Master Thesis

The predictive power of tweets
an exploratory study

Hannah Tops
3279421

October 2013

Supervisors:
Alexis Dimitriadis (Universiteit Utrecht)
Antal van den Bosch (Radboud Universiteit Nijmegen)

Utrecht University
Faculty of Humanities
Cognitive Artificial Intelligence

Radboud Universiteit Nijmegen
Faculty of Arts
Communication and Information Sciences

Abstract

We describe a system that estimates when an event is going to happen from a stream of microtexts on Twitter referring to that event. Using a Twitter archive of 60 known football events, this problem is transferred into a classification problem. Different training procedures were followed, such as varying the training data and hierarchical classification. The best performing method was on average 52.3 hours off, and especially the tweets that referred to an event that was still far away appeared to be hard to predict. Comparing the performance of the system to the performance of humans on the same task, it appeared that there is room for improvement for the system.

Acknowledgments

In Utrecht as well as in Nijmegen some people have played an important role in the process of writing my thesis. First of all I would like to thank my supervisors, Alexis Dimitriadis and Antal van den Bosch. Alexis always made me think of why I was doing things the way I did, and, seeing things from a different perspective, he came up with some fresh ideas. With Antal I had nice discussions about the steps that I had taken or about further steps I had to take. He also made me wiser about subjects outside the scope of my thesis. Second, my roommates in Nijmegen. Florian Kunneman was always very helpful to me, and he always made time to answer my questions. Ali Hürriyetoglu was funny to work with, especially on rainy days. Third I would like to thank Sophia Katrenko for helping me finding a subject for my thesis, and pointing me to the work done in Nijmegen. I also would like to thank how helped me with my experiment.

Last but not least I would like to thank my friends and family for giving me great support and some necessary distraction from thesis-writing. Special thanks to Sabine Oudt, Otto Rottier and my parents, who looked over my thesis, advised me and created a perfect study climate for me, especially in the last weeks.

Contents

Chapter 1

Introduction

Text classification studies are gaining in importance recently because of the availability of an increasing number of electronic documents of a variety of sources. Analyzing these large data sets can lead to new insights that otherwise will not be discovered [18]. In this thesis we focus on one of those sources of electronic documents, namely Twitter. Twitter[1] is a microblogging service for posting messages, called tweets, with a maximum length of 140 characters. Like other social media, it has acquired immense popularity in recent years. Users of Twitter often communicate what they did in the past, are doing at the moment, or will be doing in the future. In other words, they often refer to an event. In this research anticipatory tweets that refer to a *future event* will be studied, in order to estimate the point in time at which the referred to event will take place. Such estimates, in combination with some aggregation or ranking, could be used to generate a forecast of future events. There are several reasons why we should want to do this. Firstly, a system like this could assist journalists, the police, and the public in being prepared for events that are several days or weeks in the future. For journalists as well as for the news-reading public this is interesting, as they both wish to be on top of the news as it happens. Nowadays, journalists are often overtaken by the public reporting on events. The massive amount of short messages posted via Twitter provide a potentially valuable source of information for this task, outperforming newswire articles in terms of dynamics and pluralism. The same goes for the police; if they know of some irregularities in advance, they can be better prepared. Secondly, and this applies to predictions in general, automatic prediction with machines has a lower cost than with humans, they are objective and can process greater amounts of data.

[1] `http://twitter.com`

1.1 Research question

The primary goal of this thesis is to provide insight into what extent machine-learning classifiers can predict the time-to-event of a tweet, in relation to the event it refers to, based on the text in this tweet. In this study the problem of predicting the time-to-event is transformed into a classification problem, by splitting the data into discrete time periods. First, tweets are collected that contain events of which we know when they happened, i.e. football matches. Then we train a machine-learning classifier to map those events onto discrete time periods. Several methods will be tried, in order to investigate the best training regime to do the prediction. In order to evaluate the outcomes, we compare the performance of the system with the performance of humans on the same task.

The main research questions of this thesis will thus be the following: *How accurately can we predict the time-to-event of events mentioned in tweets? How do following different procedures influence the performance?*

1.2 Relevance to CAI

Cognitive Artificial Intelligence is an inter-disciplinary science leaning on, and integrating four disciplines: philosophy, psychology, language and computer science. This thesis integrates two of the four mentioned disciplines: language and computer science. We enable computers to derive meaning from natural language input, using statistically based algorithms, where rules are are automatically learned through the analysis of large corpora of typical real-world examples.

1.2.1 Position in the field of CAI

With the increasing availability of electronic documents from a variety of sources, the automatic processing of documents is gaining more importance and has become one of the key methods for the organization of information and discovery of knowledge [2]. One of those resources are social media, such as Twitter. Twitter aggregates opinions and feelings of diverse groups of people at low cost, and gives an opportunity to discover, for example, the characteristics of informal language or social structures [38]. Unique advantages of Twitter are the timeliness of tweets, their rich meta data (on user, time, and optionally location), their large quantity, they are multilingual, the fact that many people are using them and their information-rich communicative conventions such as hashtags. On the other hand, Twitter is highly fragmented into short, often noisy messages carrying a large amount of redundant information. Tweets also have the tendency to be self-contained, which means they usually only have a simple discourse and pragmatic

structure [24]. Besides that, the huge amounts of tweets can sometimes be hard to handle and bring a redundancy of information as well.

Previous work in event extraction has focused largely on news articles, as this genre of text once was the best source of information on current events. Twitter has become an important complementary source of such information, outperforming news articles in terms of dynamics and pluralism, and several studies noticed this. Ritter *et al.* [24] train on annotated open-domain event mentions in tweets in order to create a calendar of events based on explicitly dates and words typical of the event, such as named entities. While Ritter *et al.* focus on the *extraction* of events, the study by Weerkamp and De Rijke [34] focuses on the *prediction* of events that are likely to become popular in the very near future, such as 'watching a specific television programme' or 'going to bed'. Closest to the current study are the studies by Kunneman *et al.* [13] and Hurriyetoglu *et al.* [11]. The first study showed that tweets posted before event start time can accurately be distinguished from tweets during and after an event. However, a classification with a higher specificity than 'before' needs to be made in order to estimate the time-to-event. In the study by Hurriyetoglu *et al.* the time-to-event in hours was estimated using regression-based methods, resulting in a performance that was at best 43 hours off on average. In contrast, current thesis investigates the performance of predicting the time-to-event in hours using classification-based methods.

1.3 Outline

First, tweets about events that have already happened, and of which we know when, were collected. We chose to collect football matches, because they occur frequently, we know their dates and they generate a sufficient amount of data. For each tweet we identify its time of posting relative to the event and then train a machine-learning classifier to map unseen tweets onto discrete time periods. Each time period will be given a temporal value (e.g. -10, thus 10 hours before event) and this value will be assigned to each tweet that is classified in that period. In the end, for each tweet the number of hours that this classification is off will be calculated for evaluation. Creating the discrete time periods is thus a necessary step in order to do the classification, and many segmentations will be possible. Given the imbalanced distribution of tweets referring to an event over time (the large majority of tweets is posted right before, during or right after an event), the segmentation of time categories is not a trivial matter. Therefore, two different segmentation methods will be compared: left branching and k-means clustering, and for each segmentation method various ways to classify the data are investigated. In summary:

- We train with three different classifiers; Naive Bayes, k-nearest neighbors

and Support Vector Machines.

- We try two different feature selection methods: Information Gain and Chi Squared (χ^2).
- We vary the number of features and the number of instances used to train with.
- We apply hierarchical classification: firstly, the tweets are classified as before the event. *or* during or after the event. Then the tweets within the 'before' class are divided into smaller time categories.
- Besides using words as features, we will investigate how the use of character n-grams effects the performance.
- In the end, we compare the best performing system to the performance of humans for this task.

Before turning to the experiments, we provide an overview of the relatively large body of recent work on Twitter. This in order to gather some knowledge about the opportunities and challenges with this social medium.

Part I

Twitter

Chapter 2

Twitter characteristics

Twitter is an on-line microblogging service that enables its users to send and read text-based messages. It asks one question, namely "What's happening?", and answers are limited to a maximum of 140 characters. Those messages are called tweets and the core functionality is to easily and quickly share messages. A user is identified by a unique user name, and if user a decides to receive all the tweets from another user b, then b is a friend of a and, conversely, a became a follower of b. It is not uncommon for users, such as celebrities, to have millions of followers. This asymmetrical relationship between friends and followers is said to be one of the reasons for the popularity of Twitter.

Figure 2.1: Example of a tweet

Figure 2.1 above shows an example of a tweet, with some of the typical things you find in a tweet:

- Other users can be mentioned or replied to by their Twitter user name, which

is preceded by the @ sign. User names can contain up to 15 characters.

- Tweets beginning with the expression 'RT' are called 'retweets'. If people like a tweet, they copy the tweet preceded by 'RT', which is called 'retweeting'.

- Words within a message that are preceded by a '#', called 'hashtag', are used to assign the message to a topic, or to indicate that the word is a keyword. Clicking on a hashtag shows all other tweets marked with that hashtag. Hashtags can also be used in a different way; to express a feeling, sarcasm or side mark: "Off the drink and back in the gym for January. Can't wait! #CrappyNewYear" [19].

- The short allowance on tweet size means that users sometimes have to resort to phrase abbreviations. Interestingly enough, there is a pretty rich and well understood set of abbreviations which is surprisingly consistent. Even with the use of abbreviations it is often not possible to post verbose tweets. This is one of the reasons that tweets often contain a URL link to a related web site, video, etc.

- Tweets often contain spelling errors or they are grammatically incorrect, which is one of the challenges of doing research on tweets.

The first version of Twitter was introduced in 2006 as an internal service for Odeo employees, it spun of into its own company in 2007. Growing from 400,000 tweets posted per quarter in 2007 to 400 to 500 million each day nowadays [10], it is one of the most popular social media. Estimates are that in the Netherlands alone each day over 3 million tweets are posted [22].

An important common characteristic among microblogging services such as Twitter is its real-time nature. Blog users typically update their blogs once every several days, whereas Twitter users often write tweets several times a day. They report what they are doing or are thinking right now, and other users often check Twitter to see what other people are doing. In such a manner, numerous updates results in numerous reports related to *events*. They include social events, such as soccer games, and presidential campaigns, and disastrous events such as riots and earthquakes [26].

Chapter 3

Research

Twitter is only about five years old, but in spite of it being relatively young it has become a popular domain in text mining, on different subjects. Authorship analysis is one of those subjects, where among others gender and age of the person who posted the tweet are detected [20, 4, 21]. This could be used to improve targeting of advertisements or marketing, and can offer new insights in how different groups of people use language. Other subjects are the detection of sarcasm [16] and emotions [25], the recognition of threatening tweets [22] and many more.

Closer to our study are the studies about the detection of events, and in this domain research is already done as well. An overview of these studies will be given now. Only the studies about the detection of events using Twitter will be reported here, because Twitter is a unique medium due to for example its real-time nature and the shortness of the tweets. In this way the reader becomes familiar with the challenges and opportunities that Twitter offers.

3.1 Tweets as (real-time) sensors of events

Since tweets can be posted easily and quickly using apps and are limited to 140 characters, they are an ideal way for people to publish things spontaneously. As a consequence, there are short delays in reflecting what Twitter users perceive, compared to for example blogs. As a result, events that just started can be shared quickly. Several papers investigated how Twitter can be employed as real-time sensor of events, from major events such as celebrity deaths [29], to natural disasters, to less significant events, such as sport games [39]. Zhao *et al.* [39] believe that if it is possible to sense such small, more frequent events, then new innovations will arise that use humans as sensors of what is happening in the world through Twitter.

3.1.1 Detecting sport games

To investigate if Twitter can detect these frequent events, Zhao *et al.* first monitored the US National Football League games, and tried to recognize the games as soon as they happen. They demonstrated that these games can be fairly reliably recognized around 40 seconds after the event has started. On average it takes 1) 17 seconds for a Twitter user to report a game event, 2) 15 seconds to do the analysis they used, 3) 5 seconds to do the computing tasks and 4) 1 second to obtain the tweet from Twitter. A drawback of their work is that it is limited to events for which keywords can be predetermined.

3.1.2 Obtaining breaking news

Sankaranarayanan *et al.* [29] used Twitter to build a news processing system, called TwitterStand. In their paper they demonstrated how Twitter can be used to automatically obtain breaking news from tweets and to automatically obtain the opinions on current news. As a measure for determining the importance of a news topic they kept track of the number of tweets on the same news topic. Since users have some meta-data information (gender, location, friends, etc), Twitter can aid in finding users that are most likely to belong to a particular geographic region and news topics can be broken down into topics that are interesting for a special group, based on their gender, interests, location, etc.

Aggregating news from Twitter differs from conventional news aggregators in several ways. First of all, not all tweets posted are related to news, so first this has to be determined. Secondly, the brevity of the tweets often leads to a lack of context. Thirdly, Twitter breaks down communication barriers, so almost any action in the real word is reported on Twitter, and there is very little lag between the event happening and the time of posting of the tweet. This is illustrated by the death of Michael Jackson, where tweets about his illness and death were reported more than an hour before conventional news media reported it [29]. On top of that, Sankaranarayanan *et al.* claim that Twitter has a wider diversity of opinions on a news topic. Lastly, Twitter has the disadvantage that not all tweets are equally credible, which can result in misinformation being put out as news.

The idea of TwitterStand is to capture tweets that correspond to late breaking news, so the large stream of tweets first has to be labeled as either 'news' or 'junk'. Therefore Sankaranarayanan *et al.* introduced seeders: 2,000 handpicked users that are known to publish news, such as newspapers that publish news in tweets, or bloggers. To distinguish news from noise, a Naive Bayes classifier is used that is trained on a training corpus of tweets that has already been marked as either news or junk. This corpus has both a static and a dynamic component. The static component is made up of a large collection of labeled tweets, and the

dynamic corpus contains recent news tweets. The idea here is that the static corpus aids in identifying news tweets on topics that we have not encountered previously, while the dynamic corpus aids us in identifying tweets about current events. Their main goal is to automatically group the tweets labeled as 'news' tweets into sets of tweets, called clusters, where each cluster has a specific topic. What makes it hard is that TwitterStand does not know the identity of the topics beforehand. To solve this, they used a clustering algorithm which allows for clustering in both content and time and maintains a list of active clusters.

