for text $\circ$ time, TXT and ICM methods are depicted in Figure 6.5 in the example of multiple rumours. Notice that in many cases both TXT and ICM underestimate the counts more than text $\circ$ time does (see rumours #0, #14, #24, #48, #49, #53, #62, #110), even though the error often remains significant for all approaches. We also notice



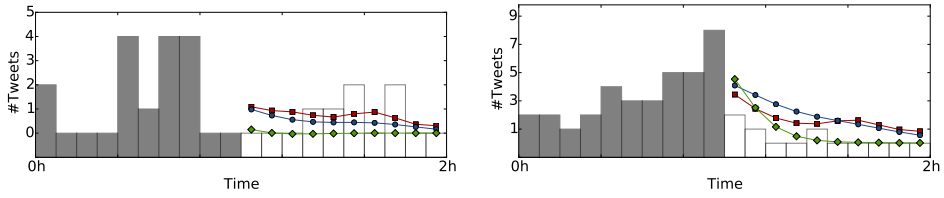
(a) LGCP text ∘ time

(b) LGCP TXT

(c) LGCP ICM+TXT

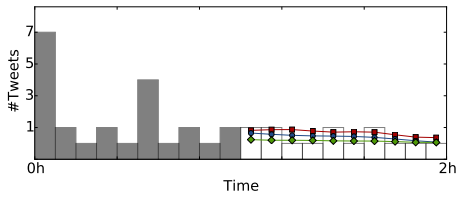
(d) LGCP ICM

Figure 6.4: Confusion matrices for LGCP text ∘ time, LGCP TXT, LGCP ICM+TXT and LGCP ICM. Each row corresponds to the true count of tweets in an interval, and each column corresponds to the predicted number of tweets. A cell i, j denotes how many times the ground truth count of tweets i is being classified as target count of tweets j.

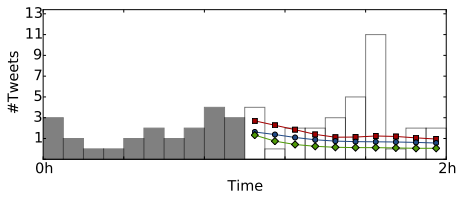


(a) rumour #0

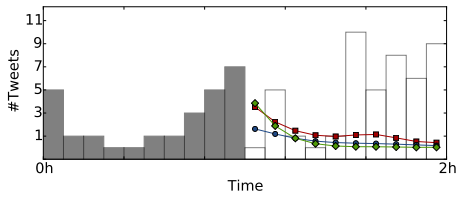
(b) rumour #12



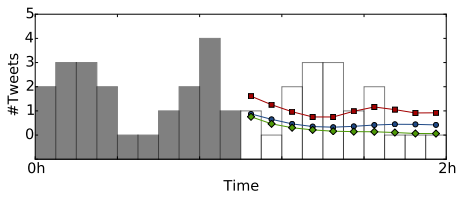
(c) rumour #14



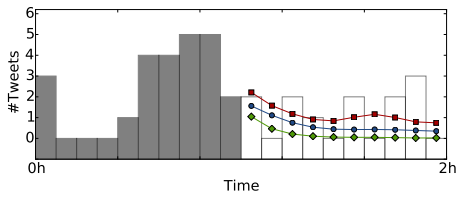
(d) rumour #24



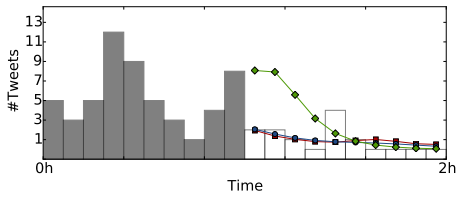
(e) rumour #48



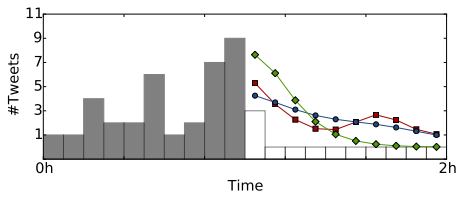
(f) rumour #49



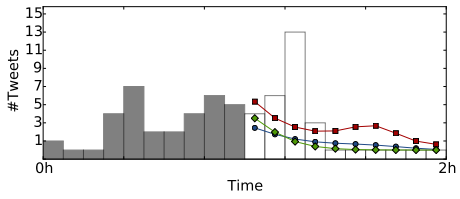
(g) rumour #53



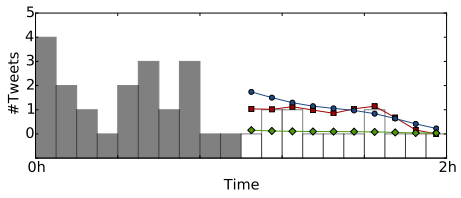
(h) rumour #57



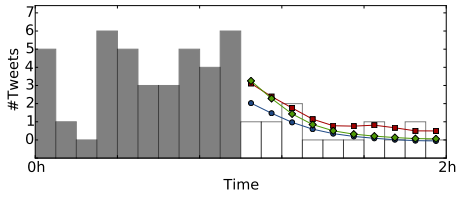
(i) rumour #60



(j) rumour #62



(k) rumour #75



(l) rumour #110

Figure 6.5: Intensity functions for different methods across example Ferguson rumours in the extrapolation setting. Dark bars denote frequencies of tweets in the train intervals, and white bars denote counts of tweets in the test intervals. Predictions from LGCPTXT are denoted by blue circles, from LGCPICM by green diamonds, and from LGCP text $\circ$ time by red squares.

cases where text ∘ time overestimates the counts, e.g. for rumour #62 it makes a spurious prediction of an increase in frequency of posts towards the end of the 2nd hour, presumably due to similarities captured by the convolution kernel which did not correspond to common future rumour trajectories.

Rumour Classification The second problem we consider is about predicting whether a rumour will be popular, framed as classification. This case study is applicable when authorities or journalists wish to track social media, e.g. for identifying rumours that are going to become popular. Here we use the rumour dataset consisting of the Ottawa and Ferguson rumours, as described in Section 2.5.2. We consider this problem for demonstrating the usefulness of the convolution kernel for different kinds of tasks, as well as to facilitate easier error analysis.

As with the earlier rumour dataset, we consider a set of rumours and based on the first hour of posts from rumour i, the task is to predict if the number of posts in the second hour equals or exceeds a threshold τ.[6] This results in a binary classification problem, in which we predict which rumours are likely to become or remain popular. For most rumours we observe a rapid decrease in the counts of posts over time: the mean of the number of posts in the first hour is 11.08, and only 1.91 in the second hour. For this reason we set the threshold to $\tau = 2$, which results in around 30% of rumours being labelled as positive instances, i.e., popular.

We model the data using supervised GP classification, and evaluate the predictive accuracy with 5-fold cross validation, reporting our results in Table 6.6. The best accuracy is achieved for the convolution kernel text ∘ time, achieving the score of 0.748. Although this is only a small improvement over the mean prediction from the time kernel, notice that the results were much more robust than the other methods (text ∘ time kernel has by far the lowest standard deviation). Overall this is a very difficult modeling problem, although our methods were able to improve over the majority baseline with the best method providing a relative error reduction of 16%. Note that the fusion kernel text+time performs worse than text ∘ time, reinforcing our conclusions from synthetic experiments that it is a less powerful method in capturing the dynamics of multivariate time series.

Next, we inspect how errors vary across rumour lengths for the text ∘ time kernel. In Figure 6.6 we plot the probability of being assigned a wrong class to rumours according to the GP output (in the y axis) against the number of posts on a rumour (in the x axis).

[6] When discussing the research question "**How can the rumour popularity prediction problem be formulated?**" in Section 1.2 we argued against such a formulation of the rumour popularity prediction problem. Here we consider it to demonstrate applicability of our convolution kernels to classification problems.

	ACC±std dev
majority	0.701±0.021
text	0.678±0.025
time	0.740±0.015
text+time	0.738±0.011
text ∘ time	0.748±0.009

Table 6.6: Results from the rumour classification experiments for a range of methods, showing classification accuracy±std dev. The results are averaged using five-fold cross validation.

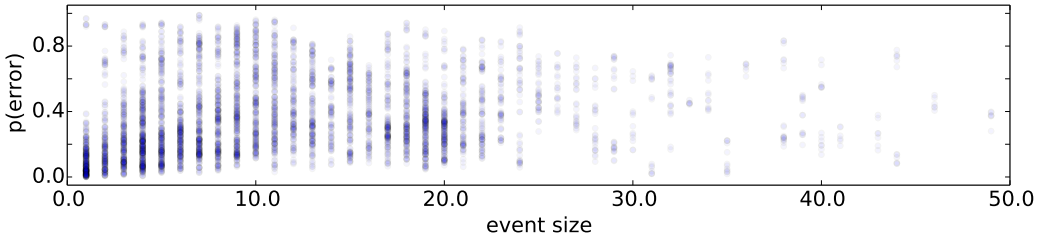


Figure 6.6: Error analysis of rumour classification, showing the probability of error versus rumour lengths (counts of posts in their first hour) for GP classification with the text∘time kernel. The x axis has been truncated to interval $[0, 50]$, which covers all but 37 rumours in our test set (which were all correctly classified).

Note that there are consistently more rumours correctly versus incorrectly classified across all rumour sizes. Even though there is a group of wrongly predicted rumours of small sizes (upper-left part), there are many more small rumours that are correctly classified (lower-left part). them.

6.7 Conclusions

In this chapter we introduced convolution kernels for discriminative modelling of time series. We showed that the kernels work well on the rumour popularity prediction task introduced in Chapter 5. We also evaluated the kernels on synthetic datasets, demonstrating with intuitive examples where one can expect the kernels to perform well, as well as indicating the applicability of convolution kernels to other types of problems.

In this and previous chapters we considered modeling of rumour popularity with point processes. In the next chapter we return to the rumour stance classification problem, which

we previously considered in Chapter 4. We will consider whether explicitly modeling rumour dynamics via point processes can also be leveraged for that task.

Chapter 7

Temporal Dynamics for Rumour Stance Classification

Chapters 5 and 6 dealt with the problem of modeling rumour popularity. We showed that the text content from tweets around a rumour and the dynamics of the rumour spread correlates with the temporal dynamics in unobserved periods of time. In this chapter we consider the idea of employing the rumour dynamics information in the context of the first research aim, **predicting rumour popularity**. In particular, we seek an answer to the last research question we posed:

Do tweet arrival times carry complementary information to text for the stance prediction task?

In particular, we show how temporal information can be used within the Hawkes process framework for predicting stance labels, leading to an approach outperforming baselines not using it.