3.1.3 Earthquake detection

Several papers analyzed Twitter messages to investigate if tweet messages can be used for monitoring and reporting earthquakes [26, 6]. Most research on this subject is done in Japan, because Japan has numerous earthquakes and numerous (geographically dispersed) Twitter users. An earthquake propagates at about 3-7 km/s, so a person who is 100 km distant from an earthquake is able to act for about 20 seconds. Moreover, strong earthquakes regularly cause a tsunami, which are often more catastrophic. Therefore, prompt notification of an earthquake is very important to decrease damage, particularly by the tsunami.

First, the tweets about the event have to be detected, which is in this case an earthquake. Earle *et al.* crawl all tweets containing the word "earthquake" or its equivalents [6]. Then Earle *et al.* detected a rapid increase in earthquake tweets, using a customized version of the so called 'short-term-average over long-term-average' algorithm [7], which was already developed in the nineties. The system detected 48 earthquakes, with only two false triggers. This number is small compared to the more than 5,000 earthquakes noticed in the earthquake catalog, but the majority of these earthquakes are too small to produce perceivable shaking or they occurred outside populated areas. The detections on Twitter are of more immediate interest for humans. Twitter detected 75% of the earthquakes within two minutes of the origin time, and this is considerably faster than the seismographic detections in poorly instrumented regions of the world. So, Twitter earthquake detection could be a helpful support for conventional earthquake detection systems.

3.1.4 Augmenting information about planned events with Twitter

In the paper by Becker *et al.* a system for augmenting information about planned events with tweets is demonstrated, using a set of automatic query building strategies. They consider a tweet to be related to an event if it provides a reflection on the event before, during or after the event occurs. The keywords connected to

events were based on information from sites such as `upcoming.com`. In order to collect the right tweets, keywords are restricted to a location and specific words describing the event. Additionally, the results from over 50 event queries were labeled by hand, and high precision tweets were used to define new queries and retrieve additional event messages. In this way, a system is build which automatically displays tweets related to an upcoming known event.

3.2 Predicting popularity and outcomes

The study by Yu *et al.* [38] gives a survey of predictions made using social media in general. They claim that, if extracted and analyzed properly, the data on social media can lead to useful predictions of certain events, for example in the realm of finance, product marketing or politics. Predictions should be done on human related events, otherwise, like in the case of an eclipse, the development of the event occurs irrespective of people's thoughts about it. Besides that, the involved events should be easy to be talked about in public, and the composition of involved persons on social media should be similar to that in the real world. Examples they give of events that meet these criteria are product marketing, macroeconomics and politics.

3.2.1 Box-office revenues

Predicting box-offices revenues with Twitter is a well studied area [38]. Reasons for this are that there is a large number of users discussing movies, there is a substantial variance of opinions, there are many different movies, the real-world outcomes can be easily observed and it is of course commercially interesting. Asur *et al.* [1] investigated if box-office revenues of movies can be predicted in advance of their release. They showed that the rate at which movie tweets are generated can be used to build a powerful model. Moreover, their predictions are consistently better than the predictions produced by the traditional prediction market, the Hollywood Stock Exchange. In addition they studied how positive and negative opinions are distributed and how they influence people. They found that the sentiment content in tweets can only improve box-office revenue predictions based on tweets that are posted after the movies are released.

3.2.2 Election prediction

Election prediction is another hot topic in the area of predicting outcomes with Twitter. Traditionally, election polls are done via surveys, but this is time con-

suming and can lead to high costs. Twitter (and other social media) provide an opportunity to overcome these problems [38]. In the paper by Tumusjan *et al.* [32] it was shown that mere the number of messages mentioning a party reflected the outcome of the German federal election in 2009. They also found that joint mentions of two parties are in line with the political coalitions in the real world. The study by Sanders *et al.* [27] showed that counts of mentions of political party names are strongly correlated with the polls and the election results, but that polls remain more accurate as a predictor of the outcome. The tweet mention counts thus could form a good complementary basis for predicting election results

There is however an ongoing debate whether social media are effective in predicting the outcomes of elections. The study by Tumasjan *et al.* [32] for example did not take into account the smaller parties running for the elections, and the results depended on the time window used to compute them. Jungherr *et al.* [12] argue that the inclusion of an extra party (the Pirate Party), would have had a large negative effect on the accuracy of the predictions. Tjong Kim Sang *et al.* [28] show that only counting tweets that mention political parties is not sufficient to obtain good predictions. Gayo-Avello [8] goes even further and reasons in his paper that predicting elections from Twitter in this way is impossible. Motives are that not everybody is using Twitter and not every Twitterer is tweeting about politics, not all tweets are trustworthy and politics is plagued with humor and sarcasm.

3.3 Future events

The studies about future events are closest to this thesis. Research on this subject can be roughly split into two kinds: studies that predict *which* events will happen in the (near) future, or the prediction of *when* an event will take place given a tweet (about an event). The first study is about the first kind, the second combines them, and we will end with the second kind of studies, which are closest to this thesis.

3.3.1 Predicting popular events in the near future

Weerkamp *et al.* [34] tried to establish a set of activities that are likely to become popular at the upcoming evening, using Dutch Twitter data. They selected tweets referring to tonight by looking for tweets containing the words "vanavond" (tonight) or "vnvnd" (2nite). Then they performed a small-scale experiment; they selected three days (June 3-5, 2012) for which they extracted activities mentioned in the tweets. They used a method proposed by Sharifi *et al.* [31] for automatically creating summaries from a set of tweets related to the same topic. Then those activities were judged by two Dutch assessors, who were asked whether the

suggested activity was 1) really an activity , 2) properly summarized and 3) could be a popular activity for that evening. Some of the activities predicted were 'I'm going to bed early tonight', 'tonight squad training' and 'final episode of'. The results show the following:

- Spam can be mistaken for activities.
- Properly summarizing the activities is very important, but hard due to a relatively small number of tweets.
- Popularity is hard to estimate and evaluate.

So predictions are possible, but there is much room for improvement. The work especially identifies several issues that need further research. One of the issues is that much of the previously done work in text mining struggles with the limited number of tweets. Another thing is the time indication, because in the current experiment, only the tweets containing 'vanavond' or 'vnvnd' are used, but people use various ways to refer to tonight. Finally, the evaluation of this activity prediction is hard, because it needs to be done in three levels. A (semi) automatic evaluation would be welcome.

3.3.2 Open domain event extraction

Ritter *et al.* [24] propose a system that can extract and categorize open-domain events from Twitter. Previous work focused largely on newswire text and Twitter's unique characteristics present new challenges and opportunities for event extraction, of which some of them we already saw in previously mentioned studies. They have to deal with the mundane events frequently mentioned on Twitter about users' daily lives, which are not considered interesting as they do not matter to a sufficiently large group of people. Furthermore, because Twitter users can talk about whatever they choose, it is unclear in advance which set of event types are appropriate. In contrast to newswire, the complex reasoning about relations between events is absent.

Ritter *et al.* proposed that important events are those events whose mentions are strongly associated with references to a unique date, as opposed to dates which are evenly distributed across the calender. To address the diversity of events they came up with an approach based on latent variable models, that uncovers an appropriate set of types which match the data. Given a raw stream of tweets, the system extracts named entities in association with event phrases and unambiguous dates which are involved in significant events. First the tweets are part-of-speech (POS) tagged, then named entities and event phrases are extracted, temporal expressions resolved, and the extracted events are categorized into types. Next

they measured the strength of association between each named entity and date, based on the number of tweets they co-occur in, in order to determine whether an event is significant.

Current tools developed for natural language processing can perform poorly when applied to Twitter text, due to its noisy and unique style. To address these issues, they utilized a named entity tagger trained on in-domain Twitter data, that was presented in an earlier study.

The results of the experiment show that accurately extracting an open-domain calendar of significant events from Twitter is indeed feasible. The approach the authors used achieved a 14% increase in F1-score[1] over a supervised baseline. Looking at the 100 highest ranked calendar entries over a two-week date range in the future, the precision of extracted (entity, date) pairs was 90%. A continuously updating demonstration of this system (an open domain event calendar) can be viewed at `http://statuscalendar.com`.

3.3.3 Predicting time-to-event

The studies by Kunneman *et al.* [13] and Hurriyetoglu *et al.* [11] are closest to our study. They both investigated how to predict the time-to-event of tweets (using football matches as events), but in different ways. The first study trained a classifier to distinguish 'before' tweets from 'during' or 'after' tweets. The second uses regression methods to assign to each test tweet the number of hours still remaining to the event.

Phase classification of events The study by Kunneman *et al.* [13] investigates if it is possible to discriminate between tweets referring to upcoming events and tweets that refer to past football matches, both for scheduled and unscheduled events. In the first experiment, different supervised machine-learning classifiers are trained to distinguish before-match tweets from tweets generated during or after a match. They classified league matches, play-off matches and tweets about the Champions League final only using tweets about league matches to train with.

The classifications reveal that 'before' tweets can be distinguished from 'not before' tweets reasonably accurately, regardless of the slight event type variations. The best performing classifier had the following F1-scores: 0.88 for the league matches, 0.79 for play-off matches and 0.74 for the Champions League final. Doing the same experiment for unscheduled events (football transfers), the performance was poor. None of the classifiers had a better F1-score than baseline performance.

[1] In statistics, the F1-score is a measure of the accuracy of the classifier. It is the harmonic mean of precision and recall, see Chapter 10.3 for a more detailed explanation.

They conclude that the presumed similarity between anticipating tweets, regardless of the event type, is not apparent.

Extracting of events continuously Hurriyetoglu *et al.* [11] investigated three regression methods that estimate the time-to-event of a series of tweets that refer to a future event. Sets of tweets are represented as a feature vector (where each word occurring in the set of tweets is a feature) and then the task to predict the time to event is seen as a regression problem: the feature vector is mapped onto a continuous numeric output, which represent the time to event.

This study did not take into account tweets that were posted during and after matches, and for both training and test sets, tweets were kept within eight days before the event (which contains 98% of the tweets). The events they used were football matches (as in the previously mentioned study), although they only used matches that had the same starting time (Sunday, 2:30 PM).

They evaluated their numeric predictions by computing the Root Mean Squared Error[2]. They calculated the RMSE scores for each of the methods and for each event separately. The results showed that the best performing method was on average (taking all events together) 43 hours off, which is better than baseline methods. However, per event the best performing method differed. Moreover, looking per hour, there were differences between the methods as well; one method was the best in predicting events that are still far away, but its performance for events that are almost happening was worst, and vice verse. Unfortunately, none of the tested systems had a consistently strong performance.

3.4 Summary

In sum, the studies described above show that Twitter indeed can be used as a new source of the prediction and extraction of events for all kinds of research areas. Especially the study about the real-time extraction of events revealed great results. The studies about whether or not it is possible to make predictions out of the stream of tweets has brought out contradictory positions. At least in the realm of politics this is not a foregone conclusion. The extraction and prediction of future events had some good results, but much room is left for improvement. In the study by [11] it appeared that there is not a single method that consistently performs the best. In the upcoming experiments we will investigate the weaknesses and strengths in the prediction of the time-to-event of tweets, and test several methods in order to obtain the optimal classification procedure.

[2]The Root Mean Squared Error, or RMSE, is a frequently used measure of the differences between values predicted by a model or an estimator and the values actually observed. For a detailed explanation see Chapter 10.1.

Part II

Technical overview

Chapter 4

Text classification

Text classification is the problem of automatically assigning texts (news articles, tweets, emails, etc) to some predefined classes. This can be done by hand, like assigning books to a specific category in the library, according to certain rules. A rule, which captures a certain combination of keywords that indicates a class, must be known beforehand. However, finding these rules is not trivial and can be labor intensive. That is why people came up with *statistical text classification*: the set of rules is learned automatically from training data. The need for manual classification is not eliminated, because the training documents have to be labeled[1], in other words each training document has to be annotated with its class. The next chapter will give a short introduction into three of the algorithms that are commonly used in statistical machine learning, and that will be used in this study as well.

To improve the efficiency and performance of these algorithms, a technique called feature selection is commonly applied in text classification. A short introduction in two of these feature selection methods will be given first.

[1]There exists a subject within the field of machine learning where this is not necessary, called unsupervised learning, that tries to find hidden structures in unlabeled data.

Chapter 5

Feature selection

Feature selection is the process of selecting a subset of the terms occurring in the training set, and using only this subset as features in the text classification. This procedure serves two main purposes. First, by decreasing the vocabulary size training will be more efficient. This is especially important for classifiers that are expensive to train, such as the Support Vector Machine, which is one of the classifiers that we will be using. Second, by selecting a subset of features often the classification accuracy can be increased, because noise features will be eliminated. A noise feature can be a rare term, that for example happens to occur in all tweets that are posted in the class 'during the event', but that has no information about that class (for example 'laptop'). Then the classifier might incorrectly classify (called 'overfit') a test tweet containing this feature as 'during the event' [17, 15]. Some classification algorithms suffer more from overfitting than others, for example the k-nearest neighbors classifier is usually not robust against overfitting. Next an explanation of two often-used feature selection methods will be given.

5.1 Information Gain

Information Gain is a common feature selection method [17], which computes the mutual information of each feature in isolation. The Information Gain of term t measures how much information the term contributes to making the correct classification. In other words, it measures the decrease in uncertainty (entropy) in making the correct classification when the value of the term (its presence or

absence) is known. The Information Gain of term t is defined to be [37]:

$$\begin{aligned} IG(t) = &- \sum_{i=1}^{m} P(c_i) \log P(c_i) + P(t) \sum_{i=1}^{m} P(c_i|t) \log P(c_i|t) \\ &+ P(\bar{t}) \sum_{i=1}^{m} P(c_i|\bar{t}) \log P(c_i|\bar{t}) \end{aligned} \tag{5.1}$$

where m is the number of classes, $P(c_i)$ the probability of class c_i and $P(t)$ and $P(\bar{t})$ the probabilities of the presence and absence of a term, respectively. The Information Gain is computed for all terms and ranked from highest to lowest.

5.2 Chi Squared

White *et al.* [35] proposed a feature selection measure that is based on the Chi Squared (χ^2) statistic. In statistics, the χ^2 test is used to examine the independence of two events. Two events, X and Y are independent if:

$$P(X \cap Y) = P(X)P(Y) \tag{5.2}$$

In text feature selection, these two events correspond to the occurrence of a particular term and class. Assuming that there is no predictive association between term t and class c, the χ^2 information can be computed using the following formula [17]:

$$\chi^2(t, c) = \sum_{e_t \in \{0,1\}} \sum_{e_c \in \{0,1\}} \frac{(N_{e_t e_c} - E_{e_t e_c})^2}{E_{e_t e_c}} \tag{5.3}$$

where $e_t = 1$ if the training instance contains term t and 0 otherwise, and $e_c = 1$ if the training instance is in class c, and 0 otherwise. N is the observed frequency and E the expected frequency. For example, E_{11} is the expected frequency of t and c occurring together in the data, assuming that they are independent. In formula: $E_{11} = N \times P(t) \times P(c)$.

A high value of χ^2 indicates that the hypothesis of independence, which implies that expected and observed counts are similar, is incorrect. If the two events are dependent, the occurrence of the term makes the occurrence of the class more likely and consequently, the term is relevant as a feature. Whether or not to reject the null hypothesis can be read from the χ^2 distribution with ν degrees of freedom, with $\nu = (m-1)(k-1)$, with m the number of classes and k the amount of terms [35].

Chapter 6

Classification methods

To investigate what would be the best method to approach our problem, we train with different classification techniques and compare their performances. The techniques we use are Naive Bayes, k-nearest neighbors (k-NN) and Support Vector Machines (SVM). These classifiers are widely known and they belong to the most influential data mining algorithms in the research community [36]. Besides that, these three classifiers are shown to be most appropriate in the existing literature about text classification [2]. Naive Bayes is a rather simple classifier, easy to implement and very fast. k-nearest neighbors classifies test data based on the closet training examples. The training phase is very short, but the classification time is long and it is difficult to find the optimal value for k. The SVM classifier is seen as one of the most effective classifiers [2], but the trainig of the classifier takes long and this classifier is somewhat harder to understand. In the upcoming sections a short and concise introduction to the central notions of these classifiers will be given, for a more thorough account the reader is asked to consult [17, 9] or one of the many other good text books on Machine Learning.

6.1 Naive Bayes

The Naive Bayes classifier is based on the so-called Bayesian theorem and it greatly simplifies learning by assuming that features in each class are independent. Despite its simplicity, Naive Bayes often competes well with more sophisticated classifiers [23]. Bayesian classifiers assign the most likely class to a given example (tweet) described by its feature vector. Bayes' theorem gives the relationship between the

probabilities of class c and feature vector $\mathbf{x}$, and the conditional probabilities[1] of c given $\mathbf{x}$ ($P(c|\mathbf{x})$) and c given $\mathbf{x}$ ($P(c|\mathbf{x})$):

$$P(c|\mathbf{x}) = \frac{P(\mathbf{x}|c)\,P(c)}{P(\mathbf{x})} \tag{6.1}$$

The Naive Bayes assumption states that each feature is conditionally independent of every other feature, in formula: $P(x_1 \dots x_n|c) = \prod_i P(x_i|c)$. Using Bayes rule and the independence assumption, the probability of an example $\mathbf{x}$ being in class c is computed as:

$$P(c|x_1 \dots x_n) = \frac{1}{P(x_1 \dots x_n)} P(c) \prod_{i=1}^{n} P(x_i|c) \tag{6.2}$$

Since the fraction does not depend on c and the values of the features x_i are given, the fraction is effectively constant. So the equation can be rewritten as:

$$P(c|x_1 \dots x_n) \propto P(c) \prod_{i=1}^{n} P(x_i|c) \tag{6.3}$$

Our goal is to find the most likely class, or maximum a posteriori (MAP) class c_{map}, for the instance $\mathbf{x}$:

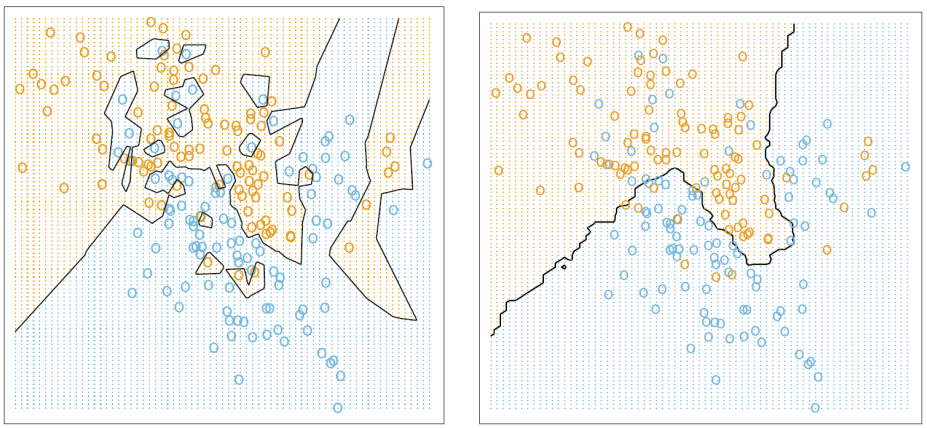
$$c_{\text{map}} = \arg\max \hat{P}(c|\mathbf{x}) = \arg\max \hat{P}(c) \prod_{i=1}^{n} P(x_i|c) \tag{6.4}$$

The prior $\hat{P}(c)$ is the relative frequency of c, which can be computed by dividing the number of instances belonging to class c by the total number of instances. The conditional probability estimates $P(x_i|c)$ are weights that indicate how suitable an instance x_i is for a class c. This probability can be estimated as the relative frequency of term x_i in instances belonging to class c. So we do not take into account the position of the feature x_i (i.e. we do not take into account the place of the feature in the training data). There is one problem left, namely that the estimate is zero for a feature-class combination that did not occur in the training data. Since we are multiplying, the conditional probability for that class will then be zero. To eliminate zeros, often add-one smoothing is used. This can be seen as a uniform prior, where each term occurs once for each class, so there is no multiplying by zero possible anymore [17].

[1] A conditional probability is the probability that an event will occur, when another event is known to occur or to have occurred

6.2 k-nearest neighbors

The k-nearest neighbors (k-NN) classifier assigns the majority class of the k-nearest neighbors to a test instance. The k-NN classifier is memory-based and requires no explicit training, it can use the unprocessed training set directly in classification.



(a) 1-nearest neighbor (b) 15-nearest neighbors

Figure 6.1: k-nearest neighbors classification [9]. These figures illustrate how the classification is influenced by the choice of k.

Given an instance, we find the k training instances closest in distance to that instance, and then classify the instances based on the class with the majority vote among the k neighbors. As distance measure we use the Euclidean distance, that is the "ordinary" distance between two points that one would measure with a ruler. Tied majorities are broken at random. k-nearest neighbors classification is often successful where each class has many possible prototypes and the decision boundary is very irregular [9].

The choice of k is very important. Taking $k = 1$, each training point is a prototype, the incorrectly labeled or atypical training points as well. So $k = 1$ is not always robust and often $k = 3$ or $k = 5$ is used, but much higher values are also possible. Figures 6.1(a) and 6.1(b) illustrate how the classification is influenced by the choice of k. Figure 6.1(a) shows the k-NN classification applied to binary data for $k = 1$ and figure 6.1(b) for $k = 15$. The decision boundary for $k = 15$ is fairly smooth compared to the decision boundary for $k = 1$.

6.3 Support Vector Machines

6.3.1 Introduction

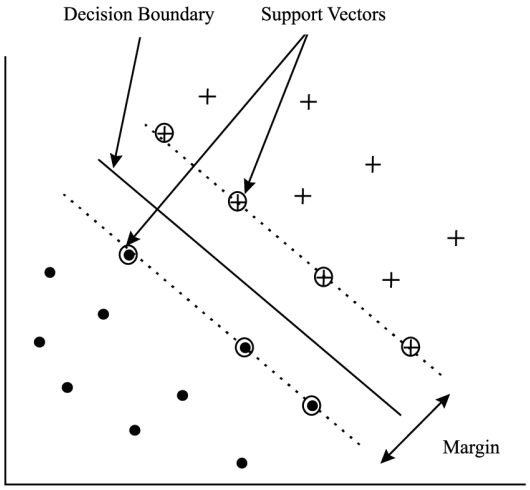


Figure 6.2: Support vectors, decision boundary and margin.

Support Vector Machines (SVMs) were introduced by Boser, Guyon and Vapnik in 1992 [3]. SVMs are developed from Statistical Learning Theory from the sixties, so they are theoretically well motivated, and since their introduction they became rather popular.

SVM is a vector spaced machine learning method that tries to find a decision surface between two classes that is maximally far from any point in the data set, where sometimes points may be discounted as outliers. The distance from this surface to the closest data point determines the margin of the classifier, and those points are called the *support vectors* (see Figure 6.2). The margin of the classifier should be as wide as possible. Since we are only looking at the closest data points, the position of the separator is specified by an (often small) subset of the training data, and the other data points play no part in determining the surface.

Why do we want to maximize the margin? Firstly, a classifier with a large margin does not make classification decisions with low certainty, so an error in measurement or an outlier will not lead to a miss-classification. Secondly, because we are seeking for a large surface instead of just a (thin) decision boundary, there are fewer options to put the surface, and therefore the classification will be faster. This is illustrated in Figure 6.3.

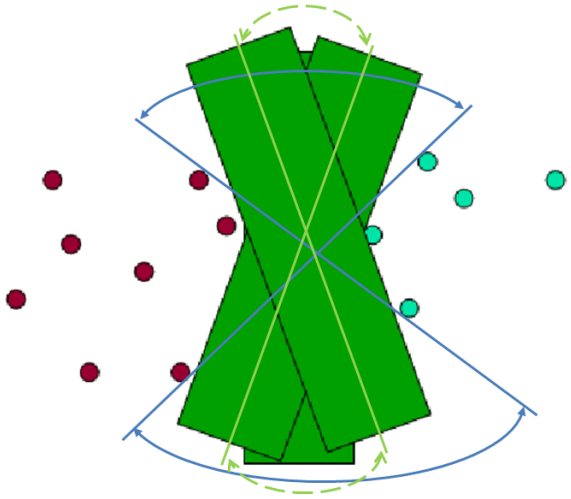


Figure 6.3: Illustration of the large-margin classification. The range of angles at which the decision surface (green, dotted line) can be placed is much smaller than for a decision line (blue).

6.3.2 Formal definition

We will give a formal definition for two-class, linearly separable data sets, in order to make the explanation manageable. The two classes are always named $+1$ or -1 (not 0 and 1). A decision boundary (or decision hyperplane) can be defined by a normal vector $\vec{\beta}$ that is perpendicular to the hyperplane, and an intercept term β_0. The training data consists of N pairs $(x_1, y_1), (x_2, y_2), \ldots (x_N, y_N)$, with $x_i \in \mathbb{R}^n$ (with n the number of features) and $y_i \in \{-1, 1\}$. All points $\vec{x}$ on the hyperplane satisfy $\vec{\beta}^T\vec{x} + \beta_0 = 0$, because the hyperplane is perpendicular to $\vec{\beta}$. The classification rule can then be stated as follows:

$$f(\vec{x}) = \text{sign}(\vec{\beta}^T\vec{x} + \beta_0) \tag{6.5}$$

We could now define the functional margin of a data point $\vec{x_i}$ with respect to a hyperplane $< \vec{\beta}, \beta_0 >$ as $y_i(\beta^T\vec{x_i} + \beta_0)$. The functional margin of the data set with respect to a decision surface is then twice the functional margin of any of the points that are closest to the decision boundary, which are the points with the smallest functional margin (see Figure 6.4). However, using this definition we can always make the functional margin bigger by scaling up β and β_0 by the same number. So we need a constraint on the size of the vector β, for which we need to do some geometry. The shortest distance from a point $\vec{x}$ to the decision boundary is perpendicular to that boundary, and hence parallel to β. Let us call this distance r (see Figure 6.4), and its direction can be described with the unit vector $\beta/\|\beta\|$. The point $\vec{x}'$ on the hyperplane closest to $\vec{x}$ can then be described

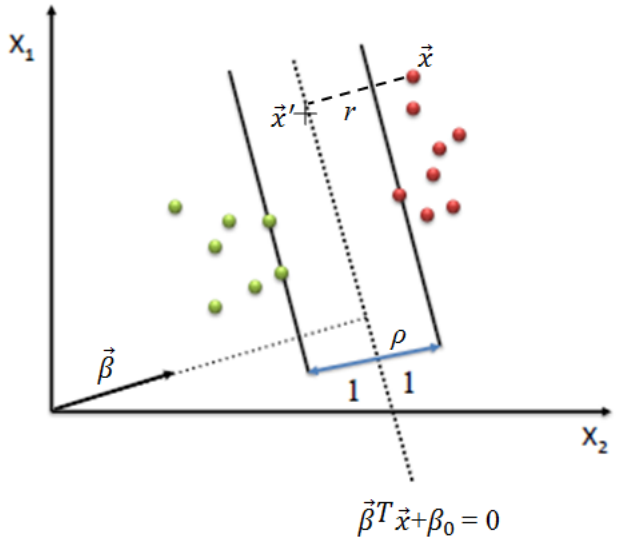


Figure 6.4: The geometric margin.

by:

$$\vec{x}' = \vec{x} - yr\frac{\beta}{\|\beta\|} \tag{6.6}$$

where multiplying by y changes the sign for being $\vec{x}$ on either side of the decision boundary. Since $\vec{x}'$ lies on the decision boundary, it satisfies $\beta^T\vec{x}' + \beta_0 = 0$. Replacing the definition of $\vec{x}'$ in (6.6) in this equation gives:

$$\beta^T(\vec{x} - yr\frac{\beta}{\|\beta\|}) + \beta_0 = 0 \tag{6.7}$$

Using the fact that $\|\beta\| = \sqrt{\beta^t\beta}$, this can be rewritten as:

$$\beta^T\vec{x} - yr\|\beta\| + \beta_0 = 0 \tag{6.8}$$

Solving for r gives[2]:

$$r = y\frac{\beta^T\vec{x} + \beta_0}{\|\beta\|} \tag{6.9}$$

Again, the points closest to the separating hyperplane are support vectors. The geometric margin of the classifier is defined as the maximum width of the band that can be drawn separating the support vectors of the two classes. That is, the maximal width of one of the fat separators in Figure 6.3, or, in formula,

[2]Recall that $y = \{-1, 1\}$

twice the minimum value over data points for r given in Equation 6.9. Scaling the parameters $\vec{\beta}$ and β_0 does not influence the geometric margin, because it will always be normalized by $\|\vec{\beta}\|$, the length of $\vec{\beta}$. So we can impose any scaling constraint on $\vec{\beta}$, without affecting the geometric margin, and we will require that $\vec{\beta} = 1$. This has the effect of making the geometric margin the same as the functional margin.