Parts of this chapter have been published as

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

7.1 Introduction

Sequence classification tasks are often associated with temporal information, where the timestamp is available for each of the data instances. For instance, in sentiment classifica-

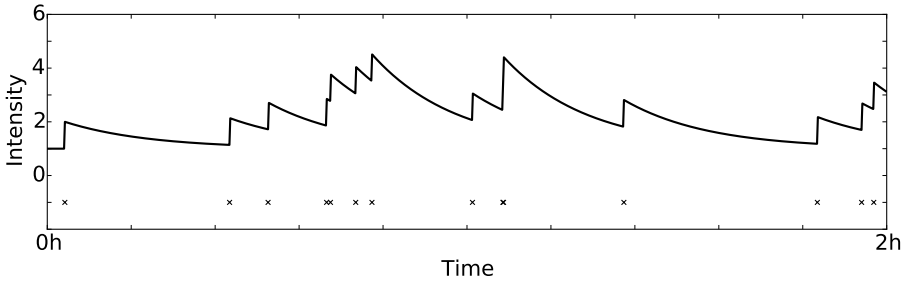


Figure 7.1: A sample drawn from a univariate Hawkes process (denoted by crosses at the bottom of the figure) and corresponding intensity function values over time. The samples have been obtained using Ogata thinning algorithm (Ogata, 2006), and the parameters of the Hawkes process are: $\mu = \gamma = 1$ and $\omega = 1$.

tion of reviews in forums, opinions of users are associated with a timestamp, indicating the time at which they were posted. Similarly, in event detection in Twitter, tweets being posted on a continuous basis need to be analysed and classified in order to detect the occurrence of some event. Nevertheless, traditional sequence classification approaches ignore the time information in these textual data sequences (Song et al., 2014; Gorrell and Bontcheva, 2016), which has also been the case for previous work on rumour stance classification (Qazvinian et al., 2011; Zeng et al., 2016b), including our own as reported in Chapter 4. In this chapter we consider continuous time information along with the textual information for classifying sequences of temporal textual data. In particular, we consider the problem of rumour stance classification in Twitter, where tweets provide temporal information associated with the textual tweet content. We introduced the rumour stance classification problem in Chapter 4, where we took an approach based on Gaussian processes. In this chapter we propose to use Hawkes processes (Hawkes, 1971), commonly used for modelling information diffusion in social media (Yang and Zha, 2013; De et al., 2015) (we introduced HP in Section 3.2.5). Hawkes processes (HP) are a self-exciting temporal point process which has been shown useful for modelling the occurrence of posts in Twitter (Zhao et al., 2015b). The model assumes that the occurrence of a tweet will influence the rate at which future tweets will arrive. Figure 7.1 shows the behaviour of the intensity functions associated with a Hawkes process. Note the intensity spikes at the points of tweet occurrences. In applications such as stance classification, different labels can influence one another. This can be modelled effectively using the mutually exciting behaviour of a multivariate Hawkes process, as depicted in Figure 7.2, where multiple intensity functions are modeled jointly for parallel sub-streams of tweets. In the end, we demonstrate how rumour dynamics convey essential information for

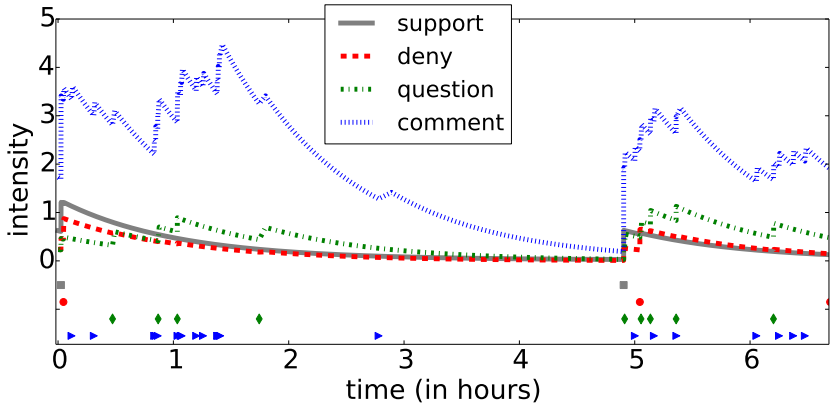


Figure 7.2: Intensities of the Hawkes processes corresponding to the four stances for an example Ferguson rumour with parameters trained in the *HP Grad.* approach. Tweet occurrences over time are denoted at the bottom of the figure by different symbols: grey squares for commenting tweets, red circles for denying tweets, greed diamonds for questioning tweets and blue triangles for commenting tweets. Notice how the intensity corresponding to the Hawkes process for the commenting stance is high throughout the rumour lifespan.

the problem, thus showing that text does not hold exclusively the information for the task of stance classification, which previous work (including ours from Chapter 4) was limited to in their study.

In Chapters 5 and 6 we employed the log-Gaussian Cox Process (LGCP) for modeling rumour dynamics. For modeling the dynamics of rumours for rumour stance classification we resort to a different model, the Hawkes process. Even though in principle it might be possible to apply LGCP for this task, we claim that Hawkes processes are a more natural approach to tackle this problem, for the following reasons:

1. Hawkes processes come with an assumption suitable for this particular application: they explicitly model the causality between tweet occurrences within a rumour. The way we used LGCP for modeling rumour dynamics in other chapters was different — we were seeking similarities in how different rumours spread over time to make predictions about future trajectories.

2. Hawkes processes are better amenable to sequence classification. They incorporate the history of tweet occurrences in the past explicitly within the shared intensity function (see Equation (7.1)). This is done by conducting a summation over influences coming from previous tweets, thus providing a dependence on previous labelings. In LGCP this could be simulated by expanding the training dataset of the underlying Gaussian process every time a consecutive tweet is labeled, however this is less nat-

ural and more computationally demanding. We elaborate on how we achieve this for Hawkes processes in Section 7.3.3.

The novel contributions of this chapter are:

1. Developing a Hawkes process model for time sensitive sequence classification.
2. Demonstrating how temporal dynamics convey important information for the rumour stance classification task.
3. Broadening the set of labels considered in previous work to include a new label *commenting*. The label is very common in our rumour datasets (see Table 2.6 for statistics about distributions of different labels).

Software used for experiments can be found at `https://github.com/mlukasik/seqhawkes`.

7.2 Problem Settings

We formalized the problem of rumour stance classification in Section 4.2. Here, we revisit the formulation to introduce the temporal feature of tweets, a source of information that we only include in this chapter. We consider a collection of rumours $R = \{R_1, \cdots, R_{|R|}\}$. Each rumour R_m consists of tweets, $R_m = \{\mathbf{p}_m^1, \ldots, \mathbf{p}_m^{|R_m|}\}$. Here, for brevity we suppress the indexing on rumour id, and represent tweets as tuples $\mathbf{p}_n = (t_n, \boldsymbol{w}_n, m_n, y_n)$. A tuple includes the following information: t_n is the posting time of the tweet, $\boldsymbol{w}_n$ is the text message, m_n is the rumour id (i.e. conversation thread) and y_n is the label, $y_n \in Y$ = {supporting, denying, questioning, commenting}.[1] Notice the commenting label is being introduced compared to Chapter 4 and to previous work on rumour stance classification (Qazvinian et al., 2011; Zeng et al., 2016b). The definitions of all stances, including the commenting stance, can be found in Section 2.2.

LOO setting We consider the Leave One Out (LOO) setting, introduced in Section 4.2. In the LOO setting, for each rumour $R_m \in R$ we construct the test set equal to R_m and the training set equal to $P \setminus R_m$ (where P is the set of posts from all rumours). The final performance scores we report are averaged across all rumours. This represents a realistic scenario where a classifier has to deal with a new, unseen rumour.

[1] Recall our notation is summarized in the Nomenclature on page xii.

Data In Chapter 4 we evaluated rumour stance classification on two datasets: the England riots and PHEME. Our Hawkes process model requires observation of all tweets from a rumour within the interval of time. This assumption is not satisfied by the England riots dataset, for which some tweets are missing. However, the PHEME datasets define a rumour as a conversation thread in Twitter, which is fully observed during data collection. Therefore, it is applicable for the assumptions entailed by our Hawkes process model. Since we only consider PHEME datasets, we treat each dataset separately, and conduct the leave one rumour out evaluation.[2] Each individual rumour in the PHEME datasets consist of a small number of tweets (as reported in Table 2.6), thus we focus on the LOO approach which does not require any target rumour annotation. Note that, similarly as in the evaluation in Chapter 4, in our evaluation future rumours may be used to predict the past.

Notice that here we extend the set of three labels used before (in the literature (Qazvinian et al., 2011) and in Chapter 4), adding the new label *commenting*. Excluding the commenting label would lead to missing tweets in the stream of posts around a rumour, violating the assumptions made by our model.

7.3 Model

Below we describe our model, which allows for incorporating two phenomena that have been ignored in previous work on rumour stance classification. First, we make use of the **continuous timestamp** information which describes a tweet, a feature which has not been used before for the rumour stance classification task. This information is not easy to use, as it is not clear how using the timestamp directly as a feature could help predict the label of a tweet. Instead, we make use of timestamps indirectly, by modeling dynamics of label occurrences, allowing for elegant incorporation of timestamp information into the model. Moreover, when predicting a label for each tweet, we make use of labels of preceding tweets. This may yield better results, since information about the **stances adjacent in time** adds a useful information for what stance a particular tweet may take. For example, if there are many supporting tweets in an interval of time, it might be likely that right afterwards another supporting post would be tweeted.

Hawkes processes are a probabilistic framework for modelling self-excitatory phenomena, which has been used for modelling memes and their spread across social networks (Yang and Zha, 2013). The intensity function of a Hawkes process models the self-exciting property by adding up the influence from past tweets. We use a multi-variate Hawkes pro-

[2]For this reason we use the four largest datasets, excluding the Germanwings crash dataset as being too small for an independent dataset.

cess for modelling the mutually exciting phenomena between the tweet labels. We described the fundamentals of point processes and Hawkes processes in Section 3.2. In this section we describe how we apply the Hawkes process framework for rumour stance classification.

7.3.1 Intensity Function

The Hawkes process, a form of a point process, models point occurrences over a space via an intensity function (for an introduction to the role of intensity function in modeling point processes see Section 3.2). Here, we define an intensity function for each rumour-stance pair. In the intensity function formulation, we assume that all previous tweets associated with a rumour m influence the occurrence of a new tweet. This allows to use information from other tweets that have been posted about rumour m. We consider the intensity function for stance y and rumour m to be a summation of the base intensity and the intensities associated with all the previous tweets about the same rumour,

$$\lambda_{y,m}(t|\mathbf{H}_{t^-}) = \mu_y + \sum_{t_\ell<t} \mathbb{I}(m_\ell = m)\alpha_{y_\ell,y}\kappa(t-t_\ell), \tag{7.1}$$

where the first term represents the constant base intensity of generating label y, and l iterates over events which happened before time t. The history of previous tweets and their assigned labels by time t is denoted by $\mathbf{H}_{t^-}$. The matrix α of size $|Y| \times |Y|$ encodes the degrees of influence between pairs of labels assigned to the tweets, e.g. a *questioning* label may influence the occurrence of a *rejecting* label in future tweets differently from how it would influence a *commenting* label. Notice how the same parameters are used across different rumours. The second term represents the influence from the tweets that happen prior to the time of interest. The influence from each tweet decays over time and is modelled using an exponential decay term,

$$\kappa(t-t_\ell) = \omega \exp\left(-\omega(t-t_\ell)\right). \tag{7.2}$$

A graphical representation of the proposed Hawkes process sequence classification model is shown in Figure 7.3. Here, we consider a sequence of 4 tweets occurring at times t_1, t_2, t_3, t_4 with labels y_1, y_2, y_3, y_4. Each label y_n corresponding to a tweet occurring at time t_n depends on all previous labels as well as the times at which previous labels have occurred. Hence this is a non-Markovian model, as all the preceding labels and timestamps influence the following label. The exponential kernel decay causes the effect of previous labels diminish over time, with most recent labels having the strongest effect on the current label. This is a reasonable assumption; in particular, in the rumour stance classification task

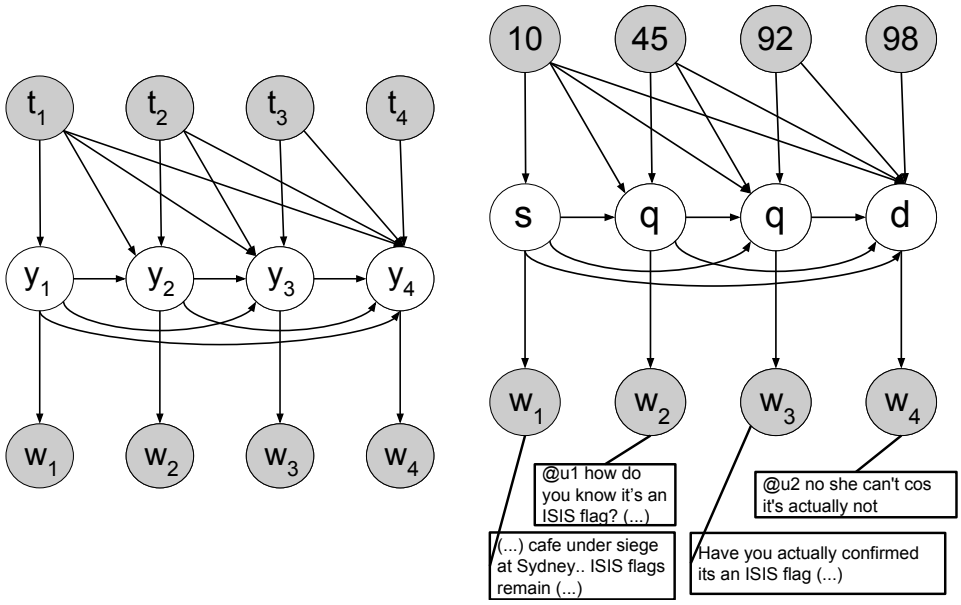


Figure 7.3: Graphical representation of Hawkes process sequence classification model (left) and an example instantiation of variables from the graphical model (right). Timestamps are denoted by variables t_i, labels by variables y_i, and text by variables w_i. Shaded circles correspond to observed variables, and white circles correspond to variables unobserved during test time. The example corresponds to the rumour about an ISIS flag remaining on display during the Sydney siege event (note this rumour has also been depicted in Figure 1.1). Timestamps expressed in relative time in minutes from the beginning of the rumour spread; *s* denotes a supporting, *q* a questioning and *d* a denying stance. Notice that in this chapter we also consider the commenting stance.

the most recent stances would have a bigger influence than earlier stances on the stance of the current tweet. Moreover, note that the dependence between labels is weighted by their influence as measured by parameter α (as specified in Equation (7.1)). Thus, even if a preceding label has occurred in a distant past but the corresponding entry in α is high, it will play a significant role in determining the current label.