As said before, we can scale the functional margin as we please, so we can assume that the functional margin of all training data points is at least 1 and that it is equal to 1 for at least one data vector. So for all data points i it holds that $y_i(\beta^T \vec{x_i} + \beta_0) \geq 1$, and the points for which the inequality is an equality are the support vectors. In equation (6.9) the distance from the decision boundary to each data point is given, and the geometric margin can then be defined as $\rho = 2/\|\beta\|$. We still want to maximize this margin, so we want to find β and β_0 such that:

- $\rho = 2/\|\beta\|$ is maximized
- for all training data $(x_i, y_i), 0 \leq i \leq N, y_i(\beta^T \vec{x_i} + \beta_0) \geq 1$

This can be more conveniently rephrased as:

Find β and β_0 such that:

- $\frac{1}{2}\beta^T\beta$ is minimized
- for all $(x_i, y_i), y_i(\beta^T \vec{x_i} + \beta_0) \geq 1$

We now try to solve a quadratic optimization problem, and those problems are a well-studied class of mathematical optimization problems, and there exist many algorithms to solve them. Solving them involves the introduction of Lagrange multipliers, but the details will not be presented now.

6.4 Summary

In this chapter we gave a short theoretical introduction to the techniques that will be used for the classification. We saw two different methods to select the most informative features, and then described three different algorithms to classify the test data. During our experiments we will investigate these methods and discover which training regime will be the best.

Part III

Experimental set up

Chapter 7

Data collection

To be able to do the classification, appropriate data must first be collected. For this study we focus on tweets about football matches played in the Eredivisie (the highest-level Dutch football league) as events. They usually generate a sufficient amount of tweets; thousands to several tens of thousands per match. Moreover, they occur frequently and they are referred to by a distinctive hashtag by convention (#ajapsv for a match between Ajax and PSV). Finally, for any selected football match the date of the match is known. To collect the tweets, we search for tweets containing the hashtag belonging to that event. Although in this way we will not find every tweet referring to the event, the training data is at least trustworthy and still generates enough data. The tweets were queried via twiqs.nl.

We selected the top 6 teams of the league in 2011, which were at that moment Ajax, PSV, Feyenoord, FC Utrecht, FC Twente and AZ, and queried tweets referring to all matches played between them in the years 2011 and 2012, which resulted in 60 matches. As mentioned before, we searched for the tweets by hashtag, and to ensure that the tweets were referring to that particular match - and not to an earlier or later match with the same clubs and the same home and away team - we only collected the tweets three weeks before or after the event. We ended up with a collection of 703,767 tweets. The event most tweeted about (62,003 tweets) is the match between Ajax and FC Twente in the spring of 2011, which is not surprising since in that match the champion (Ajax or FC Twente) of that year was decided. The match between FC Utrecht and AZ, in the spring of 2011 as well, generates the fewest tweets (716), and on average 11,730 tweets were collected per match.

In Table 7.1 two examples of collected tweets are shown. For each tweet we have the event it is referring to (first column), the ID of the tweet (second column), the

Table 7.1: Examples of the collected tweets

Event	ID	User name	Date	Time	Message
ajaaz_s11	43401817899536384	Maxiecosy	2011-03-03	21:06:14	`haha zondag naar #ajaaz`
psvfey_s12	173759034179059712	Luccpeeterss	2012-02-26	14:19:16	`nu gaat het echte werk beginnen #psvfey`

user name (third column), the time and date of posting (fourth and fifth column) and the content of the tweet (last column).

Further investigation of the data reveals that no less than 42% of the tweets are retweets, and we decide to remove these tweets. They are primarily removed because they can be posted several days after the original tweet was posted while containing the same words. Another reason is that it is often a marketing stunt that receives a lot of retweets and therefore can mislead the classifier. An example of such a marketing stunt is shown below, which received almost 200 retweets.

```
rt @psv wat moet je doen rt dit bericht van @psv en maak daarmee kans
op een van de twintig gratis kijkcodes voor #feypsv op psv tv on
```

7.1 Data representation

The tweets are put into a sparse binary format. Each tweet is converted into a vector, containing the numbers of the features that the tweet contained. Each feature has a index, and if a tweet contains a certain feature then the related index is added to the vector. At the end we add the temporal value, based on the time of posting relative to start time of the match they were referring to. An example:

$$\overbrace{4, 54, 152, 918, 8281, 10659, 20990, 63791}^{\text{features}}, \overbrace{-460}^{\text{time}}$$

Chapter 8

Categorization

We build a classification-based system, in contrast to a regression based system. We thus map unseen tweets onto discrete time segments, and in order to do that we have to distribute the continuous stream of the time in tweets between several time categories. Since the creation of the time categories is just a necessary step in order to be able to do the classifications, we can choose them any way we want, as long as we think that it will yield a good time-to-event prediction in the end.

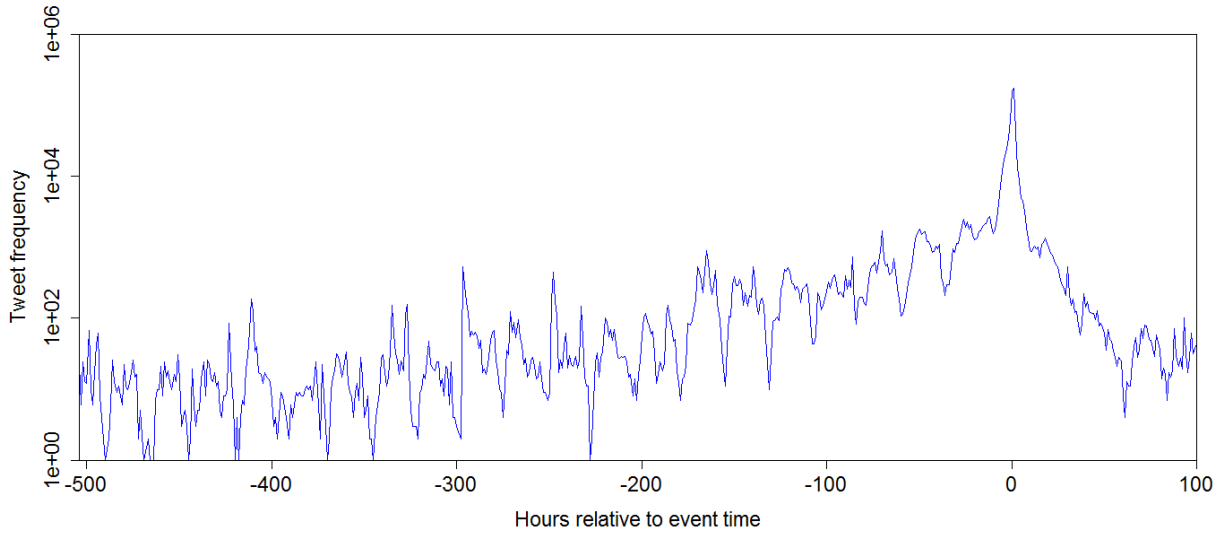


Figure 8.1: Average number of tweets per hour before and after the event starting time. Y-axis is logarithmic.

At the most general level, tweets referring to an event can be divided into the categories 'before', 'during' and 'after'. As we are interested in estimating the time-to-event for each tweet at some resolution, the 'before' category needs to be divided into more specific time categories. These time categories, as said before, are just a step that is necessary in order to be able to do classifications, and we should investigate which distribution of time categories gives best results in the end. One of the most straightforward ways to do this would be to distinguish time categories by a fixed length of time. However, as can be seen in Figure 8.1, the vast majority of the 'before' tweets is posted during the hours right before the start of an event (note that the y-axis of the figure is logarithmic). Taking all events together, we see that in the final hour before the events started, 62,000 tweets are posted, compared to fewer than 7,500 during the sixth hour before the events. Thus, when splitting the data into fixed time segments longer than a few hours, almost all tweets would be residing the category closest to event time, and it can be expected that such a distribution will not lead to good predictions of the time to event. Another straightforward way would be to distinguish time categories by a fixed number of tweets. This again does not seem the right way because of the highly skewed distribution of the tweets over time. For example, dividing the 'before' category in 7 classes, where each class contains an equal amount of tweets, would give us 5 classes that contain only the 24 hours closest to event, and there will be two classes left for the remaining tweets. Therefore we experiment with two alternative division schemes: 'left branching' and 'k-means' clustering.

8.1 Left branching

Applying the left branching scheme, the data is split into progressively longer time categories, where each split breaks the remaining tail in two. Starting from the 'before' category as a whole and moving from the start of the events backwards, categories are split at the hour where the amount of tweets on both sides is approximately equal. After each split, the category on the right (closer to event start time) is fixed and another split is performed on the category on the left (see Figure 8.2). This process is repeated until the most distant time category is more than 10 days before the start of an event. Splitting further would give classes with too little instances to train on. This resulted in seven 'before' categories.

This division scheme takes the imbalanced temporal distribution of the data into account by making splits based on frequency. The time segments closer to the event will still represent shorter time spans than the time segments further away, but in this way large variations of time segment lengths are restricted by only making splits in a left branching fashion. The time categories that resulted from this division scheme are displayed in the upper rows of Table 8.1.

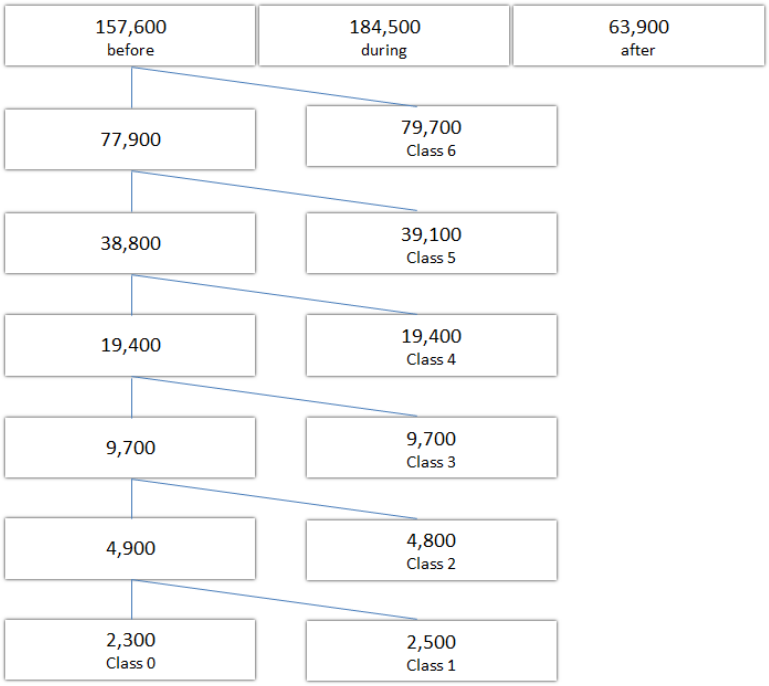


Figure 8.2: Schematic diagram of the left branching method.

8.2 k-means clustering

For the k-means clustering division scheme we selected the time categories with the smallest Sum of the Squared Error (SSE) with respect to their centroids. In order to compare k-means clustering to the left branching approach, we fixed the number of time categories to 7. These seven categories each contained an equal amount of tweets. Then for each time category the centroid was computed, and for each instance the Euclidean distance to the centroids is computed. Instances are assigned to the category with the nearest centroid, and then the SSE with respect to the new centroids is recomputed for the new division. This process is repeated until the SSE stops decreasing. The result of this division scheme is displayed in the lower rows of Table 8.1.

In comparison to the left-branching scheme, k-means clustering takes both the content and the distribution of tweets into account. This way, time segments that are more coherent (possibly in a contextually meaningful way) will be favored. As can be seen from Table 8.1, k-means clustering leads to more compact time categories close to event start time and less compact time categories further away from an event, in comparison to the left-branching approach.

Table 8.1: The two categorization schemes. The upper row gives the categorization in hours, the middle row in days, and the lower row the number of tweets (round to hundredths) in each category.

Category	0	1	2	3	4	5	6	7	8
Left Branching	$504-298$ $21-12$ 2,300	$297-219$ $12-9$ 2,500	$218-151$ $9-6$ 4,800	$150-79$ $6-3$ 9,700	$78-32$ $3-1$ 19,400	$31-5$ $1-0$ 39,100	$4-1$ $0-0$ 79,700	during	after
***k*-means**	$504-267$ $21-11$ 2,100	$266-132$ $11-6$ 7,200	$131-69$ $5-3$ 9,700	$68-34$ $3-1$ 12,400	$33-13$ $1-1$ 22,600	$12-4$ $1-0$ 32,300	$3-1$ $0-0$ 71,300	during	after

Chapter 9

Features

As features we extract all uni-, bi- and trigrams of words from the tweets. By combining the three sorts of n-grams, bonuses are awarded to matchings on longer n-grams, on top of the weights that their underlying unigrams already represent. We mark the beginning and end of each tweet as well, as extra bi- and trigrams (for example '*start*, goal', means that the tweet started with 'goal'). This results in a huge amount of potential features; almost 4 million (see Table 9.1). As we explained in Chapter 5, such a giant vocabulary can decrease the classification performance. So we will have to reduce the amount of features that we use in our classification. First we perform a selection based on frequency; only those unigrams that appear more than 10 times, and those bi- and trigrams that appear more than 40 times are selected. This reduces the amount of features to a little more than 60,000 features (see Table 9.1). Additionally the length of the tweet, rounded at a step size of five words, is encoded as an extra feature.

Table 9.1: Number of features with and without frequency based feature selection (FS).

	unigrams	bigrams	trigrams	total
without FS	209,738	1,227,256	2,507,746	3,944,740
with FS	18,768	25,599	19,422	63,789

Then we select those features from the remaining 64 thousand that are most relevant, with the two feature selection methods Information Gain and χ^2 (see Chapter 5). To have a clue how the two feature selection methods differ, and what kind of features are selected, the top 30 of each of the four combinations (two segmentation and two feature selection methods) are shown in appendix A.

9.1 Number of selected features

Applying the feature selection methods gives a ranking from features containing the highest amount of information to the lowest. The number of features that will be used by the classifiers has to be decided. We varied this number from 10,000 in the main experiment, to 1,000 and 100 in the additional experiments. By using small amounts of features, (a lot of) information will be thrown away, and some tweets will even have no features at all. Using a large amount of features will make the classifier less efficient, and at some point using more features does not (significantly) attribute to the performance of the classifier anymore, and may even lead to worse performance due to overfitting. Table 9.2 shows the percentage of tweets that only have a few (zero or one) features, for different amounts of selected features. The amount of instances that does not contain any feature at all appears to be approximately equally distributed in relation to the classes. The classifiers will classify these tweets as 'during', being the largest class (and hence the largest probability that the instance will belong to that class). Selecting 10,000 features ensures that all tweets are represented by one or more features. Interestingly, using χ^2 feature selection always entails a higher percentage of tweets containing only a few features.

Table 9.2: Percentage of tweets with 0 or 1 features

# selected features	#features in each tweet	IG	χ^2
100	0	1.94%	5.43%
100	1	14.95%	25.22%
1,000	0	0.01%	0.05%
1,000	1	0.43%	0.04%
10,000	0	0%	0%
10,000	1	0.03%	0.04%

Once the features are selected, the three classifiers will classify the test data into the classes.

Chapter 10

Evaluation

10.1 RMSE

To test the performance of our systems, we performed 'leave-six-events-out' cross-validation. This method approximates 10-fold cross-validation, and involves repeated splits between 6 events as test data and 54 as training data (remember that there are 60 events in total). This is repeated ten times such that each event is used once as test data. Folds were split at the event level because it is unrealistic to both train and test on tweets referring to the same event.

In order to score the amount of time in hours that a system is off in its estimations, we calculated the Root Mean Squared Error (RMSE), a common metric for evaluating numeric predictions. The sum of the squared differences between the actual value v_i and the predicted value e_i of all predictions is calculated, their mean is computed, and then the square root is taken to produce the RMSE (see equation 10.1).

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(v_i - e_i)^2} \tag{10.1}$$

As predicted value for each tweet we took the mean, median, first and last element of the predicted class, in order to investigate what would be the best value to take as predicted value. For example, when a tweet is posted 78 hours before an event and the system classifies the tweet in a category that ranges from 266 to 132 hours before event time, the squared errors are given in Table 10.1.

As only the seven 'before' categories can be expressed as numeric values, as opposed to the nominal 'during' and 'after' categories, the RMSE can only be

Table 10.1: Example of Squared Errors for a tweet that is posted 78 hours before the event, and is classified in a category that ranges from 266 to 132 hours before event

	Predicted value category $(-266, -132)$	Squared Error	
first element	-266	$(78-266)^2$	$= 35,344$
last element	-132	$(78-132)^2$	$= 2,916$
mean	-173.8	$(78-173.8)^2$	$= 9,177.64$
median	-163	$(78-163)^2$	$= 7,225$

calculated when both the classification and the true label are one of these seven categories. On top of that, if we would include the calculation of the RMSE of the 'during' and 'after' classes, it would be harder to correctly interpret the evaluation. Firstly, the 'after' class is not further divided into classes, so the RMSE of instances will still be high, even if they are classified correctly. Besides that, it is harder to know what caused the better (or worse) RMSE. Is it because the large (and rather small in terms of the distribution of hours) 'during' class was better classified? That is not what we are primarily interested in.

Only reporting the RMSE of the 'before' classes is not sufficient, because then we are not informed about the amount of tweets that is classified as 'during' or 'after'. Therefore, in addition to their RMSE we measured the *responsiveness* of each system: the relative number of occasions where the classifier generated a 'before' classification (classifying with one of the categories 0-6) when it should have. A system with a low RMSE and a high responsiveness is to be favored over a system with a low RMSE and a low responsiveness.

10.2 Standard deviation

We are calculating the mean RMSE of the 10 folds of the 'leave-six-events-out' cross validation. In order to express the variability in RMSE of the 10 folds and to measure the confidence of our conclusions, we calculate the standard deviation. The standard deviation σ is calculated as follows:

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2} \tag{10.2}$$

with N the number of folds, μ the mean RMSE of the folds and x_i the RMSE of fold i.

The certainty with which these averages are measured are expressed in the standard deviation. To be able to compare the analytic results obtained with two different methods on the same sample, we apply a significance test; a two-tailed paired t-test. The t-test gives the probability that the difference between the means of the classification with two different methods is caused by chance. It is customary to say that if this probability is less than 0.05, that the difference is 'significant', and the difference is not caused by chance [14].

10.3 F1-score and distance error

10.3.1 F1-score

Apart from RMSE rates, we calculated the F1-score [33] and distance error for each separate category. The F1-score is the harmonic mean of precision and recall. The precision of the classifier is the proportion of retrieved material that is relevant (classified correctly), recall is the proportion of relevant material actually retrieved. Recall and precision attempt to measure the ability of the classifier to retrieve relevant documents while at the same time holding back non-relevant one. In formula:

$$\begin{aligned} \text{precision} &= \frac{tp}{tp + fp} \\ \text{recall} &= \frac{tp}{tp + fn} \\ \text{F1-score} &= 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \end{aligned} \tag{10.3}$$

where tp = true positives, fp = false positives and fn = false negatives as defined in Figure 10.1. For example, computing the F1-score for class 1, a tweet that belongs in class 1 but is predicted as another class (even as 'during' or 'after'), is a false negative. The F1-score is indicative of the correct classification of each category taken individually. Besides calculating the F1-score for each separate category, we calculated the average F1-score as well. Apart from calculating the average of the F1-scores, we also calculated the average F1-scored according to the MICRO-average method. For the last method, the precision and recall, for a system with n classes, are calculated as follows [30]:

$$\begin{aligned} \text{precision} &= \frac{tp_1 + \cdots + tp_n}{tp_1 + \ldots tp_n + fp_1 + \cdots + fp_n} \\ \text{recall} &= \frac{tp_1 + \cdots + tp_n}{tp_1 + \ldots tp_n + fn_1 + \cdots + fn_n} \end{aligned} \tag{10.4}$$

The MICRO-F1 score can now be calculated in the same way as the F1-score. We chose to calculate the MICRO-F1 scores as well, since the micro-averaging is more suitable when a dataset varies in size [30], giving each instance the same weight, which is the case in our experiments. For the F1-scores we calculate the standard deviations as well.

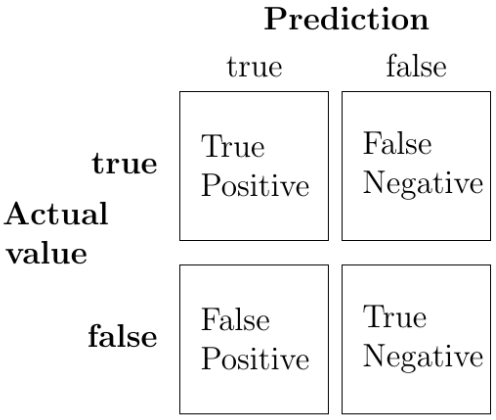


Figure 10.1: Confusion matrix

10.3.2 Distance error

The distance error takes into account the sequential ordering of the categories by executing higher penalties for estimations further off from the actual category. When a tweet should be classified as class 0 but is classified as class 2, it receives 2 penalty points. This is calculated for each class, and then averaged by the total amount of tweets in that class. The distance error is an approximation of the RMSE error, but still has an intuitive meaning as indicating how 'off' a classifier is in identifying the correct segment in the array of segments.

While the ultimate goal is to minimize the RMSE, the latter two evaluation measures can inform us whether there are substantial differences in the classification performance between the different classes. RMSE is the most relevant metric, since the target of the developed system is to be as accurate as possible in estimating the time-to-event. The question which classification system is best can only be valued by its effect on lowering RMSE on unseen data.

In this way an in-depth analysis is given of the performance of the classifiers, such that we understand the strengths and weaknesses of the system.

10.4 Baseline

A baseline provides a starting point from which a comparison of the performances of the classifiers can be made. Baselines are the most often forgotten component within design monitoring and evaluation [5], yet they are key to proving some improvement has actually been achieved. So we computed two baselines, called the '6-class' baseline and the 'informed' baseline.

The '6-class' baseline refers to the baseline strategy of labeling all tweets as belonging to class 6. The informed baseline consists of manually creating a set of words or expressions that are expected to be typical (temporal) words for specific classes. For example, all tweets that contain the word 'vanavond' (tonight), are classified as belonging to class 5 for both segmentation methods. The set of words we used is displayed in Table A.2. The tweets that did not contain a word of the manually selected set are given the label 'class 6'. The outcomes of these baselines will be displayed in the results of the main experiment.

Chapter 11

Software

We used different sources of software in order to do the classifications. We calculated the classes for the two categorizations (left branching and k-means) with R, a free software environment for statistical computing and graphics (see `http://www.r-project.org/`). To extract the uni-, bi- and trigrams, to transfer the tweets into a sparse binary format with the selected features, and to evaluate the classifications we used Python, a widely used general-purpose, high-level programming language (see `http://www.python.org`). For the feature selection we made use of TiMBL[1]. For the classification we used stimbl, which is a version of TiMBL that is faster and can better handle sparse vectors.

[1]The Tilburg Memory-Based Learner, see `http://ilk.uvt.nl/timbl/`

Part IV

Results

Chapter 12

Main experiment

Investigating what would be the best method to predict the time-to-event, we train and test tweets using different classifiers (SVM, k-NN, Naive Bayes), using different feature selection methods (Information Gain and χ^2) and using different clustering methods (k-means and left branching). In this chapter the results of the main experiment will be shown, using all available training data and 10,000 features to train with. First, the results of the left branching classification will be shown, then of the k-means, and then the performances of the different classification procedures will be compared.

Table 12.1: F1-scores of the left-branching categorization method. 'Avg.F1' reveals the 'average F1', M-F1 the MICRO-F1 and 'Inf. bl.' the 'informed baseline'. The small numbers represent the standard deviations.

		before							during	after	Avg.F1	M-F1
Alg.	FS	0	1	2	3	4	5	6	7	8		
k-nn	IG	.10 .03	.04 .02	.23 .08	.17 .02	.22 .02	.38 .02	.53 .02	.63 .03	.41 .02	.30 .02	.48 .02
k-nn	χ^2	.10 .04	.04 .03	.23 .08	.17 .03	.22 .02	.38 .02	.52 .02	.63 .03	.41 .03	.30 .02	.48 .02
NB	IG	.17 .04	.05 .02	.28 .10	.21 .02	.27 .04	.48 .02	.58 .01	.52 .02	.54 .03	.35 .02	.51 .02
NB	χ^2	.16 .04	.04 .03	.27 .11	.20 .02	.26 .04	.48 .02	.54 .01	.58 .02	.53 .03	.35 .03	.49 .02
SVM	IG	.21 .07	.05 .03	.36 .11	.25 .05	.38 .06	.50 .03	.60 .03	.72 .05	.56 .04	.40 .03	.56 .03
SVM	χ	.23 .07	.07 .03	.36 .10	.26 .05	.38 .03	.50 .03	.60 .02	.72 .03	.56 .03	.41 .03	.55 .02
6-class bl.		.00 .00	.00 .00	.00 .00	.00 .00	.00 .00	.00 .00	.41 .04	.00 .00	.00 .00	0.04 .00	.36 .04
Inf. bl.		.02 .02	.01.03	.40 .08	.32.05	.02 .01	.44 .03	.36 .04	.24 .02	.04 .01	0.21 .03	.39 .03

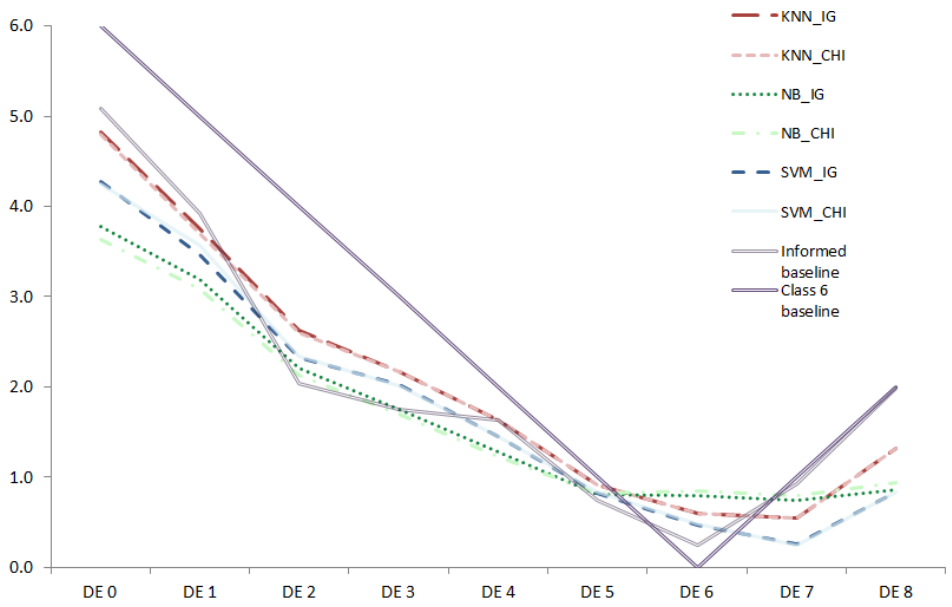


Figure 12.1: The distance errors for each class of the left-branching method. The x-as represents the classes, the y-as the distance errors.