7.3.2 Likelihood

The parameters governing the intensity function are learnt by maximizing the likelihood of generating the tweets. The complete likelihood, as derived in Section 3.2, is given by

$$L(\boldsymbol{t}, \boldsymbol{y}, \boldsymbol{m}, W|\boldsymbol{\beta}, \boldsymbol{\mu}, \boldsymbol{\gamma}, \boldsymbol{\alpha}, \omega) = \prod_{n=1}^{N} p(\boldsymbol{w}_n|y_n) \times \left(\prod_{n=1}^{N} \lambda_{y_n,m_n}(t_n|\mathbf{H}_{t_n^-}) \right) \times p(E_T), \quad (7.3)$$

where the first term provides the likelihood of generating text given the label (we denote the nth row of matrix W by $\boldsymbol{w}_n$). This language component is modelled as a multinomial distribution conditioned on the label,

$$p(\boldsymbol{w}_n|y_n) = \prod_{v=1}^{V} \beta_{y_n v}^{W_{nv}}, \quad (7.4)$$

where V is the vocabulary size and β is the matrix of size $|Y| \times V$ specifying the language model for each stance. The second term provides the likelihood of occurrence of tweets at times $t_1, \ldots, t_N$ and the third term provides the likelihood that no tweets happened in the intervals between tweets within the time horizon $[0, T]$. See Section 3.2 for explanation of how the joint likelihood of points under a point process model is derived. The values of the second term over time are illustrated in Figure 7.2 for an example Ferguson riots rumour. At each point of time, each of the four intensity functions is influenced by the preceding tweets. Notice how tweets from different stances influence each of the intensity functions differently, which is due to different cell values in the α parameter matrix.

7.3.3 Prediction

The prediction task we consider is to infer the stances expressed by tweets. We employ the model for this task by conducting a sequence classification, where for each tweet we predict the stance maximizing the joint likelihood from equation (7.3) of a sequence of

tweets ending at this particular tweet. Thus, when predicting the label for the kth tweet,

$$\hat{y_k} = \arg\max_{y \in Y} L\left((t_1, \ldots, t_k), (y_1, \ldots, y_{k-1}, y), (m_1, \ldots, m_k), (\boldsymbol{w}_1, \ldots, \boldsymbol{w}_k)\right). \quad (7.5)$$

The classification of a sequence of tweets is conducted in a greedy, sequential manner. This means that once a label is chosen for a tweet, it is fixed and used for finding stances of the following tweets. Notice this is suboptimal. An approach which is optimal from the model's perspective would be to select a sequence of stances maximizing the joint likelihood, although this would be prohibitively complex due to the non-Markovian nature of the technique. Another approach could be to use a beam search (Norvig, 1992), trying to achieve a compromise between speed and size of the search space of the inspected solutions, which is an appealing avenue for future work.

Informally speaking, our approach to classification can be viewed as corresponding to a Naive Bayes classifier, where a constant label prior is replaced by a time-varying prior dependent on the previous tweets. In particular,

$$\hat{y_k} = \arg\max_{y \in Y} L\left((t_1, \ldots, t_k), (y_1, \ldots, y_{k-1}, y), (m_1, \ldots, m_k), (\boldsymbol{w}_1, \ldots, \boldsymbol{w}_k)\right) \quad (7.6)$$

$$= \arg\max_{y \in Y} \left(\prod_{n=1}^{k-1} p(\mathbf{w_n}|y_n)\right) p(\mathbf{w_k}|y) \times \left(\prod_{n=1}^{k-1} p_{y_n,m_n}\left(t_n|\mathbf{H}_{t_n^-}\right)\right) p_{y,m_k}\left(t_k|\mathbf{H}_{t_k^-}\right) \quad (7.7)$$

$$= \arg\max_{y \in Y} p(\boldsymbol{w}_k|y) \times p_{y,m_k}\left(t_k|\mathbf{H}_{t_k^-}\right), \quad (7.8)$$

where $p_{y,m}(t)$ is the pdf corresponding to stance y and rumour m.

7.3.4 Parameter Optimization

We estimate the parameters of the model by maximizing the joint log-likelihood of the tweets under the Hawkes process,

$$l(\boldsymbol{t}, \boldsymbol{y}, \boldsymbol{m}, W) = -T|R| \sum_{y=1}^{|Y|} \mu_y - \sum_{y=1}^{|Y|} \sum_{\ell=1}^{N} \alpha_{\alpha_{y_\ell,y}} K(T - t_\ell) + \sum_{n=1}^{N} \log\left(\lambda_{y_n,m_n}(t_n|\mathbf{H}_{t_n^-})\right) + \sum_{n=1}^{N} \sum_{v=1}^{V} W_{nv} \log \beta_{y_n v}, \quad (7.9)$$

where $K(T - t_\ell) = 1 - \exp(-\omega(T - t_\ell))$ (see Appendix A for the derivation). Note that β is independent from the first two terms. Equating the derivative of the log-likelihood with respect to β to 0 leads to a closed form solution, which after applying the Laplacian smoothing (Manning et al., 2008) takes the form:

$$\beta_{yv} = \frac{\sum_{n=1}^{N} \mathbb{I}(y_n = y) W_{nv} + 1}{\sum_{n=1}^{N} \sum_{v=1}^{V} \mathbb{I}(y_n = y) W_{nv} + V}.$$

Optimization with respect to ω is non-convex. Similar to Yang and Zha (2013), we fix the decay parameter ω, in our case to 0.1, and leave investigation of ways for optimizing for ω to future work (we study the sensitivity of the model to ω in Figure 7.5). Now, the challenge in optimizing for μ and α comes from the $\log$ term in Equation (7.9), as it does not lead to closed form solutions.

Approximation Based Optimization (*HP Approx.*) In one approach to μ and α optimization we approximate the $\log$ term in Equation (7.9) by taking the $\log$ inside the summation terms in Equation (7.1). Notice that this approximation is equivalent to assuming that the intensity function instead of taking the form from Equation (7.1) takes the form:

$$\hat{\lambda}_{y,m}(t|\mathbf{H}_{t^-}) = \mu_y \times \prod_{t_\ell < t} \mathbb{I}(m_\ell = m) \alpha_{y_\ell, y} \kappa(t - t_\ell), \tag{7.10}$$

meaning that the influences from the previous tweets are multiplied instead of being added. The log-likelihood under this assumption takes the form

$$\begin{aligned} l(\boldsymbol{t}, \boldsymbol{y}, \boldsymbol{m}, W|\boldsymbol{\beta}, \boldsymbol{\mu}, \boldsymbol{\gamma}, \boldsymbol{\alpha}, \omega) = &-T|R| \sum_{y=1}^{|Y|} \mu_y - \sum_{y=1}^{|Y|} \sum_{\ell=1}^{N} \alpha_{\alpha_{y_\ell, y}} K(T - t_\ell) \\ &+ \sum_{n=1}^{N} \left(\log \mu_{y_n} + \sum_{t_\ell < t} \mathbb{I}(m_\ell = m) \log(\alpha_{y_\ell, y} \kappa(t - t_\ell)) \right) \\ &+ \sum_{n=1}^{N} \sum_{v=1}^{V} W_{nv} \log \beta_{y_n v}, \end{aligned} \tag{7.11}$$

which leads to closed form updates for μ and α obtained after equating the corresponding derivatives to 0:

$$\mu_y = \frac{\sum_{n=1}^{N} \mathbb{I}(y_n = y)}{T|R|},$$

$$\alpha_{xy} = \frac{\sum_{n=1}^{N}\sum_{l=1}^{n}\mathbb{I}(m_l{=}m_n)\mathbb{I}(y_l = x)\mathbb{I}(y_n{=}y)}{\sum_{k=1}^{N}\mathbb{I}(y_k = x)K(T - t_k)}.$$

Gradient Based Optimization (*HP Grad.*) Without any approximation, the log-likelihood of our model takes the form

$$\begin{aligned} l(\boldsymbol{t}, \boldsymbol{y}, \boldsymbol{m}, W|\boldsymbol{\beta}, \boldsymbol{\mu}, \boldsymbol{\gamma}, \boldsymbol{\alpha}, \omega) = -T|R|\sum_{y=1}^{|Y|}\mu_y - \sum_{y=1}^{|Y|}\sum_{\ell=1}^{N}\alpha_{\alpha_{y_\ell,y}}K(T - t_\ell) \\ + \sum_{n=1}^{N}\log\left(\mu_y + \sum_{t_\ell<t}\mathbb{I}(m_\ell = m)\alpha_{y_\ell,y}\kappa(t - t_\ell)\right) \quad (7.12) \\ + \sum_{n=1}^{N}\sum_{v=1}^{V}W_{nv}\log\beta_{y_n v}. \quad (7.13) \end{aligned}$$

The optimization of log-likelihood under the Hawkes process model as specified in Equation (7.13) is a sum of linear and logarithms of linear functions of the parameters. Both linear functions and logarithms of linear functions are concave, and sum of concave functions results in a concave function (Boyd and Vandenberghe, 2004). Therefore, the log-likelihood under the Hawkes process model is jointly concave with respect to μ and α.

In our second approach to optimizing the parameters (*HP Grad.*) we find parameters using joint gradient based optimization over μ and α, using the partial derivatives of the log-likelihood, $\frac{\partial l}{\partial \mu}$ and $\frac{\partial l}{\partial \alpha}$. The gradients of the log-likelihood under the Hawkes process model with respect to μ and α are given by:

$$\frac{\partial l}{\partial \mu_y} = -T|R| + \sum_{n=1}^{N}\frac{\mathbb{I}(y_n = y)}{\mu_{y_n} + \sum_{t_\ell<t}\mathbb{I}(m_\ell = m)\alpha_{y_\ell,y}\kappa(t - t_\ell)} \quad (7.14)$$

$$\frac{\partial l}{\partial \alpha_{x,y}} = -\sum_{\ell=1}^{N}\mathbb{I}(y_l = x)\alpha_{\alpha_{y_\ell,y}}K(T - t_\ell) \quad (7.15)$$

$$+ \sum_{n=1}^{N}\frac{\mathbb{I}(y_n = y)\sum_{t_\ell<t}\mathbb{I}(m_\ell = m)\mathbb{I}(y_\ell = x)}{\mu_{y_n} + \sum_{t_\ell<t}\mathbb{I}(m_\ell = m)\alpha_{y_\ell,y}\kappa(t - t_\ell)} \quad (7.16)$$

In optimization, we represent both μ and α as exponentials to ensure positivity, and employ L-BFGS approach to gradient based optimization. Note that since we do not obtain closed form solutions, this approach is more computationally complex than the *HP Approx.* approach (with naive implementation of *HP Approx.* taking $O(N^2)$ time, and naive implementation of *HP Grad.* taking $O(N^2)$ time per a single step of the gradient optimization).

7.4 Experiments

We conduct experiments using the PHEME rumour datasets. We consider our Hawkes process model described in Section 7.3 as well as a set of baseline approaches.

7.4.1 Baselines

We compare our model against the following baselines:

CLASS CONDITIONED LANGUAGE MODEL (*Class LM*) considers only the textual information through the multinomial distribution defined in Equation (7.4). This is equivalent to fixing μ and α parameters to 0.

MAJORITY VOTE classifier based on the training label distribution, which corresponds to Hawkes process where only μ is optimized, while fixing other parameters to 0.

NAIVE BAYES models the text using a multinomial likelihood and a prior over label frequencies (Manning et al., 2008). This is equivalent to fixing the α matrix to 0.

MAXENT Logistic Regression was the first method employed for rumour stance classification (Qazvinian et al., 2011). We use ℓ_1 regularisation with the cost coefficient selected via grid search over the training set from the list of values: [0.00001, 0.0001, 0.001, 0.01, 0.1, 1, 10, 100]. Here and in other baselines we do grid search via 3-fold cross-validation over the training set, where two folds are used for training and one for evaluation of the proposal coefficient.

SVM Support Vector Machines with the cost coefficient selected via grid search from the list of values: [0.00001, 0.0001, 0.001, 0.01, 0.1, 1, 10, 100].

RF We showed the Random Forests classifier to be a competitive baseline in the experiments in Chapter 4. We select the hyperparameters of RF via grid search over the cross product between the considered hyperparameter values. The hyperparameters that we consider are: the splitting criterion measuring the quality of a split (optimized for from the list [Gini impurity, entropy]), the number of trees (optimized for from the list [10, 50, 100, 150, 200]), the minimum number of samples in a node to perform a split (optimized for from the list [2, 5, 10]). We use bootstrap samples when choosing data for each tree.

GP-ICM Gaussian processes in conjunction with the ICM multi-task learning kernel (see Section 4.3 for its introduction) was a competitive approach in Chapter 4.

CRF Conditional Random Field (Lafferty et al., 2001) is an algorithm for structured prediction which can be seen as an extension of logistic regression to structured outputs (Sutton and McCallum, 2012). We employ linear chain CRFs over temporally ordered sequences of text represented using bags of Brown clusters

(similarly as in the other approaches). The model uses state (text-stance) and transition (bigrams of consecutive stances) features. We employ ℓ_2 regularisation on the weights with the cost coefficient selected via grid search from the list of values: [0.00001, 0.0001, 0.001, 0.01, 0.1, 1, 10, 100].

Similarly as in the previous chapters, we use the Gpy implementation of Gaussian processes (The GPy authors, 2015), and the Scikit-learn implementations of SVMs, MaxEnt, and RF classifiers (Pedregosa et al., 2011). For experiments with CRFs we use the CRFsuite implementation (Okazaki, 2007).

7.4.2 Results

In this section we report results of experiments on rumour stance classification with the Hawkes process based model. The results are shown in Table 7.1. As in Chapter 4 we report Micro-F1 and Macro-F1 scores for the methods. Notice that here we consider four categories (as opposed to three in Chapter 4, as we added the *commenting* stance). Now, as shown in the stance distributions in Table 2.6, the commenting class is dominating with the supporting class being second most frequent, and overall the denying and questioning classes are in minority (with Sydney Siege dataset being an exception). This class imbalance poses a significant challenge.

In terms of Macro-F1 score, the Class LM turns out to be the best performing method, achieving the best result on three datasets and second best on one. *HP Grad.* turns out to be competitive, as it is always among the three best performing approaches, always outperforming the CRF baseline. GP-ICM achieves the best result on the Charlie Hebdo dataset, and is the second best method on two datasets. On the Ferguson dataset it doesn't do as well, achieving the fourth best result.

The Micro-F1 scores are significantly higher than the Macro-F1 scores, which is the result of the class imbalance of the four stances. In particular, the majority classification already leads to high Micro-F1 score values across the datasets. We can observe that in terms of Micro-F1 score *HP Approx.* beats all other methods on three out of four datasets, and achieves the second score on the Sydney siege dataset after CRFs. Therefore, if what one cares for is to minimize the absolute number of mistakes made, *HP Approx.* is the method of choice. Note the Class LM is the worst according to the Micro-F1 score for this metric, despite being very strong for the Macro-F1 score. This is due to Class LM not learning any prior about the labels, which leads it to making more mistakes overall, while not demoting the minority classes. NB leads to significant improvements over Class LM, which shows that incorporating a prior to language information is essential for making

	Model	Ottawa	Ferguson	Charlie Hebdo	Sydney Siege
Macro-F1	Majority	0.190	0.200	0.201	0.194
	GP-ICM	0.424	0.292	**0.430**	0.407
	MaxEnt	0.343	0.247	0.320	0.379
	SVM	0.354	0.200	0.351	0.377
	RF	0.377	0.279	0.340	0.371
	Class LM	**0.427**	**0.344**	0.428	**0.415**
	NB	0.406	0.313	0.397	0.386
	CRF	0.361	0.293	0.355	0.350
	HP Grad.	0.424	0.331	0.419	0.395
	HP Approx.	0.323	0.260	0.326	0.325
Micro-F1	Majority	0.615	0.669	0.674	0.634
	GP-ICM	0.627	0.648	0.707	0.663
	MaxEnt	0.652	0.668	0.703	0.647
	SVM	0.646	0.669	0.699	0.673
	RF	0.638	0.666	0.703	0.662
	Class LM	0.532	0.496	0.634	0.516
	NB	0.618	0.620	0.702	0.620
	CRF	0.665	0.677	0.680	**0.705**
	HP Grad.	0.634	0.632	0.718	0.630
	HP Approx.	**0.678**	**0.684**	**0.729**	0.686

Table 7.1: Micro-F1 and Macro-F1 scores for different methods and different proportions of the initial tweets annotated from the target rumour/event on the England riots and the PHEME datasets.

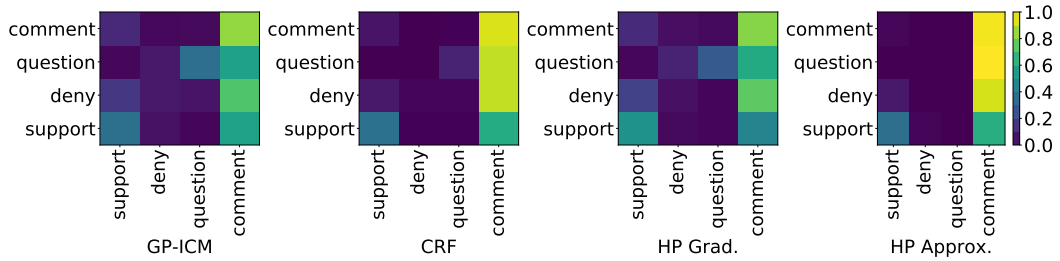


Figure 7.4: Cross-stance confusion rates for competitive methods on the Ottawa dataset. A cell i, j denotes what percentage of times the ground truth stance i is being classified as stance j. The statistics are also reported in Table 7.2. Note stances are imbalanced, with commenting stance being predominant, as reported in Table 2.6, thus different errors contribute differently to the Micro-F1 score.

fewer mistakes and getting higher micro-F1 score. Also observe that CRF outperforms *HP Grad.* on all but Charlie Hebdo dataset, being a strong method according to Micro-F1 score.

Our research question posed in this chapter was whether incorporating temporal information may improve results on the rumour stance classification task. Notice that to this end, comparison of the Hawkes process model against Naive Bayes classifier is the most fair, since both use the same approach to modeling text, and they only differ in how they construct the prior. Namely, NB uses a frequency based prior, whereas *HP Grad.* incorporates temporal information and the dependence on previous labels into it. We notice that *HP Grad.* outperforms NB according to both metrics for all datasets. This is a very encouraging result, since it shows that incorporating temporal information into a prior boosts performance.

The other baselines are competitive, in particular GP-ICM outperforms NB according to Micro-F1 for all datasets, and according to Macro-F1 for all but Ferguson dataset. NB (and Hawkes process based approaches) make a naive assumption when modeling text in that text features (Brown clusters) are independent. This potentially leads to worse results. A good future work direction is considering a different approach to modeling text within the Hawkes process model, thus hopefully leading to even better results while still leveraging the temporal information.

In Figure 7.2 we show plots of the intensity function of the *HP Grad.* model for rumour #1 from the Ferguson dataset. Notice the self-exciting property, with spikes in the intensity functions for different labels at times when tweets occur. Moreover, spikes occur even when a tweet from a different label is posted, for example around 1 hour and 50 minutes into the rumour lifespan a *questioning* tweet is posted which causes a spike in intensity for *commenting* tweets.

Ottawa								
Class	Classifier	Performance			Misclassification rates			
		P	R	F1	S	D	Q	C
supporting (S)	GP-ICM	0.468	0.366	0.411	0.367	0.044	0.012	0.578
	CRF	0.667	0.373	0.478	0.373	0.006	0.006	0.615
	HP Grad.	0.529	**0.509**	**0.519**	0.509	0.025	0.012	0.453
	HP Approx.	**0.806**	0.360	0.498	0.360	0.012	0.000	0.627
denying (D)	GP-ICM	**0.548**	**0.359**	**0.434**	0.158	0.066	0.053	0.724
	CRF	0.500	0.013	0.026	0.066	0.013	0.013	0.908
	HP Grad.	0.097	0.039	0.056	0.197	0.040	0.013	0.750
	HP Approx.	0.000	0.000	0.000	0.066	0.000	0.000	0.934
questioning (Q)	GP-ICM	0.185	0.066	0.097	0.016	0.063	0.359	0.563
	CRF	0.545	0.094	0.160	0.000	0.000	0.094	0.906
	HP Grad.	**0.529**	**0.281**	**0.367**	0.016	0.094	0.281	0.609
	HP Approx.	0.000	0.000	0.000	0.000	0.000	0.000	1.00
commenting (C)	GP-ICM	0.687	0.838	0.755	0.112	0.023	0.027	0.838
	CRF	0.667	0.942	0.781	0.052	0.000	0.006	0.942
	HP Grad.	**0.699**	0.817	0.754	0.119	0.037	0.027	0.817
	HP Approx.	0.667	**0.981**	**0.794**	0.019	0.000	0.000	0.981

Ferguson								
Class	Classifier	Performance			Cross-classification rates			
		P	R	F1	S	D	Q	C
supporting (S)	GP-ICM	0.367	0.137	0.199	0.137	0.000	0.025	0.839
	CRF	0.585	0.193	0.290	0.193	0.000	0.031	0.776
	HP Grad.	0.436	**0.317**	**0.367**	0.317	0.031	0.037	0.615
	HP Approx.	**0.759**	0.137	0.232	0.137	0.000	0.000	0.863
denying (D)	GP-ICM	0.345	**0.106**	**0.163**	0.037	0.012	0.012	0.940
	CRF	**1.000**	0.012	0.024	0.012	0.012	0.024	0.951
	HP Grad.	0.143	0.049	0.073	0.061	0.049	0.049	0.842
	HP Approx.	0.000	0.000	0.000	0.012	0.000	0.000	0.988
questioning (Q)	GP-ICM	0.125	0.012	0.022	0.011	0.011	0.106	0.872
	CRF	0.188	0.032	0.055	0.011	0.000	0.032	0.957
	HP Grad.	**0.190**	**0.085**	**0.118**	0.032	0.011	0.085	0.872
	HP Approx.	0.000	0.000	0.000	0.000	0.000	0.000	1.00
commenting (C)	GP-ICM	0.680	0.921	0.783	0.050	0.009	0.021	0.921
	CRF	0.691	0.962	0.804	0.029	0.000	0.009	0.962
	HP Grad.	**0.699**	0.853	0.768	0.085	0.027	0.035	0.853
	HP Approx.	0.682	**0.991**	**0.808**	0.009	0.000	0.000	0.991

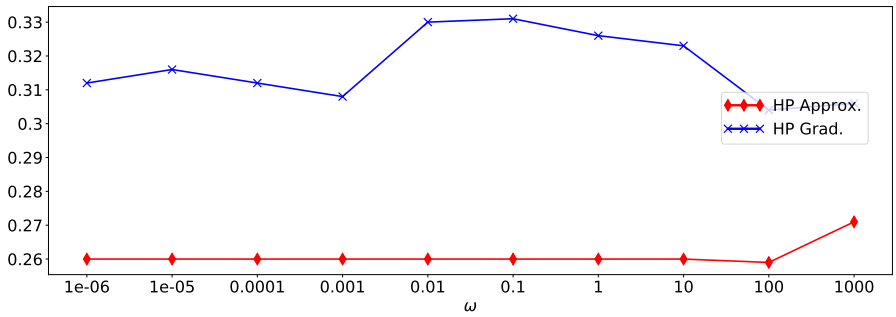
Table 7.2: Per-class precision, recall and F1 scores for the best-performing classifiers (GP-ICM, CRF, *HP Grad.* and *HP Approx.*) on the PHEME datasets. Cross-classification rates denote how often the target stance (denoted by the row) is being misclassified as another stance (denoted by the column). Note stances are imbalanced, with commenting stance being predominant, as reported in Table 2.6. Figure 7.4 graphically illustrates the cross-classification rates for the top approaches on the Ottawa dataset.