12.1 Left branching

The F1-scores for each of the three classifiers k-NN (with $k = 1$), Naive Bayes and SVM for the left-branching categories are shown in Table 12.1. The results show that making the correct classification is harder if the event is further away, although categories 0 to 4 do not show a linear pattern, with the absolute worst performance for category 1. Especially class 2 has a high F1-score compared to the scores of class 1 and class 3, but the standard deviations of this class are high as well. The 'during' category has the highest F1-score (0.72), which is likely to be at least partly due to the fact that the majority of tweets is posted during the event. The performance of the different systems shows that SVM consistently achieves the highest F1-score for every category, while Naive Bayes outperforms k-NN for every category but 'during'. This pattern is also reflected in the MICRO-F1 score.

Compared to the informed baseline, the MICRO-F1 scores improved with an increase varying from 0.09 (k-NN) to 0.17 (SVM). Noteworthy, looking per class, in case of classes 2 and 3 (3 to 9 days before the event) the informed baseline has higher F1-scores. The best MICRO-F1 score is 0.56, which is not really high.

Taking a look at the distance errors (see Figure 12.1), we see that the classification is skewed to the 'during' class. Interestingly, the Naive Bayes classifier seems to be less skewed; this classifier has the best (lowest) distance errors for the first 6 classes, whereas for class 6, 7 and 8 they are the worst (highest). Not surprisingly,

all distance errors are improved compared to the 6-class baseline except for class 6. However, compared to the informed baseline not all classes improved. The lower distance errors for class 5 and 6 can be expected (because all instances that do not contain any of the selected features are assigned to class 6), but for class 2 and 3 they can not. The F1-scores of the informed baseline of these classes outperformed the other classifications as well. The words chosen for the informed baseline can apparently better recognize these two classes than the classification algorithms.

12.1.1 k-nearest neighbors

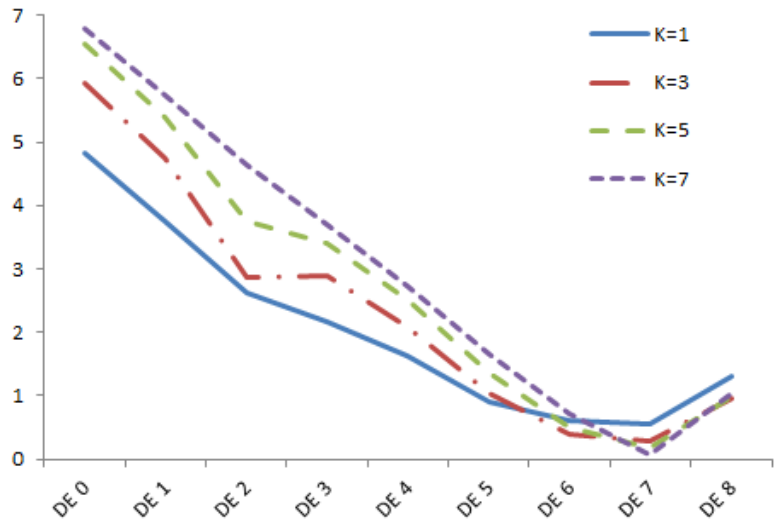


Figure 12.2: k-NN distance errors for $k = 1, 3, 5, 7$ applied to the left branching categorization method. The x-as represents the classes, the y-as the distance errors.

In terms of distances errors k-NN is again outperformed by SVM. In Section 6.2 we saw that the choice of k is important, so we increase this parameter in steps of 2 until $k = 7$. A plot of the average distance error for each k per category (Figure 12.2) shows that performance does not improve by increasing k. With higher k relatively more instances are classified as belonging to the majority 'during' category. For $k = 7$ there is almost a one-to-one mapping of the distance of each class to the 'during' class and the distance error. This tendency is also reflected in the F1-scores (see Table 12.2). For $k = 7$ the scores are below 0.07 for all 'before' classes, except the class closest to the event. $k = 3$ is compelling with $k = 1$, with a surprisingly high F1-score for class 2, but the F1-scores for class 0 and 1 are lower.

Table 12.2: F1-scores of the k-nearest neighbors classification, using the left-branching method and Information Gain as feature selection. 'Avg.F1' reveals the 'average F1', and M-F1 the MICRO-F1. The small numbers represent the standard deviations.

		before							during	after	Avg.F1	M-F1
	k	0	1	2	3	4	5	6	7	8		
k-NN	1	.10 .03	.04 .02	.23 .08	.17 .02	.22 .02	.38 .02	.53 .02	.63 .03	.41 .02	.30 .02	.48 .02
k-NN	3	.06 .03	.01 .01	.38 .12	.15 .04	.21 .04	.36 .03	.56 .02	.65 .04	.41 .03	.31 .02	.50 .03
k-NN	5	.00 .01	.00 .00	.28 .09	.08 .04	.08 .05	.20 .03	.49 .02	.61 .05	.28 .02	.23 .02	.40 .04
k-NN	7	.00 .00	.00 .00	.06 .03	.04 .03	.03 .05	.07 .02	.37 .03	.57 .05	.11 .02	.14 .01	.26 .03

Thus, increasing the parameter k does not improve the results of the k-nearest neighbors classifier. Moreover, it has the opposite effect. Although the MICRO-F1 score is a little bit higher for $k = 3$ than for $k = 1$, we decide to use $k = 1$ for the other experiments, because this method produced slightly better F1-scores for the first two classes and it produced better distance errors.

12.2 k-means clustering

Table 12.3: F1-scores of the k-means categorization method. 'Avg.F1' reveals the 'average F1', M-F1 the MICRO-F1 and 'Inf. bl.' the 'informed baseline'. The small numbers represent the standard deviations.

		before							during	after	Avg.F1	M-F1
Alg.	FS	0	1	2	3	4	5	6	7	8		
k-NN	IG	.11 .04	.23 .06	.20 .02	.18 .03	.34 .03	.33 .02	.48 .01	.63 .03	.41 .02	.32 .02	.44 .02
k-NN	χ^2	.11 .04	.23 .06	.20 .02	.18 .03	.34 .03	.33 .02	.48 .01	.63 .03	.41 .03	.32 .02	.44 .02
NB	IG	.21 .05	.27 .08	.24 .03	.21 .04	.45 .05	.45 .03	.49 .01	.59 .02	.54 .04	.38 .02	.47 .02
NB	χ^2	.21 .06	.27 .09	.24 .03	.21 .04	.44 .05	.45 .03	.48 .01	.58 .02	.53 .04	.38 .03	.46 .02
SVM	IG	.25 .06	.35 .07	.30 .03	.29 .03	.56 .05	.43 .03	.55 .02	.72 .03	.56 .03	.45 .02	.52 .02
SVM	χ^2	.25 .06	.35 .07	.30 .03	.30 .05	.57 .06	.43 .04	.55 .02	.72 .03	.56 .03	.45 .02	.52 .02
6-class bl.		.00 .00	.00 .00	.00 .00	.00 .00	.00 .00	.00 .00	.37 .04	.00 .00	.00 .00	.04 .00	.32 .03
Inf. bl.		.02 .02	.40 .07	.35 .07	.02 .01	.57 .05	.39 .05	.33 .04	.24 .02	.04 .01	.26 .04	.37 .02

The per-category F1-scores for k-means clustering are given in Table 12.3. Again we see that making the right classification is harder if the event is further

away, but this effect is less than for the left branching clustering; the F1-scores for the first 5 'before' classes are better, and for the last two a bit worse. There is no category that performs poorly, like it was the with class 1 in the left branching segmentation. Again, the MICRO-F1 and average F1-scores of the classification algorithms are always better than baseline performance, and again for two classes (class 1 and 2, which amounts to 11-3 days before the event) the informed baseline scored the best. The highest standard deviation is found in class 1.

Looking at the distance errors, we see again that Naive Bayes has the lowest scores for the classes furthest away from the event, and this time they are always better than the informed baseline. The other patterns are similar to the left branching method as well.

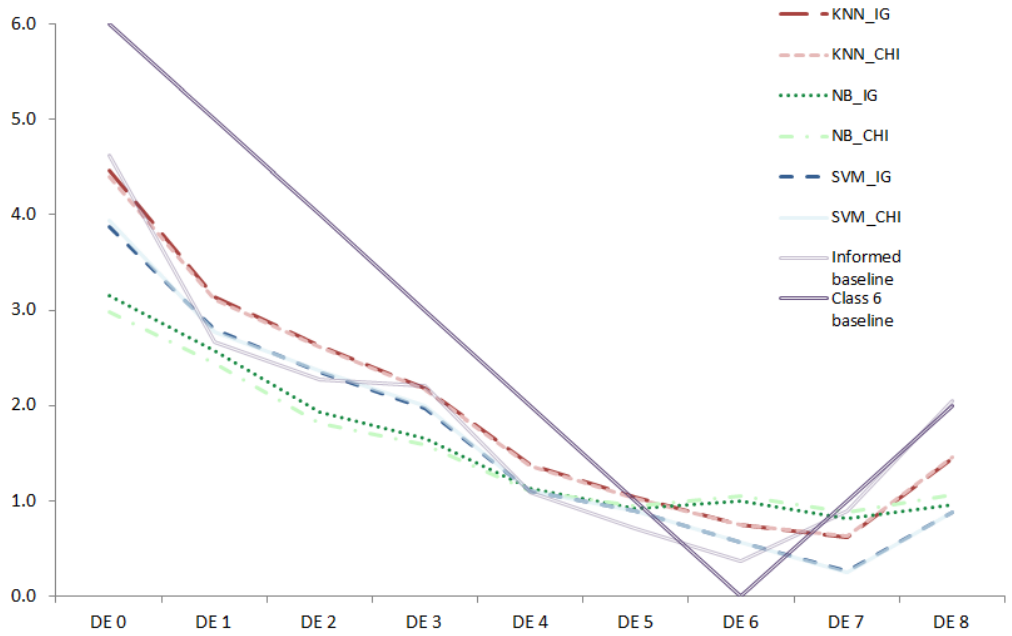


Figure 12.3: The distance errors for each class of the k-means method. The x-as represents the classes, the y-as the distance errors.

12.3 Comparison

Left branching vs. k-means To see how off the estimation of the time-to-event is and to compare the performance of the two segmentation systems, the RMSE[1]

[1]In Section 10.1 we described that the RMSE is computed using different numbers (representing the hours before event time) that represent the classes (the first, last, mean or median of each class). It appeared that using the last number (closest to event for each class) always produced the best RMSE, so only this RMSE will be reported.

Table 12.4: RMSE (hours) for the left branching and k-means segmentations. The responsiveness is given in brackets. Diff. is the difference in RMSE, calculated as left branching - k-means. For each RMSE score the Standard Deviation (σ) is calculated.

Alg.	FS	Left branching	σ	k-means	σ	Diff.
k-NN	IG	62.7 (.78)	7.1	60.5 (.77)	6.6	2.2
k-NN	χ^2	62.1 (.78)	6.8	60.6 (.77)	6.6	1.5
NB	IG	58.9 (.81)	5.2	56.6 (.81)	5.3	2.3
NB	χ^2	61.0 (**.82**)	4.8	58.4 (**.82**)	4.8	2.6
SVM	IG	54.8 (.75)	5.3	53.3 (.74)	5.3	1.5
SVM	χ^2	54.2 (.74)	6.0	**52.3** (.73)	6.4	0.9
6-class baseline		67.9 (1.00)	7.4	67.9 (1.00)	7.4	0.0
Informed baseline		61.3 (.89)	6.2	61.5 (.89)	6.3	0.2

and responsiveness are given in Table 12.4. The best scoring method has an RMSE of 52.3 hours, with a responsiveness of 73%. All RMSE scores of the classification algorithms are better than both baseline performances, except for the combination k-NN and left branching. In contrast, the responsiveness is significantly worse in all cases.

Comparing the two segmentation schemes we see that, while the responsiveness is a little bit higher for systems applied to the left-branching segmentation scheme, segmenting the 'before' classes by k-means clustering leads to a better RMSE for every system, achieving improvements of up to 2.6 hours. The differences between the two segmentation methods are highest for the Naive Bayes classifier and lowest for SVM.

The standard deviations are about 10 percent of the RMSE score and, although they are pretty high, performing a paired t-test reveals that the differences between the two segmentation methods are significant for all six classifications ($p < 0.01$).

Looking at the performances of the two classification methods at the level of identifying the right segment, we see that the left branching method consistently achieves better MICRO-F1 scores. This may seem odd, since the RMSE rates are consistently better in the case of k-means clustering. Taking a closer look at the F1-scores and distance errors per category reveals that this worse overall performance is due to the lower scores for the 'before' categories closest to event time and those categories contain the most instances. This can also be seen from the average F1 scores, which show the opposite effect. The better RMSE is reflected in the higher F1-scores and lower distance errors for the categories far before event time. There

is especially a high improvement in the k-means clustering for categories 1 and 4.

Classifiers The SVM classifier outperforms k-NN and Naive Bayes by several hours, with the lowest RMSE of 52.3. The responsiveness for SVM is the worst, however, indicating that this classifier has a higher tendency to over-predict the 'during' and 'after' categories. Naive Bayes (with χ^2) has the best responsiveness of 82%. k-nearest neighbors has a better responsiveness of 3 to 4 absolute percent compared to SVM, but the RMSE is increased by 7 or more hours, which leads to an RMSE higher than the informed baseline in case of left branching.

The standard deviation is highest for k-nearest neighbors, and lowest for Naive Bayes, for both segmentation methods. The k-NN classifier thus performs differently across the 10 folds. Compared to the 6-class baseline, the RMSE scores of the six classifications are significantly different ($p < 0.01$). However, compared to the informed baseline, only the RMSE scores of SVM (in all cases) and of Naive Bayes with k-means clustering and Information Gain show a significant difference ($p < 0.01$). For the other classifications the paired t-test revealed that $p > 0.1$, except for Naive Bayes with k-means clustering and χ^2 ($p = 0.07$). Compared to both baselines all responsiveness results are significantly worse ($p < 0.01$).

Table 12.5: RMSE (hours) for the k-nearest neighbors classifier using left branching and Information Gain. The responsiveness is given in brackets.

Alg.	k	RMSE	σ
k-NN	1	62.7 (.78)	7.5
k-NN	3	55.8 (.69)	7.3
k-NN	5	56.1 (.51)	7.2
k-NN	7	55.1 (.30)	8.8

Looking at the RMSE by increasing k for k-NN in Table 12.5, we see a decreasing RMSE compared to $k = 1$, but this is greatly a the expense of the responsiveness, which is only 30% for $k = 7$. This affirms our choice for $k = 1$.