Charlie Hebdo								
Class	Classifier	Performance			Cross-stance confusion rates			
		P	R	F1	S	D	Q	C
supporting (S)	GP-ICM	0.559	0.419	0.479	0.420	0.000	0.004	0.576
	CRF	0.595	0.373	0.458	0.373	0.000	0.004	0.623
	HP Grad.	0.592	**0.614**	**0.603**	0.614	0.042	0.000	0.381
	HP Approx.	**0.757**	0.343	0.472	0.343	0.000	0.000	0.657
denying (D)	GP-ICM	**0.552**	**0.314**	**0.400**	0.143	0.018	0.054	0.786
	CRF	0.500	0.018	0.034	0.089	0.018	0.036	0.857
	HP Grad.	0.071	0.018	0.029	0.161	0.018	0.000	0.821
	HP Approx.	0.000	0.000	0.000	0.071	0.000	0.000	0.929
questioning (Q)	GP-ICM	0.167	0.018	0.032	0.020	0.020	0.314	0.647
	CRF	0.273	0.059	0.097	0.020	0.000	0.059	0.922
	HP Grad.	**0.471**	**0.157**	**0.235**	**0.059**	0.020	0.157	0.765
	HP Approx.	0.000	0.000	0.000	0.000	0.000	0.000	100.0
commenting (C)	GP-ICM	0.747	0.885	0.810	0.097	0.006	0.013	0.885
	CRF	0.729	0.915	0.811	0.076	0.001	0.007	0.915
	HP Grad.	**0.775**	0.848	0.810	0.124	0.015	0.013	0.848
	HP Approx.	0.727	**0.968**	**0.830**	0.031	0.001	0.000	0.968
Sydney Siege								
Class	Classifier	Performance			Cross-stance confusion rates			
		P	R	F1	S	D	Q	C
supporting (S)	GP-ICM	0.545	0.404	0.464	0.404	0.018	0.009	0.570
	CRF	0.710	0.341	0.461	0.341	0.009	0.000	0.650
	HP Grad.	0.550	**0.570**	**0.559**	0.570	0.036	0.009	0.386
	HP Approx.	**0.809**	0.323	0.462	0.323	0.000	0.000	0.677
denying (D)	GP-ICM	**0.531**	**0.172**	**0.260**	0.067	0.079	0.034	0.820
	CRF	0.222	0.022	0.041	0.034	0.022	0.022	0.921
	HP Grad.	0.167	0.101	0.126	0.135	0.101	0.056	0.708
	HP Approx.	1.000	0.020	0.040	0.011	0.000	0.000	0.989
questioning (Q)	GP-ICM	0.280	0.079	0.123	0.081	0.030	0.172	0.717
	CRF	**0.438**	0.071	0.122	0.051	0.010	0.071	0.869
	HP Grad.	0.204	**0.111**	**0.144**	0.081	0.051	0.111	0.758
	HP Approx.	0.000	0.000	0.000	0.000	0.000	0.020	0.980
commenting (C)	GP-ICM	0.700	0.885	0.781	0.086	0.015	0.014	0.885
	CRF	0.684	0.952	0.796	0.032	0.006	0.010	0.952
	HP Grad.	**0.715**	0.787	0.749	0.118	0.045	0.051	0.787
	HP Approx.	0.675	**0.978**	**0.798**	0.022	0.000	0.000	0.978

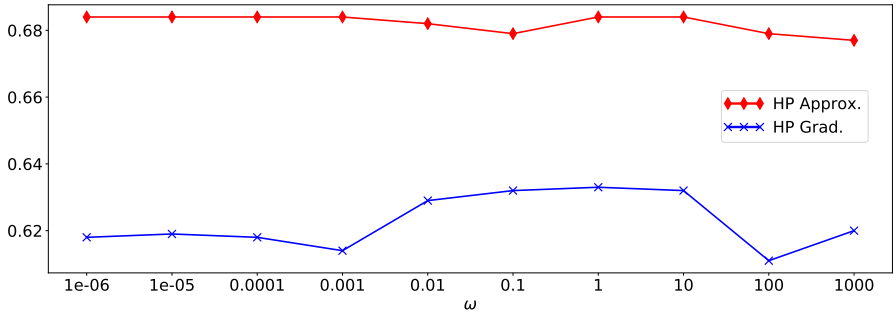
Table 7.3: Per-class precision, recall and F1 scores for the best-performing classifiers (GP-ICM, CRF, *HP Grad.* and *HP Approx.*) on the PHEME datasets. Cross-stance confusion rates denote how often the target stance (denoted by the row) is being misclassified as another stance (denoted by the column). Note stances are imbalanced, with commenting stance being predominant, as reported in Table 2.6.

Analysis of the Best Performing Methods Next, we analyze the results of four classifiers, including the three that have been our contribution in this thesis (GP-ICM, CRF, *HP Approx.* and *HP Grad.*) by looking at the per-class performance. Tables 7.2 and 7.3 report per-class precision, recall and F1 scores for the four PHEME datasets. They also report statistics on the cross-stance confusion rates across the four stances that the methods made. The cross-stance classifications are also depicted graphically for the Ottawa riots in Figure 7.4. We can observe that for all the dataset-class pairs, *HP Approx.* misclassifies almost all denying and questioning tweets, assigning most of them to the commenting class. At the same time, *HP Approx.* makes the least mistakes on the commenting stance, which is predominant in out datasets (as shown in Table 2.6, commenting stance is 3-4 times more frequent than the second most popular stance in each of the four used PHEME datasets). Precision on the supporting stance is the highest for *HP Approx.*, however recall and F1-scores are higher for *HP Grad.* on this stance. Thus, we can see that *HP Approx.* makes the smallest number of mistakes (as shown in Table 7.1) due to focusing on the majority class, being the commenting stance. Interestingly, GP-ICM performs best on the denying category (which we found also in our experiments in Chapter 4), whereas *HP Grad.* performs best on the questioning stance. Moreover, both GP-ICM and CRF make fewer mistakes on the commenting stance than *HP Grad.*, which presumably is the cost for *HP Grad.* performing well on supporting and questioning stances. Notice how the high stance imbalance poses a challenge to all approaches. In particular, the two under-represented stances (denying and questioning) are in most cases misclassified as commenting stance by the four methods (GP-ICM, CRF, *HP Approx.* and *HP Grad.*) across all datasets. This shows the need for further work on making better predictions on these minority stances.

Sensitivity analysis In Figure 7.5 we report results from Hawkes process based models with different values of ω controlling how historical tweets influence the intensity function value at a particular point of time, as shown in formula 7.2. Notice how *HP Approx.* is unaffected by varying values of the hyperparameter, showing a slight change in performance for the larger hyperparameter values. *HP Approx.* consistently performs well on the commenting stance at the cost of its performance on the underrepresented stances. Performance of *HP Grad.* varies for different values of ω, however also stays at the same range. We can see that there is an increase in performance for $\omega = 0.01$, and then drop for $\omega = 10$, therefore it is important to set the hyperparameter value to an appropriate range. Recall that $\omega = 0.1$ was used in all previous experiments, which represents a reasonable default.



(a) Macro-F1



(b) Micro-F1

Figure 7.5: Macro-F1 (top) and Micro-F1 (bottom) scores for HP methods for different hyperparameter values ω on the Ferguson dataset. Notice how *HP Approx.* is mostly insensitive to ω values. *HP Grad.* is affected by ω values to a larger extent.

7.5 Conclusions

In this chapter we proposed a novel model based on Hawkes processes for sequence classification of stances in Twitter which takes into account temporal information in addition to text. Using Twitter datasets and experimenting on rumour stance classification of tweets, we have shown that HP is a competitive approach, which at the time of being published as (Lukasik et al., 2016b) outperformed a range of strong benchmark methods in terms of Micro-F1 score by augmenting the class-conditional model with an informative prior based on temporal dynamics. Our experiments posit the importance of making use of temporal information available in tweets, which along with the textual content provide valuable information for the model to perform well on the task.

Chapter 8

Conclusions

In this thesis we considered two applications of modeling rumours in social media. The first application is rumour stance classification, where tweets around a rumour are classified regarding their stances towards rumour veracity. We refined the experimental settings of this problem and introduced models which incorporate features ignored in previous work. The second application we considered is rumour popularity prediction, which we defined as predicting the number of tweets in time intervals where a rumour has not been observed. We motivated this definition and used point processes for modeling rumour popularity. We showed how different features can be used within the point process framework, yielding improvements over the baselines. Below, we revisit the research questions we posed at the beginning of the thesis regarding the two applications, and review the contributions. Next, we provide ideas for future research.

8.1 Contributions

We review the contributions of this thesis by revisiting the research questions we listed in the Introduction chapter.

1. The first aim of this thesis was **predicting stance of tweets regarding rumours**.

   (a) **What would be a realistic and fair evaluation framework for rumour stance classification?**

   We concluded that evaluation from previous work was unsatisfactory, as it ignored important aspects of the task. Cross-validation was being conducted over tweets coming from different times and rumours (Qazvinian et al., 2011), which

did not correspond to how a journalist would require the system to work. Instead, we proposed a more realistic framework, where we classify tweets from a target rumour, in either the Leave One Out fashion (with no annotation of tweets from the target rumour) or the Leave Part Out fashion (where annotation of several initial tweets from the target rumour is available). In this way we do not allow for a situation where training tweets from a rumour occurred later than the test tweets, a situation which makes the evaluation unrealistic. We also considered both evaluation settings across rumour datasets, where at each iteration a separate rumour *dataset* is left out, as opposed to leaving a separate rumour within the same dataset. This evaluation proved to be even more challenging.

Moreover, we incorporated the questioning and the commenting labels into the problem, which were previously ignored, but are important for getting a better picture of a collective stance of a rumour.

(b) **Can information about stances of tweets from one rumour be useful for predicting stances of tweets from another rumour?**

We showed that the key to obtaining good results is using reference rumours, i.e. other rumour annotations which are available during training. We showed how a choice of multi-task learning kernels within the Gaussian process framework (GP-ICM) yields improvements compared to the GPs with a single-task kernel. We also demonstrated how GP-ICM was competitive to other approaches.

(c) **Do tweet arrival times carry complementary information to text for the stance prediction task?**

In this work, we introduced a continuous timestamp as a new feature for the rumour stance classification. We achieved this using a point process framework by modeling arrivals of tweets coming from different categories over time. We viewed the problem in a sequence classification setting, allowing us to use information about neighbouring tweets' labels at prediction time. Leveraging information from both textual and temporal characteristics, allowed the point process to outperform strong baselines.

2. The second aim was **predicting rumour popularity**.

(a) **How can the rumour popularity prediction problem be formulated?**

Appreciating how difficult and subjective it is to define what a popular rumour is, we decided to avoid defining a popular rumour and instead we defined the problem in a way that allows a journalist to decide for herself whether the future

rumour trajectory corresponds to an important rumour.[1] In particular, we define the rumour popularity prediction problem as that of predicting counts of tweets in future time intervals. We also consider an even more fine grained version, where future tweet arrival times from a rumour are predicted. We frame this problem in a supervised learning setting, where a number of initial tweets from the target rumour are observed, as well as tweets from historical rumours. The motivation for this task stems from the fact that capturing rumour popularity early on during their propagation would be of great help for journalists and officials, who need to deal with numerous rumours circulating in social media.

We motivated point processes for the task, a probabilistic framework for modeling events (tweets) over continuous space (time). We used Gaussian processes (GP) within a log-Gaussian Cox Process (LGCP) framework, which enjoys the benefits of flexible kernel specification and efficient hyperparameter optimization.

(b) **Can information about tweet arrivals from one rumour bring useful information for predicting the popularity of another rumour?**

We showed that when making predictions about a rumour, data from other rumours brings useful information. In particular, modeling multiple rumours via multi-task learning kernels led to significantly better results than those obtained by baselines.

(c) **Does the text content of tweets convey useful information for the rumour popularity prediction task?**

Temporal dynamics of how a rumour spreads is one feature we used to predict how a rumour propagates during intervals of time the rumour was not observed. A different source of information about a rumour is text. Users express their opinions about a rumour, providing a signal we hypothesised might be useful for rumour propagation. In the end, we observed that kernels which employ text yielded significantly better results than those which do not. This shows that text conveys important information about how a rumour would spread (or has spread during unobserved periods of time).

(d) **Does information about how text usage in tweets changes over time convey useful information for the rumour popularity prediction task?**

In order to capture text change over time, we introduced a convolution kernel for

[1] We also consider a classification setting of the rumour popularity prediction problem in Section 6.6 to demonstrate applicability of our convolution kernels to classification problems.

comparing sequences of posts over continuous time. The kernel convolves posts from the two time series, comparing pairs of tweets in terms of both their textual content and the timestamps of their occurrences. We observed a significant improvement when using this kernel compared to other kernels which do not use as much information from the time series. This demonstrates that change of tweet content over time conveys useful information for rumour popularity prediction.

8.2 Future Work

Below we explore avenues for future research.

Rumour stance classification Findings from previous work, such as Castillo et al. (2013) and Procter et al. (2013b), suggested that the aggregate stance of individual tweets correlates with actual rumour veracity. Previous work experimented with employing the stance expressed in the reactions of individual tweets for predicting the actual veracity of the rumour in question (Liu et al., 2015). This raises the question whether veracity classification improves when rumour stance classification is conducted with the methods introduced in this thesis.

In our rumour stance classification experiments in Chapter 4 we found Gaussian processes to be a competitive approach. However, applying Gaussian processes to large datasets may be challenging due to its time complexity. Gaussian processes require inverting a kernel matrix K of size $N \times N$ (where N is the number of tweets), which takes $O(N^3)$ time, a prohibitive operation for large N. Nevertheless, there exist approaches for scaling up Gaussian processes. One approach, called the inducing points technique, is based on choosing a subset of training examples and using them as a proxy for what is learnt by the model (Titsias, 2009). It could be used for running experiments on larger rumour datasets.

In our work we employed Brown clusters for representing text from tweets. It is worth exploring alternative text representation approaches, e.g. using word embeddings in vector space (Mikolov et al., 2013).

Previous work indicated that non-linear kernels outperform linear kernels on a range of text regression tasks (Beck, 2017). It may be useful to explore non-linear kernels for the rumour stance classification task, moving beyond the linear kernel.

Some future work has already been done on the rumour stance classification task. In collaborative work with the author, Zubiaga et al. (2016a) incorporated the Twitter conversation thread information within a Conditional Random Field (CRF) framework, and

demonstrated on the PHEME rumour datasets how the model outperforms baselines. The task keeps attracting attention from the community, e.g. through the recently organized SemEval 2017 task on rumour stance classification (Derczynski et al., 2017).

Modeling rumour dynamics In our experiments we employed the temporal dynamics of tweet arrivals, text content of tweets and rumour ids. This set of features can be extended to also incorporate network information. Users participate in the rumour discussion to different degrees, and so modeling what users participate in a discussion about a rumour might lead to more accurate predictions about future rumour popularity. For example, finding that an active user participates in a discussion about a rumour might lead to a prediction about them tweeting later on about the same rumour.

In our experiments, we considered the initial two hours of a rumour lifespan, using the first one hour of tweets for making predictions about the distribution of tweets in the second hour. However, different rumours may spread over different intervals of time, with some spanning over hours, and others spanning over days or weeks. An approach which would account for that by making use of all observed tweets from different rumours would be desirable. The challenge here is designing an appropriate approach to compare timestamps across different rumours.

In our work we employed Laplace approximation to deal with the inference with the log-Gaussian Cox process. It is possible to apply a fully Bayesian inference via Markov Chain Monte Carlo methods, which could help to better model the rumour dynamics through avoiding the caveats of approximations (Adams et al., 2009)

Our method is generalizable to problems other than modeling rumour popularity. One potential application could be advertisement campaigns. Before launching a campaign one could try to predict how successful it is going to be, and modify it accordingly. Before the application of our models, the characteristics of advertisement campaigns would need to be investigated (e.g., Does text change indicate how popular a campaign is going to be? Are there common patterns in how advertisement campaigns propagate?)

An interesting direction towards extending the work could be to try other multi-task learning methods. In particular, we assumed a single lengthscale across all rumours during training. One could consider modeling different rumours with different lengthscales.

In Chapter 7 we showed how rumour dynamics brings helpful information to the rumour stance classification task. Rumour dynamics can also be helpful for other applications. For rumour veracity classification, the arrivals of tweets over time can denote how likely a rumour is fake (i.e., some arrival patterns might correspond to Internet trolls posting). Moreover, rumour detection could also benefit from modeling rumour dynamics, as tweets

from a single rumour might correspond to a group of tweets happening in a relatively narrow interval of time (e.g. a tweet which happens long after a previous tweet from a rumour might likely be about a different topic).

Convolution kernels for comparing sequences of posts over time The developed convolution kernel can be applied to applications which can be framed as modeling outputs over the time series inputs. Direct modeling of time series can be viewed as a special case of such an approach, in which the response variable is the future value of the time series. However, discriminative modeling allows application to other tasks where an output variable is less closely related to the time series values. For example, we may be interested if a particular stream of posts in social media corresponds to a disaster event, or if variation of CO_2 across time in a given location indicates an alarming problem for the environment. In both instances the response variable is a classification output not directly related to the time series dynamics, however there are likely to be characteristics of the time series inputs which can be exploited in modeling the response variable. Thus, whenever one deals with time series discriminative problems and applies a kernelized approach, the introduced kernel can be used standalone or in conjunction with other kernels (summation, multiplication, tensor product are some of the operations preserving the Mercer's condition). Some of the promising problems that our approach might be evaluated on include event detection (e.g., is the set of Twitter posts a coherent event), event classification (e.g., is the given event a disaster?), clustering news outlets (represented by journal articles over time), or classifying blogs (represented by blog posts over time).

Sequence classification of text over continuous time The developed sequence classification algorithm can be applied to other problems than rumour stance classification. Whenever one deals with streaming text in continuous time, the algorithm might prove useful. Thus, our approach is applicable to other domains than social media, as in most corpora timestamps are assigned to text (e.g., as a time of creation). Examples where our approach might be evaluated are classification of posts in social media regarding hate speech (users might be influenced by what has been posted before), classification of tweets regarding named entities they refer to (as context of the conversation might be bringing useful information), classification of books regarding their genre (different genres might enjoy varying popularity over time), and classification of journal articles with respect to their reliability (different posting patterns may indicate reliability of the content).

Similar to Yang and Zha (2013), we fixed the decay parameter ω. It would be beneficial to explore ways for optimizing for ω. Optimization of the likelihood with respect to ω is

non-convex, which makes it a challenging task.

8.3 Final Remarks

In this thesis we investigated two rumour applications: (1) rumour stance classification, where tweets around a rumour are classified regarding their stance towards the rumour veracity, and (2) rumour popularity prediction, where counts of tweets from unobserved periods of time are predicted. We introduced realistic settings for the two applications, and investigated probabilistic models incorporating different sources of information: text, continuous timestamps, and out of domain rumours, all of which we found to improve predictions. We also introduced novel machine learning techniques, convolution kernels for streams of text over continuous time, and a point process based sequence classification algorithm. Both of the developed methods are applicable beyond Twitter data. Time is typically neglected, but, as we have shown, provides an important yet complex signal. Harnessing time effectively is an important challenge which could have widespread impact across text processing and related fields.

Appendices

Appendix A

Derivation of the Log Likelihood under the Hawkes Process Model

Recall that in Section 3.2.5 we described the Hawkes process model with the log-likelihood given by

$$\begin{aligned}\ell(\boldsymbol{t}, \boldsymbol{i}, \mathbf{m}, W|\boldsymbol{\beta}, \boldsymbol{\mu}, \boldsymbol{\gamma}, \boldsymbol{\alpha}, \omega) = &\sum_{n=1}^{N}\sum_{v=1}^{V} W_{nv}\log\beta_{m_n,v} + \sum_{n=1}^{N}\log\left(\lambda_{i_n,m_n}(t_n|\mathbf{H}_{t_n^-})\right) \\ &- \sum_{i=1}^{|U|}\sum_{m=1}^{|R|}\int_0^T \lambda_{i,m}(s|\mathbf{H}_{s^-})ds,\end{aligned} \tag{A.1}$$

and the intensity function given by

$$\lambda_{i_n,m_n}(t|\mathbf{H}_{t^-}) = \mu_{i_n}\gamma_{m_n} + \sum_{\ell=1}^{n-1}\mathbb{I}(m_\ell = m)\alpha_{i_\ell,i_n}\kappa(t-t_\ell). \tag{A.2}$$

Here we show how the log-likelihood from Equation (A.1) can be simplified by representing the integral term in a closed form.

$$\begin{aligned}\sum_{i=1}^{|U|}\sum_{m=1}^{|R|}\int_0^T \lambda_{i,m}(s|\mathbf{H}_{s^-})ds &= \sum_{i=1}^{|U|}\sum_{m=1}^{|R|}\int_0^T \left(\mu_i\gamma_m + \sum_{t_\ell<t}\mathbb{I}(m_\ell = m)\alpha_{\alpha_{i_\ell},i}\kappa(t-t_\ell)\right)dt \\ &= \underbrace{T\sum_{i=1}^{|U|}\sum_{m=1}^{|R|}\mu_i\gamma_m}_{Z} + \sum_{i=1}^{|U|}\sum_{m=1}^{|R|}\int_0^T\sum_{t_\ell<t}\mathbb{I}(m_\ell = m)\alpha_{\alpha_{i_\ell},i}\kappa(t-t_\ell)dt\end{aligned}$$

$$= Z + \sum_{i=1}^{|U|} \int_0^T \sum_{t_\ell < t} \alpha_{\alpha_{i_\ell,i}} \kappa(t - t_\ell) dt \sum_{m=1}^{|R|} \mathbb{I}(m_\ell = m)$$

$$= Z + \sum_{i=1}^{|U|} \int_0^T \sum_{t_\ell < t} \alpha_{\alpha_{i_\ell,i}} \kappa(t - t_\ell) dt,$$

because each tweet belongs to exactly one rumour,

$$= Z + \sum_{i=1}^{|U|} \sum_{n=1}^{N+1} \int_{t_{n-1}}^{t_n} \sum_{t_\ell < t} \alpha_{\alpha_{i_\ell,i}} \kappa(t - t_\ell) dt,$$

where $t_0 = 0$ and $t_{N+1} = T$,

$$= Z + \sum_{i=1}^{|U|} \sum_{n=1}^{N+1} \sum_{\ell=1}^{n-1} \alpha_{\alpha_{i_\ell,i}} \int_{t_{n-1}}^{t_n} \kappa(t - t_\ell) dt,$$

because for timestamps from each interval there is a fixed number of tweets that occurred before,

$$= Z + \sum_{i=1}^{|U|} \sum_{n=1}^{N+1} \sum_{\ell=1}^{n-1} \alpha_{\alpha_{i_\ell,i}} \left(L(t_n - t_\ell) - L(t_{n-1} - t_\ell) \right)$$

$$= Z + \sum_{i=1}^{|U|} \sum_{\ell=1}^{N} \sum_{n=l+1}^{N+1} \alpha_{\alpha_{i_\ell,i}} \left(L(t_n - t_\ell) - L(t_{n-1} - t_\ell) \right)$$

$$= Z + \sum_{i=1}^{|U|} \sum_{\ell=1}^{N} \alpha_{\alpha_{i_\ell,i}} \sum_{n=l+1}^{N+1} \left(L(t_n - t_\ell) - L(t_{n-1} - t_\ell) \right)$$

$$= Z + \sum_{i=1}^{|U|} \sum_{\ell=1}^{N} \alpha_{\alpha_{i_\ell,i}} \left(L(T - t_\ell) - L(t_{l+1-1} - t_\ell) \right)$$

$$= Z + \sum_{i=1}^{|U|} \sum_{\ell=1}^{N} \alpha_{\alpha_{i_\ell,i}} \left(L(T - t_\ell) - L(0) \right)$$

$$= Z + \sum_{i=1}^{|U|} \sum_{\ell=1}^{N} \alpha_{\alpha_{i_\ell,i}} K(T - t_\ell)$$

$$= T \sum_{i=1}^{|U|} \sum_{m=1}^{|R|} \mu_i \gamma_m + \sum_{i=1}^{|U|} \sum_{\ell=1}^{N} \alpha_{\alpha_{i_\ell,i}} K(T - t_\ell), \tag{A.3}$$

where:

$$L(x) = \int \kappa(x) dx = -\exp(-\omega x), \tag{A.4}$$

and

$$\begin{aligned} K(T - t_\ell) &= L(T - t_\ell) - L(0) \\ &= -\exp\left(-\omega(T - t_\ell)\right) - -\exp\left(-\omega(0)\right) \\ &= 1 - \exp\left(-\omega(T - t_\ell)\right). \end{aligned} \tag{A.5}$$

Plugging the result from Equation (A.3) into Equation (A.1) we obtain the following form for the log likelihood under the Hawkes process model:

$$\begin{aligned} \ell(\boldsymbol{t}, \boldsymbol{i}, \mathbf{m}, W | \boldsymbol{\beta}, \boldsymbol{\mu}, \boldsymbol{\gamma}, \boldsymbol{\alpha}, \omega) = &\sum_{n=1}^{N} \sum_{v=1}^{V} W_{nv} \log \beta_{m_n, v} \\ &+ \sum_{n=1}^{N} \log\left(\lambda_{i_n, m_n}(t_n)\right) \\ &- T \sum_{i=1}^{|U|} \sum_{m=1}^{|R|} \mu_i \gamma_m - \sum_{i=1}^{|U|} \sum_{\ell=1}^{N} \alpha_{\alpha_{i_\ell, i}} K(T - t_\ell) \end{aligned} \tag{A.6}$$

We use this result in Chapter 7 when developing the sequence classification methods based on Hawkes processes.