Per segment, the F1-scores of SVM are always the best, for both left branching and k-means. Naive Bayes scores better than k-NN for all 'before' classes. The distance errors show that for both segmentation methods Naive Bayes has the lowest errors for the first 5 'before' classes, and then SVM takes over.

Feature selection methods Alternating the Information Gain and χ^2 feature selection method is most influential in the case of Naive Bayes, and significant

with $p < 0.01$. The RMSE of Information Gain is 1.1 (left branching) and 1.8 (k-means) hours better than the RMSE of χ^2, with only a 0.01 worse responsiveness. However, the distance errors of χ^2 are slightly lower for the first 5 classes. The differences of the two feature selection methods are negligible and not significant ($p > 0.1$) for the other classifiers, both for k-means and left branching.

Development of RMSE over time For the combination Naive Bayes and Information Gain and for the combination SVM and χ^2 for k-means clustering, the RMSE per hour[2] is plotted in Figure 12.4. In this graph the development of the RMSE over time can be read, and we indeed see that the RMSE of events further away is much higher. Besides that, we see that Naive Bayes has a tendency to have lower RMSE than SVM for tweets further away from the event.

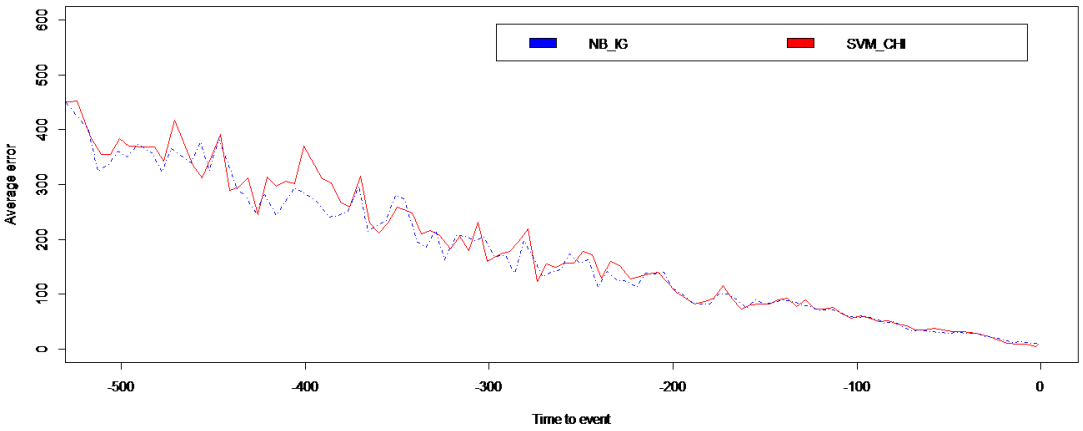


Figure 12.4: RMSE per hour for the k-means segmentation method. The x-as reveals the hour that the tweet should be classified as.

12.4 Conclusions

In sum, this experiment shows that it is hard to predict an event that is still far away; the best prediction was 52.3 hours off. There is a tendency to classify tweets with the most frequent 'during' class, which is reflected in the increasing distance error for tweets posted longer before the event. The standard deviation appeared to be around 6 hours, which reveals that performance across the 10 different folds is not consistent, especially for k-NN.

[2]The average of every 5 hours is taken.

The method of time segmentation and the chosen algorithm is important. We saw that the categorization using the k-means method produces better results than the left branching method. This can be explained by the fact that the clustering for the k-means method takes into account content-based characteristics of event-related tweets over time. In contrast, the left branching clustering assumes some mathematical distribution of occurrences, by dividing the clusters in two. Looking at the clusters in Table 8.1, we see therefore a, for humans, more natural clustering scheme for the k-means clustering than for the left branching. For example, class 1 for left branching has the 12-9 days before the event, and for k-means the 11-6 days. Given that the football matches are played on Friday, Saturday or Sunday, the '11-6' category contains roughly the class for which the event will take place 'next week'. The '12-9' category, however, partly contains tweets for which the event will take place 'next week', but class 2 ('9-6' days) has some tweets of this sort as well. In sum, there is no explicit temporal word for this relatively small class, that is far away of the event. Something similar shows up if you compare class 5 (5 to 31 hours before event) of left branching, with class 4 and 5 (33-13 and 12-4 hours before event) of the k-means clustering. Actually, all classes of the k-means clustering algorithm are pretty natural divisions to humans, whereas this is thus not the case for the left branching scheme.

Investigating the different classifiers we see that k-nearest neighbors has an overall worse performance. SVM has lowest RMSE and best MICRO-F1 scores, but Naive Bayes has a better responsiveness. Furthermore, the distance errors and RMSE per hour shows that Naive Bayes has better results for the classes that contain tweets that are furthest away of the event. Since we ware most interested in these tweets, Naive Bayes should definitely not be ignored. The different feature selection methods do not really differ from each other.

Chapter 13

Using less data

In this chapter, we explore the results of the classification using less data, i.e. using 1) less data instances and 2) less features. The first investigation is interesting for real-world situations where there is less data available. The second is interesting because, as explained in Section 9, the choice of features for the classification is very important , and it is not always the case that more means better. On top of that, using less data generally will make the training and classification faster.

13.1 Using less instances

13.1.1 Using one third of the training set

In this part we investigate how results are influenced using only one third of the data set to train on. We choose to randomly select a third of the training data set, which gives us a set of 135,506 tweets. Then we repeat all the classifications that were done in the previous section. To avoid an overload of data we will now only give a short description of the evaluation in terms of F1-scores and distance errors.

Left branching The F1-scores for the left branching segmentation show small declines (most below 0.03) for each class and classification method. The MICRO-F1 scores only dropped by 0.01 or 0.02. Looking at distance errors we again see minor differences, especially for the three classes closest to the event and the 'during' class (classes 4-7). Taking a closer look per classifier, we see that Naive Bayes is mostly affected; where training with all data leads to the best distance

errors for the first 5 classes for Naive Bayes in comparison to the other classifiers, this difference is now almost gone.

k-means Investigating the F1-scores of the k-means segmentation, we find a similar pattern. Looking at distance errors we see again that the scores of Naive Bayes dropped the most, but only for the first two classes and the differences are less. k-NN and SVM both show minor differences.

RMSE

The RMSE is given in Table 13.1. Comparing this with the RMSE using all instances, we see that the performance is a bit worse, with on average a decline of the RMSE of 1.9 hours. There is not much difference between the two segmentation methods, on average k-means is effected more by just 0.2 hours. Of the three classifiers, k-NN is most effected. In contrast, the combination Naive Bayes and Information Gain is hardly effected. Table 13.2 reveals that the differences classifying with k-NN are significant in all cases, with Naive Bayes only for k-means clustering and with SVM only for Information Gain. Again, the best RMSE is gained using k-means as categorization, using χ^2 and SVM to train and the best responsiveness is gained using Naive Bayes and χ^2.

The responsiveness is not really influenced by the smaller data set, with differences below 0.03, but some of the differences are significant (see Table 13.2). We see that the standard deviations, compared to training with the whole data set, are larger for Naive Bayes and SVM, but lower for k-NN.

Table 13.1: RMSE (hours) for the left branching and k-means segmentations, using all, one third and $\frac{1}{30}^{th}$ of the training data. The responsiveness is given in brackets.

		left branching						k-means			
		all		$\frac{1}{3}^{th}$		$\frac{1}{30}^{th}$		all		$\frac{1}{3}^{th}$	
Alg.	FS	RMSE	σ	RMSE	σ	RMSE	σ	RMSE	σ	RMSE	σ
k-NN	IG	62.7 (.78)	7.1	64.4 (.77)	6.8	67.9 (.75)	6.1	60.5 (.77)	6.6	62.7 (.76)	6.3
k-NN	χ^2	62.1 (.78)	6.8	65.0 (.77)	6.3	68.3 (**.76**)	6.3	60.6 (.77)	6.6	62.9 (.77)	6.3
NB	IG	58.9 (.81)	5.2	59.3 (.80)	6.0	61.6 (.75)	6.6	56.6 (.81)	5.3	57.7 (.79)	5.7
NB	χ^2	61.0 (**.82**)	4.8	63.2 (**.81**)	6.1	62.5 (**.76**)	6.3	58.4 (**.82**)	4.8	60.8 (**.81**)	5.9
SVM	IG	54.8 (.75)	5.3	56.6 (.75)	6.0	57.1 (.72)	7.3	53.3 (.74)	5.3	55.3 (.74)	6.4
SVM	χ^2	**54.2** (.74)	6.0	**55.7** (.74)	6.3	**56.4** (.71)	7.3	**52.3** (.73)	6.4	**54.2** (.73)	6.4

Table 13.2: Results of the two tailed paired t-test, comparing RMSE and responsiveness (in brackets) using the whole data set with using $\frac{1}{3}^{th}$ or $\frac{1}{30}^{th}$ of the data. If $p > 0.05$, results are shown in bold.

		Left branching				k-means	
		$\frac{1}{3}^{th}$		$\frac{1}{30}^{th}$		$\frac{1}{3}^{th}$	
k-NN	IG	.000	(.000)	.000	(.000)	.000	(.000)
k-NN	χ^2	.000	(.000)	.000	(.000)	.000	(.001)
NB	IG	**.819**	(.000)	**.094**	(.000)	.044	(.000)
NB	χ^2	**.127**	(**.080**)	**.558**	(.000)	.010	(**.286**)
SVM	IG	.013	(.056)	**.097**	(.000)	.009	(.002)
SVM	χ^2	**.173**	(**.305**)	**.055**	(.000)	**.141**	(**.059**)

Summary As expected, the use of only one third of the training data worsened the performance, but not very dramatically, and not all differences were significant. The RMSE dropped between 0.4 and 2.9 hours, and the responsiveness was not greatly affected. Looking at distance errors and F1-scores, Naive Bayes was mostly affected for the first few classes.

13.1.2 Using only one thirtieth

To know how well the classifiers would be perform using only a small subset of the training data, we did a short experiment for the left branching categorization using only $\frac{1}{30}^{th}$ of the training data (13,550 instances). The RMSE scores are shown in Table 13.1, and we see that now results are getting worse; the RMSE dropped by a couple of hours, but the difference, except for k-nearest neighbors, is not statistically significant (see Table 13.2). The standard deviations are larger as well, so the differences in performance between the folds are pretty high. The responsiveness is significantly affected, by 0.02 to 0.06 percent. The F1-scores and distance error show a similar effect; the F1-scores and distance errors dropped 3 to 4 times as much as they did going from all data to one third.

13.1.3 Conclusions

It can be concluded that using one third of the training data that was available in this experiment, is sufficient. The additional 2/3 of the training data leads to improvement, but not a dramatic one, and the improvements are not always significant. The differences are rather small (0.4 to 2.9 hours) and they did not contain any spikes. The responsiveness hardly increases using more data. On

average k-NN benefits the most of using more data.

Comparing the classification times (see Table 13.3), we see that it takes more than 10 times as long to train the SVM classifier with the complete data set than with $\frac{1}{3}$, and more than 5 times for k-nearest neighbors. So in experiments where a small amount of data or less time is available, this classification is possible without any large consequences.

Using a very small training set (of about 15,000 instances) has a greater impact on the scores. The RMSE scores are 1.5 to 6 hours worse, and we see a significant decline in responsiveness.

The classification time of SVM drops to 1 minute, SVM shows the smallest declines and still scored better than baseline. So for fast experiments or to have an indication about the classification using the SVM classifier this could be an option.

Table 13.3: Classification times (in hours) of the three classifiers. The percentages denote the percentage of the classification time of the main experiment.

		SVM		KNN		NB
# features	# instances	Time	Percentage	Time	Percentage	Time
10,000	all	13:19	100%	1:27	100%	<0:01
1,000	all	5:18	39.8%	0:48	55.2%	<0:01
100	all	0:54	6.8%	0:20	23.0%	<0:01
10,000	1/3	1:08	8.5%	0:17	19.5%	<0:01
10,000	1/30	0:01	.1%	0:01	1.1%	<0:01

13.2 Using less features

It seems reasonable that the performance of the system will decrease using less features to train with, like it was the case in the previous section using less training instances. However, as explained in Chapter 5, using more features does not always entails a better classification, due to overfitting. In the next section we will see how the classification is influenced using only 1,000 or even 100 features to train with. Again, we only give a short description of the evaluation in terms of F1-scores and distance errors. We first provide the results of these evaluation measures for left branching and k-means clustering for both 1,000 and 100 features, and then compare the RMSE scores.

13.2.1 Left branching

1,000 features Investigating the F1-scores using 1,000 features to train with, reveals that the differences with the main experiment are somewhat more capricious then the differences were using less training instances. The scores now even improve for some classes, namely for the classes 2 and 4. Still, the changes are not very big. Comparing classifiers we see that the classification using SVM and χ^2 has a larger decline than the classification using the other algorithms and feature selection methods, which is reflected in a decrease in MICRO-F1 of 0.09. k-NN seems to be least effected by the decrease in features.

Looking at distance errors, the errors of classes that are furthest away grew the most, up to differences of 0.7. The k-NN classifier is hardly affected, whereas SVM now follows a path more similar to that of k-nearest neighbors. The classification using the Naive Bayes algorithm has again lowest distance errors for the classes furthest away from the event. Remarkably, where the performances of χ^2 and Information Gain were very similar for all classifiers using 10,000 features, they are now more disassociating from each other, with χ^2 performing slightly better.

100 features In terms of F1-scores we see larger differences, up to declines of 0.29 and even some improvements of up to 0.12. Again, k-NN is least influenced by the decrease in features, and the results for SVM generally decreased the most. Looking per class, we see that the F1-scores of class 1, 3 and the 'during' class dropped a lot and, in contrast, the scores of class 2 and 4 improved. The improvements and losses of the k-NN and Naive Bayes algorithms are averaging, which is reflected in the MICRO-F1 scores, whereas the MICRO-F1 score of the SVM algorithm is a lot worse.