Bibliography

Adams, R. P., Murray, I., and MacKay, D. J. C. (2009). Tractable nonparametric Bayesian inference in Poisson processes with Gaussian process intensities. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 9–16, New York, NY, USA. ACM.

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.

Allcott, H. and Gentzkow, M. (2017). Social media and fake news in the 2016 election. *Journal of Economic Perspectives*, 31(2):211–36.

Allport, G. W. and Postman, L. (1947). The psychology of rumor. *Journal of Clinical Psychology*.

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.

Ando, S. and Suzuki, E. (2014). Discriminative learning on exemplary patterns of sequential numerical data. In *ICDM*, pages 1–10.

Augenstein, I., Rocktäschel, T., Vlachos, A., and Bontcheva, K. (2016). Stance detection with bidirectional conditional encoding. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 876–885, Austin, Texas. Association for Computational Linguistics.

Bar-Joseph, Z., Gitter, A., and Simon, I. (2012). Studying and modelling dynamic biological processes using time-series gene expression data. *Nature Reviews Genetics*, 13(8):552–564.

Beck, D. (2017). *Gaussian Processes for Text Regression*. PhD thesis, Department of Computer Science, The University of Sheffield.

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.

Bergstra, J. and Bengio, Y. (2012). Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13:281–305.

Bhaskaran, K., Gasparrini, A., Hajat, S., Smeeth, L., and Armstrong, B. (2013). Time series regression studies in environmental epidemiology. *International Journal of Epidemiology*.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Bitvai, Z. and Cohn, T. (2015a). Day trading profit maximization with multi-task learning and technical analysis. *Machine Learning*, 101(1-3):187–209.

Bitvai, Z. and Cohn, T. (2015b). Predicting peer-to-peer loan rates using Bayesian non-linear regression. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence*, pages 2203–2209.

Bonilla, E., Chai, K. M., and Williams, C. (2008). Multi-task Gaussian process prediction. In Platt, J., Koller, D., Singer, Y., and Roweis, S., editors, *Advances in Neural Information Processing Systems 20*, pages 153–160. MIT Press, Cambridge, MA.

Boyd, S. and Vandenberghe, L. (2004). *Convex Optimization*. Cambridge University Press, New York, NY, USA.

Brix, A. and Diggle, P. J. (2001). Spatiotemporal prediction for log-Gaussian Cox processes. *Journal of the Royal Statistical Society Series B*, 63(4):823–841.

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*, 18:467–479.

Castillo, C., Mendoza, M., and Poblete, B. (2011). Information credibility on Twitter. In *Proceedings of the 20th International Conference on World Wide Web*, WWW '11, pages 675–684.

Castillo, C., Mendoza, M., and Poblete, B. (2013). Predicting information credibility in time-sensitive social media. *Internet Research*, 23(5):560–588.

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.

Collins, M. and Duffy, N. (2001). Convolution kernels for natural language. In *Advances in Neural Information Processing Systems 14*, pages 625–632. MIT Press.

De, A., Valera, I., Ganguly, N., Bhattacharya, S., and Gomez-Rodriguez, M. (2015). Modeling opinion dynamics in diffusion networks. *CoRR*, abs/1506.05474.

Derczynski, L., Bontcheva, K., Liakata, M., Procter, R., Wong Sak Hoi, G., and Zubiaga, A. (2017). SemEval-2017 task 8: RumourEval: determining rumour veracity and support for rumours. *ArXiv e-prints*.

Derczynski, L., Bontcheva, K., Lukasik, M., Declerck, T., Scharl, A., Georgiev, G., Osenova, P., Lobo, T. P., Kolliakou, A., Stewart, R., et al. (2015). Pheme: computing veracitythe fourth challenge of big social data.

DiFonzo, N. and Bordia, P. (2007). Rumor, gossip and urban legends. *Diogenes*, 54(1):19–35.

Donovan, P. (2007). How idle is idle talk? one hundred years of rumor research. *Diogenes*, 54(1):59–82.

Du, N., Farajtabar, M., Ahmed, A., Smola, A. J., and Song, L. (2015). Dirichlet-Hawkes processes with applications to clustering continuous-time document streams. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '15, pages 219–228, New York, NY, USA. ACM.

Dzogang, F., Lansdall-Welfare, T., and Cristianini, N. (2016). Seasonal fluctuations in collective mood revealed by Wikipedia searches and Twitter posts. In *IEEE International Conference on Data Mining Workshops, ICDM Workshops 2016, December 12-15, 2016, Barcelona, Spain.*, pages 931–937.

Evgeniou, T. and Pontil, M. (2004). Regularized multi–task learning. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '04, pages 109–117.

Ferreira, W. and Vlachos, A. (2016). Emergent: a novel data-set for stance classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association*

for Computational Linguistics: Human Language Technologies, pages 1163–1168, San Diego, California. Association for Computational Linguistics.

Fornaciari, T., Celli, F., and Poesio, M. (2013). The effect of personality type on deceptive communication style. In *Intelligence and Security Informatics Conference (EISIC), 2013 European*, pages 1–6.

Friggeri, A., Adamic, L., Eckles, D., and Cheng, J. (2014). Rumor cascades. In *International AAAI Conference on Weblogs and Social Media*.

Gelman, A., Carlin, J. B., Stern, H. S., and Rubin, D. B. (2003). *Bayesian Data Analysis*. Chapman and Hall/CRC.

Gomez Rodriguez, M., Leskovec, J., and Krause, A. (2010). Inferring networks of diffusion and influence. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '10, pages 1019–1028, New York, NY, USA. ACM.

Gomez-Rodriguez, M. and Schölkopf, B. (2012). Submodular inference of diffusion networks from multiple trees. In *Proceedings of the 29th International Conference on Machine Learning, ICML 2012, Edinburgh, Scotland, UK, June 26 - July 1, 2012*.

Goode, L. (2009). Social news, citizen journalism and democracy. *New Media & Society*, 11(8):1287–1305.

Gorrell, G. and Bontcheva, K. (2016). Classifying Twitter favorites: like, bookmark, or thanks? *Journal of the Association for Information Science and Technology*, 67(1):17–25.

Guerin, B. and Miyazaki, Y. (2006). Analyzing rumors, gossip, and urban legends through their conversational properties. *The Psychological Record: Vol. 56: Iss. 1, Article 2.*

Guo, W. and Diab, M. (2013). Improving lexical semantics for sentential semantics: modeling selectional preference and similar words in a latent variable model. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 739–745, Atlanta, Georgia. Association for Computational Linguistics.

Gupta, S., Sakamoto, K., and Ortony, A. (2013). Telling it like it isnt: A comprehensive approach to analyzing verbal deception. In *F. Paglieri, L. Tummolini, R. Falcone & M.*

Miceli (Eds.), The goals of cognition: Festschfit for Cristiano Castelfranchi. London, College Publications.

Gnen, M. and Alpaydin, E. (2011). Multiple kernel learning algorithms. *Journal of Machine Learning Research*, 12:2211–2268.

Hamdi, S., Gancarski, A. L., Bouzeghoub, A., and Yahia, S. B. (2016). Tison: trust inference in trust-oriented social networks. *ACM Transactions on Information Systems (TOIS)*, 34(3):17.

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

Hamidian, S. and Diab, M. T. (2016). Rumor identification and belief investigation on Twitter. In *Proceedings of the 7th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis, WASSA@NAACL-HLT 2016, June 16, 2016, San Diego, California, USA*, pages 3–8.

Hasan, K. S. and Ng, V. (2013). Stance classification of ideological debates: data, models, features, and constraints. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pages 1348–1356.

Hassan, N., Li, C., and Tremayne, M. (2015). Detecting check-worthy factual claims in presidential debates. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, CIKM '15, pages 1835–1838, New York, NY, USA. ACM.

Hassan, N., Sultana, A., Wu, Y., Zhang, G., Li, C., Yang, J., and Yu, C. (2014). Data in, fact out: Automated monitoring of facts by factwatcher. *Proc. VLDB Endow.*, 7(13):1557–1560.

Haussler, D. (1999). Convolution kernels on discrete structures. Technical Report UCS-CRL-99-10, University of California at Santa Cruz, Santa Cruz, CA, USA.

Hawkes, A. G. (1971). Spectra of some self-exciting and mutually exciting point processes. *Biometrika*, 58(1):83–90.

Hawkes, A. G. and Oakes, D. (1974). A cluster process representation of a self-exciting process. *Journal of Applied Probability*, 11(3):493–503.

He, X., Rekatsinas, T., Foulds, J., Getoor, L., and Liu, Y. (2015). HawkesTopic: a joint model for network inference and topic modeling from text-based cascades. *ICML*.

Jia, S., Lansdall-Welfare, T., and Cristianini, N. (2016). Gender classification by deep learning on millions of weakly labelled images. In *IEEE International Conference on Data Mining Workshops, ICDM Workshops 2016, December 12-15, 2016, Barcelona, Spain.*, pages 462–467.

Kadous, M. W. and Sammut, C. (2005). Classification of multivariate time series and structured data using constructive induction. *Mach. Learn.*, 58(2-3):179–216.

Karimi, M. R., Tavakoli, E., Farajtabar, M., Song, L., and Gomez-Rodriguez, M. (2016). Smart broadcasting: do you want to be seen? In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, pages 1635–1644.

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

Kempe, D., Kleinberg, J., and Tardos, E. (2003). Maximizing the spread of influence through a social network. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '03, pages 137–146, New York, NY, USA. ACM.

Keogh, E. J. and Pazzani, M. J. (2000). Scaling up dynamic time warping for datamining applications. In *Proc. of 6th KDD*, pages 285–289.

Kobayashi, R. and Lambiotte, R. (2016). TiDeH: time-dependent Hawkes process for predicting retweet dynamics. *CoRR*, abs/1603.09449.

Kwak, H., Lee, C., Park, H., and Moon, S. (2010). What is Twitter, a social network or a news media? In *Proceedings of the 19th International Conference on World Wide Web*, WWW '10, pages 591–600, New York, NY, USA. ACM.

Lafferty, J. D., McCallum, A., and Pereira, F. C. N. (2001). Conditional Random Fields: probabilistic models for segmenting and labeling sequence data. In *Proc. of International Conference on Machine Learning (ICML)*, pages 282–289.

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.

Leskovec, J., Backstrom, L., and Kleinberg, J. (2009). Meme-tracking and the dynamics of the news cycle. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '09, pages 497–506, New York, NY, USA. ACM.

Leskovec, J., Krause, A., Guestrin, C., Faloutsos, C., VanBriesen, J., and Glance, N. (2007). Cost-effective outbreak detection in networks. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '07, pages 420–429, New York, NY, USA. ACM.

Lewandowsky, S., Ecker, U. K., Seifert, C. M., Schwarz, N., and Cook, J. (2012). Misinformation and its correction continued influence and successful debiasing. *Psychological Science in the Public Interest*, 13(3):106–131.

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

Linderman, S. W. and Adams, R. P. (2014). Discovering latent network structure in point process data. In *Proceedings of the 31st International Conference on Machine Learning, Cycle 2*, volume 32 of *JMLR Proceedings*, pages 1413–1421. JMLR.org.

Liu, X., Nourbakhsh, A., Li, Q., Fang, R., and Shah, S. (2015). Real-time rumor debunking on Twitter. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, CIKM '15, pages 1867–1870, New York, NY, USA. ACM.

Lodhi, H., Saunders, C., Shawe-Taylor, J., Cristianini, N., and Watkins, C. (2002). Text classification using string kernels. *J. Mach. Learn. Res.*, 2:419–444.

Lukasik, M., Bontcheva, K., Cohn, T., Zubiaga, A., Liakata, M., and Procter, R. (2016a). Using Gaussian processes for rumour stance classification in social media. *CoRR*, abs/1609.01962.

Lukasik, M. and Cohn, T. (2016). Convolution kernels for discriminative learning from streaming text. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence*.

Lukasik, M., Cohn, T., and Bontcheva, K. (2015a). Classifying tweet level judgements of rumours in social media. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Lukasik, M., Cohn, T., and Bontcheva, K. (2015b). Point process modelling of rumour dynamics in social media. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (ACL-IJCNLP)*, pages 518–523.

Lukasik, M., Srijith, P. K., Cohn, T., and Bontcheva, K. (2015c). Modeling tweet arrival times using log-Gaussian Cox processes. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 250–255.

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

Ma, J., Gao, W., Wei, Z., Lu, Y., and Wong, K.-F. (2015). Detect rumors using time series of social context information on microblogging websites. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, CIKM '15, pages 1751–1754, New York, NY, USA. ACM.

Manning, C. D., Raghavan, P., and Schütze, H. (2008). *Introduction to Information Retrieval*. Cambridge University Press.

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.

Middleton, S. E. and Krivcovs, V. (2016). Geoparsing and geosemantics for social media: spatio-temporal grounding of content propagating rumours to support trust and veracity analysis during breaking news. *ACM Transactions on Information Systems*, 34(3):1–27.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space. *CoRR*, abs/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., Kiritchenko, S., Sobhani, P., Zhu, X., and Cherry, C. (2016). Semeval-2016 task 6: detecting stance in tweets. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 31–41, San Diego, California. Association for Computational Linguistics.

Møller, J. and Syversveen, A. R. (1998). Log Gaussian Cox processes. *Scandinavian Journal of Statistics*, pages 451–482.

Nel, F., Lesot, M.-J., Capet, P., and Delavallade, T. (2010). Rumour detection in information warfare: Understanding publishing behaviours as a prerequisite. In *NATO RTO-MP-IST-091 Information Assurance and Cyber Defence*.

Nickisch, H. and Rasmussen, C. E. (2008). Approximations for binary Gaussian process classification. *Journal of Machine Learning Research*, 9:2035–2078.

Norman, W. T. (1963). Toward an adequate taxonomy of personality attributes: replicated factors structure in peer nomination personality ratings. *Journal of abnormal and social psychology*, 66:574–583.

Norvig, P. (1992). *Paradigms of Artificial Intelligence Programming: case Studies in Common LISP*. Artificial intelligence programming languages. Morgan Kaufman Publishers.

Ogata, Y. (2006). On lewis' simulation method for point processes. *IEEE Trans. Inf. Theor.*, 27(1):23–31.

Okazaki, N. (2007). Crfsuite: a fast implementation of conditional random fields (crfs).

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.

Papadopoulos, S., Bontcheva, K., Jaho, E., Lupu, M., and Castillo, C. (2016). Overview of the special issue on trust and veracity of information in social media. *ACM Transactions on Information Systems (TOIS)*, 34(3):14.

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.

Petrovic, S. (2012). *Real-time event detection in massive streams*. PhD thesis, The University of Edinburgh.

Preotiuc-Pietro, D. (2014). *Temporal models of streaming social media data*. PhD thesis, Department of Computer Science, The University of Sheffield.

Preotiuc-Pietro, D. and Cohn, T. (2013). A temporal model of text periodicities using Gaussian processes. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '13, pages 977–988.

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 (ACL-IJCNLP)*, pages 1754–1764.

Procter, R., Crump, J., Karstedt, S., Voss, A., and Cantijoch, M. (2013a). Reading the riots: what were the police doing on Twitter? *Policing and society*, 23(4):413–436.

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

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.

Rabiner, L. R. (1989). A tutorial on hidden markov models and selected applications in speech recognition. In *Proc. of the IEEE*, pages 257–286.

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

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

Ritter, A., Clark, S., Mausam, and Etzioni, O. (2011). Named entity recognition in tweets: an experimental study. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 1524–1534, Stroudsburg, PA, USA. Association for Computational Linguistics.

Rogers, S. and Girolami, M. (2012). *A First Course in Machine Learning*. CRC Press.

Rosnow, R. L. (1991). The psychology of rumor. *American Psychologist, Vol 46(5)*, pages 484–496.

Shah, K., Cohn, T., and Specia, L. (2013). An investigation on the effectiveness of features for translation quality estimation. In *Machine Translation Summit XIV*, pages 167–174, Nice, France.

Shahriari, B., Swersky, K., Wang, Z., Adams, R., and de Freitas, N. (2015). Taking the human out of the loop : a review of bayesian optimization. Technical report, Universities of Harvard, Oxford, Toronto, and Google DeepMind.

Shibutani, T. (1969). Improvised news: A sociological study of rumor. *Social Research*, 36(1):169–171.

Shumway, R. H. and Stoffer, D. S. (2006). *Time Series Analysis and Its Applications: with R Examples (Springer Texts in Statistics)*. Springer, 2nd edition.

Simma, A. and Jordan, M. I. (2010). Modeling events with cascades of poisson processes. In *UAI*, pages 546–555.

Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C. D., Ng, A. Y., and Potts, C. (2013). Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Stroudsburg, PA. Association for Computational Linguistics.

Song, G., Ye, Y., Du, X., Huang, X., and Bie, S. (2014). Short text classification: a survey. *Journal of Multimedia*, 9(5).

Sridhar, D., Getoor, L., and Walker, M. (2014). Collective stance classification of posts in online debate forums. In *ACL Joint Workshop on Social Dynamics and Personal Attributes in Social Media*.

Srijith, P. K., Lukasik, M., Bontcheva, K., and Cohn, T. (2017). Longitudinal modeling of social media with Hawkes process based on users and networks. In *Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*.

Sutton, C. and McCallum, A. (2012). An introduction to conditional random fields. *Found. Trends Mach. Learn.*, 4(4):267–373.

Tausczik, Y. R. and Pennebaker, J. W. (2010). The psychological meaning of words: LIWC and computerized text analysis methods. *Journal of Language and Social Psychology*, pages 0261927–09351676.

The GPy authors (2015). GPy: a Gaussian process framework in Python. `http://github.com/SheffieldML/GPy`.

Thorne, J. and Vlachos, A. (2017). An extensible framework for verification of numerical claims. In *Proceedings of the Software Demonstrations of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, pages 37–40. Association for Computational Linguistics.

Titsias, M. K. (2009). Variational learning of inducing variables in sparse Gaussian processes. In *In Artificial Intelligence and Statistics 12*, pages 567–574.

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

Vanhatalo, J., Riihimäki, J., Hartikainen, J., Jylänki, P., Tolvanen, V., and Vehtari, A. (2013). Gpstuff: Bayesian modeling with Gaussian processes. *J. Mach. Learn. Res.*, 14(1):1175–1179.

Vishwanathan, S. V. N., Schraudolph, N. N., Kondor, R., and Borgwardt, K. M. (2010). Graph kernels. *J. Mach. Learn. Res.*, 11:1201–1242.

Vlachos, A. and Riedel, S. (2014). Fact checking: task definition and dataset construction. In *Proceedings of the ACL 2014 Workshop on Language Technologies and Computational Social Science*, pages 18–22, Baltimore, MD, USA. Association for Computational Linguistics.

Vlachos, A. and Riedel, S. (2015). Identification and verification of simple claims about statistical properties. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2596–2601, Lisbon, Portugal. Association for Computational Linguistics.

Vosoughi, S. (2015). *Automatic Detection and Verification of Rumors on Twitter*. PhD thesis, Massachusetts Institute of Technology.

Wang, Y., Xie, B., Du, N., and Song, L. (2016). Isotonic Hawkes processes. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, pages 2226–2234. JMLR.org.

Wu, Y., Wu, G., and Chang, E. Y. (2005). Multi-view sequence-data representation and non-metric distance-function learning.

Xing, Z., Pei, J., and Keogh, E. (2010). A brief survey on sequence classification. *SIGKDD Explor. Newsl.*, 12(1):40–48.

Xing, Z., Pei, J., Yu, P. S., and Wang, K. (2011). Extracting interpretable features for early classification on time series. In *SDM*, pages 247–258.

Yang, S.-H. and Zha, H. (2013). Mixture of mutually exciting processes for viral diffusion. In *ICML (2)*, volume 28 of *JMLR Proceedings*, pages 1–9.

Ye, L. and Keogh, E. (2011). Time series shapelets: a novel technique that allows accurate, interpretable and fast classification. *Data Mining and Knowledge Discovery*, 22(1-2):149–182.

Zammit-Mangion, A., Dewar, M., Kadirkamanathan, V., and Sanguinetti, G. (2012). Point process modelling of the Afghan War Diary. *Proceedings of the National Academy of Sciences of the United States of America*, 109(31):12414–12419.

Zarezade, A., Upadhyay, U., Rabiee, H. R., and Gomez-Rodriguez, M. (2016). RedQueen: an online algorithm for smart broadcasting in social networks. *CoRR*, abs/1610.05773.

Zeng, L., Starbird, K., and Spiro, E. S. (2016a). Rumors at the speed of light? modeling the rate of rumor transmission during crisis. *2016 49th Hawaii International Conference on System Sciences (HICSS)*, 00:1969–1978.

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

Zhao, Q., Erdogdu, M. A., He, H. Y., Rajaraman, A., and Leskovec, J. (2015a). Seismic: a self-exciting point process model for predicting tweet popularity. *CoRR*, abs/1506.02594.

Zhao, Z., Resnick, P., and Mei, Q. (2015b). Early detection of rumors in social media from enquiry posts. In *International World Wide Web Conference Committee (IW3C2)*.

Zhou, D. X., Resnick, P., and Mei, Q. (2011). Classifying the political leaning of news articles and users from user votes. In *ICWSM*, pages 417–424.

Zhou, K., Zha, H., and Song, L. (2013a). Learning social infectivity in sparse low-rank networks using multi-dimensional Hawkes processes. In *AISTATS*, volume 31 of *JMLR Workshop and Conference Proceedings*, pages 641–649. JMLR.org.

Zhou, K., Zha, H., and Song, L. (2013b). Learning triggering kernels for multi-dimensional Hawkes processes. In *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*, ICML'13, pages III–1301–III–1309. JMLR.org.

Zubiaga, A., Aker, A., Bontcheva, K., Liakata, M., and Procter, R. (2017). Detection and resolution of rumours in social media: a survey. *ArXiv e-prints*.

Zubiaga, A., Kochkina, E., Liakata, M., Procter, R., and Lukasik, M. (2016a). Stance classification in rumours as a sequential task exploiting the tree structure of social media conversations. In *Proceedings of the 26th International Conference on Computational Linguistics (COLING)*, pages 2438–2448.

Zubiaga, A., Liakata, M., and Procter, R. (2016b). Learning reporting dynamics during breaking news for rumour detection in social media. *CoRR*, abs/1610.07363.

Zubiaga, A., Liakata, M., Procter, R., Bontcheva, K., and Tolmie, P. (2015a). Towards detecting rumours in social media. In *AAAI Workshop on AI for Cities*.

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

Zubiaga, A., Tolmie, P., Liakata, M., and Procter, R. (2015b). D2.4 qualitative analysis of rumours, sources, and diffusers across media and languages. Technical report, University of Warwick.