The first thing to notice when looking at the distance errors is that the slope is much more steeper from class 0 to class 4, indicating that the classifiers are heavily influenced by the large 'during' class. Furthermore, looking at the first classes, we see that the differences between the algorithms are very small, but this time differences between the feature selection methods have emerged. The χ^2 feature selection method has lower distance errors. This is in contrast with the distance errors of the classification with 10,000 features, where the differences between the distance errors of the classifiers were large, and not the differences between the features selection methods.

13.2.2 k-means

1,000 features Exploring the F1-scores, we see improvements of up to 0.07 for class 1 and 4. Again, the SVM classifier has largest drops, where for the

other classifiers most of the F1-scores only changed by 0.01 (both positively and negatively). The distance errors once again show that the performance of the SVM classifier degrades to the performance of the k-NN classifier.

100 features The F1-scores of the k-means categorization show again, even bigger, improvements for class 1 and class 4, whereas the F1-scores of class 0, 6 and the 'after' class decreased a lot. The SVM classifier has the worst scores for each class compared to its performance with 10,000 features, which is mirrored in the low MICRO-F1 scores.

The graph of the distance errors of the k-means segmentation shows similar effects as the graph of the left branching segmentation.

So, in contrast with the decrease in training instances, a decrease in features does not lead to a consistent worse performance for all classes.

13.2.3 RMSE

The RMSE in hours for the left branching and k-means segmentations using $10,000$, $1,000$ and 100 features are given in Table 13.4. Comparing $10,000$ features with $1,000$ for the left branching method, we see that the RMSE is improved, especially for Naive Bayes, for which the improvements are significant (see Table 13.5).

However, the responsiveness is now worse in all cases, and this difference is significant ($p < 0.01$ in all cases). Especially the responsiveness for SVM decreases a lot. The standard deviation scores are lower for k-NN, higher for SVM with Information Gain as feature selection method, and for the other classification methods it stayed almost the same.

Doing the same for the k-means method, we see a similar pattern. All RMSE scores are again a bit lower than the RMSE scores of the main experiment, but for k-NN and SVM with Information Gain this difference is not significant. For k-NN responsiveness is least effected, whereas for the SVM classifier the responsiveness of the SVM classifier is very negatively influenced by the decrease in features. All differences in responsiveness are significant.

Using 100 features lowers the RMSE compared to the main experiment. However, this is at the expense of a big drop in responsiveness. For k-NN they drop with around 0.1, and for Naive Bayes and SVM they even drop with around 0.2. Where classification using the features selected by the χ^2 algorithm often was the best using 10,000 features, we now see that Information Gain has the best RMSE scores. This can be surprising, because the distance errors were better for χ^2, but it can be explained by the fact that χ^2 has a better responsiveness.

Table 13.4: RMSE (hours) for the left branching and k-means segmentations, given for 100, 1,000 and 10,000 features. The responsiveness is given in brackets. Each second row reveals the standard deviations.

		Left branching			k-means		
Alg.	FS	10,000	1,000	100	10,000	1,000	100
k-NN	IG	62.7 (.78)	60.4 (.77)	56.5 (**.67**)	60.5 (.77)	59.4 (**.76**)	54.0 (.62)
σ		7.1	5.2	5.3	6.6	5.3	5.2
k-NN	χ^2	62.1 (.78)	60.0 (.76)	57.2 (.66)	60.6 (.77)	58.3 (.75)	54.3 (.60)
σ		6.8	5.9	5.2	6.6	5.6	5.4
NB	IG	58.9 (.81)	55.9 (**.78**)	53.8 (**.67**)	56.6 (.81)	54.0 (**.76**)	50.7 (**.64**)
σ		5.2	5.6	6.1	5.3	5.3	4.6
NB	χ^2	61.0 (**.82**)	57.2 (**.78**)	59.2 (.63)	58.4 (**.82**)	54.2 (**.76**)	51.3 (.62)
σ		4.8	4.9	5.7	4.8	5.0	6.0
SVM	IG	54.8 (.75)	53.8 (.66)	**50.6** (.53)	53.3 (.74)	51.4 (.64)	**49.6** (.53)
σ		5.3	6.7	6.3	5.3	7.0	6.7
SVM	χ^2	**54.2** (.74)	**52.3** (.64)	54.3 (.53)	**52.3** (.73)	**50.3** (.63)	51.1 (.51)
σ		6.0	6.0	6.0	6.4	5.9	7.9

Table 13.5: Results of the two tailed paired t-test, comparing the RMSE using 10,000 features to train with 1,000 and 100 features to train. If $p > 0.05$, results are shown in bold. The results of the t-test for responsiveness show that $p < 0.01$ in all cases.

		Left branching		k-means	
		1000	100	1000	100
k-NN	IG	**0.158**	0.002	**0.124**	0.000
k-NN	χ^2	0.015	0.002	0.009	0.000
NB	IG	0.002	0.005	0.002	0.029
NB	χ^2	0.002	0.014	0.000	**0.060**
SVM	IG	**0.270**	0.020	**0.105**	0.038
SVM	χ^2	**0.217**	**0.734**	0.036	**0.442**

The standard deviations for SVM and Naive Bayes are on average 1 to 2 hours higher, whereas they are 2 to 5 hours lower for k-NN, and for this classifier all differences are significant with $p < 0.01$. For SVM with χ^2 the differences are not significant ($p > 0.1$). Apparently the SVM with χ^2 classification is not very consistent across the folds.

13.2.4 Conclusions

Training with less features does not result in an *overall* worse performance, like it was the case in the experiment where less data instances were used. The F1-scores of some classes improved, of some lowered and of some stayed almost the same. Often the RMSE score improved by a few hours, whereas the responsiveness lowered, especially in cases of 100 features.

The choice of the feature selection method becomes more important; Information Gain produces better RMSE scores and responsiveness.

The differences in standard deviations reveal that for k-NN the folds are performing more consistent using less features, whereas for the other two classification algorithms the performance is evidently generally less consistent and showing higher standard deviations.

The influences of the decrease in features is different for the three classifiers. For k-nearest neighbors it seems better to use 1,000 than to use 10,000 features, especially bearing in mind the reduction in classification times (see Table 13.3). In contrast, the SVM classifier produces very bad results (even if you bear in mind that the classification time is reduced by more than 50 percent). Since the classification times of Naive Bayes are very low in all cases, we do not have to take them into account. Looking at responsiveness it is always better to use 10,000 features. However, except for the combination left branching and χ^2, all RMSE scores are better with smaller numbers of features. For Naive Bayes one thus has to make a well-considered decision if one plans to use less features.

13.2.5 Clarification lower RMSE 100 features

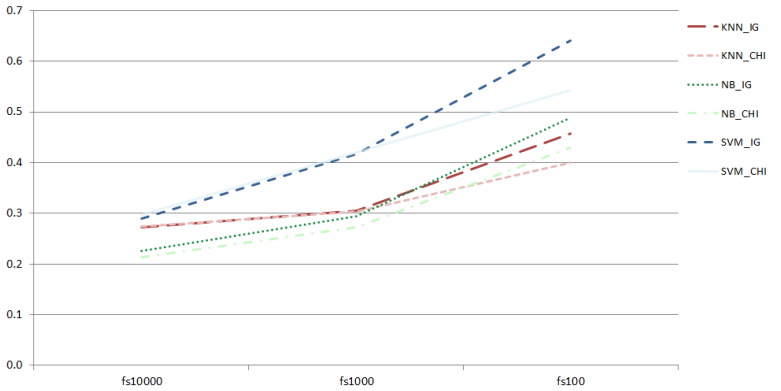
Even with the decrease in responsiveness it still seems odd that the RMSE is decreased using less features. Is it the case that, if we could improve the responsiveness (for example by hierarchical classification, see next chapter), the system will produce better results in cases of training with less features? To answer this question, we take a closer look at the classifications. For left branching, we calculate the proportion of tweets in class 0 that is classified as 'during' or 'after', and we calculate the same proportion for the classes 1 to 6 taken together. For the k-means clustering we do the same, except that we take the proportion of class 0 and 1, and of 2-6 (because class 0 is smaller for k-means clustering). These will

be called 'class Zero' and 'class One' respectively (for both segmentation methods), and the results can be found in Figures 13.1(a), 13.1(b) (left branching) and 13.1(c) and 13.1(d) (k-means). The first thing which strikes one is that generally the error of class Zero grows faster than the error of class One. This is already an explanation for the decrease in RMSE: we saw that on average the RMSE of the first classes is much higher than the RMSE of the classes closest to event (see Figure 12.4). So, if relatively more instances that belong to the classes furthest away of the event are classified as 'during' and 'after', and thus not included in the calculation of the RMSE, the RMSE will decrease automatically. Since we are most interested in the classes that are 1 or 2 weeks away of the event, this lower RMSE is not very satisfying.

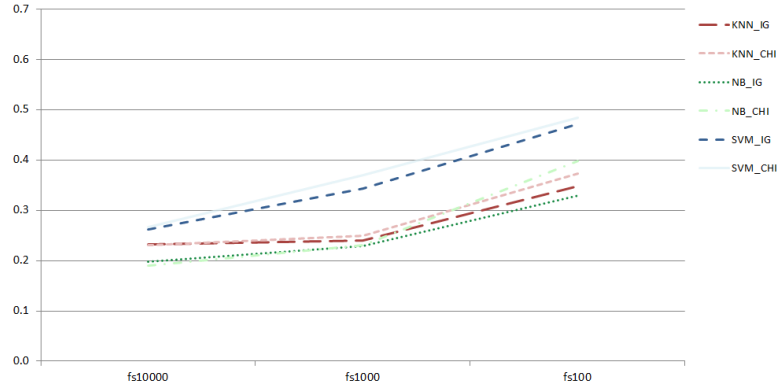
The fact that Information Gain has lower RMSE in all cases can be explained by these graphs as well: we see that for class Zero, for each classifier the amount of cases that is classified as 'during' or 'after' is higher for Information Gain than for χ^2, whereas for class One it is the opposite, except for k-NN. But there we see that the difference in RMSE between the two feature selection methods is only 0.3.

Moreover, taking a closer look at the Naive Bayes classification for the left branching segmentation and using 100 features, we see, in the case of using Information Gain as feature selection method, that more instances are classified as 'during' or 'after' than in the case of using χ^2 for class Zero, whereas for class One we see an opposite effect. If we now compare the RMSE scores, we see that the RMSE using Information Gain is much lower than the RMSE of χ^2 (53.8 vs 59.2). A similar pattern is found generally: in cases were relatively more instances that belong to class Zero are classified as 'during' or 'after' and relatively less instances that belong to class One, the RMSE improves.

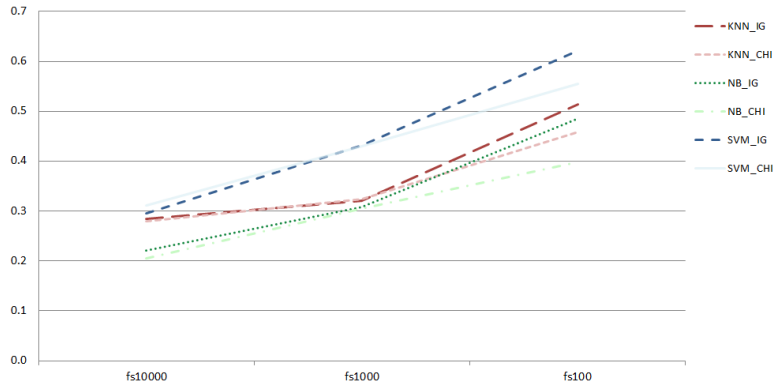
In sum, it does not seem to be the case that in case of using only 100 features, once instances are classified as 'before', the classification will be better. Instead, the instances that are harder to predict, in other words the instances that had high RMSE in the main experiment, are dismissed.



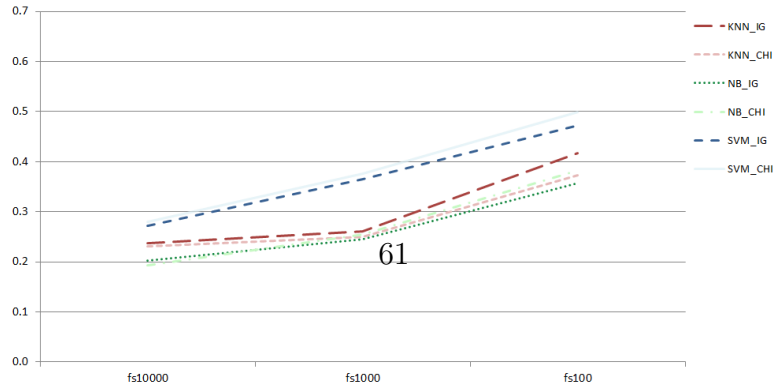
(a) Left branching, class Zero



(b) Left branching, class One



(c) k-means, class Zero



(d) k-means, class One

Figure 13.1: Proportion of tweets that is classified as ‘during’ and ‘after’

Chapter 14

Hierarchical classification

One of the major issues of the classification is the large 'during' class, which biases all classifiers. Hierarchical classification might offer a solution, where we first use a classifier that separates 'before' tweets from 'during' and 'after' tweets, and then further classify these 'before' tweets. The obvious advantage is that during the second step the large 'during' class cannot influence the results anymore. The disadvantage is that tweets that are wrongly classified in the first layer of classification as 'during' or 'after' are discarded, and some tweets will be wrongly classified as 'before' in this first step. Another disadvantage of hierarchical classification is that it takes more effort. To restrict the amount of data, we confined ourselves to the k-means segmentation, using only the Naive Bayes and SVM classifier, since these showed the best results in the main experiment.

14.1 First layer of classification

Table 14.1: Responsiveness and F1-scores of the first step of hierarchical classification.

		Responsiveness		F1-score	
Alg.	FS	before	during & after	before	during & after
NB	IG	.83	.76	.78	.79
NB	χ^2	.83	.75	.78	.79
SVM	IG	.75	.86	.78	.83
SVM	χ^2	.75	.86	.78	.83

The responsiveness and F1-scores of this classification are given in Table 14.1. Looking at responsiveness, we see that Naive Bayes performed better for the 'before' class, and SVM for the other class. Looking at F1-scores there is no difference between the two classifiers for the 'during' class, but SVM has the best scores of 0.83 for the 'during' and 'after' class. The differences between the two feature selection methods are negligible.

14.2 Second layer of classification

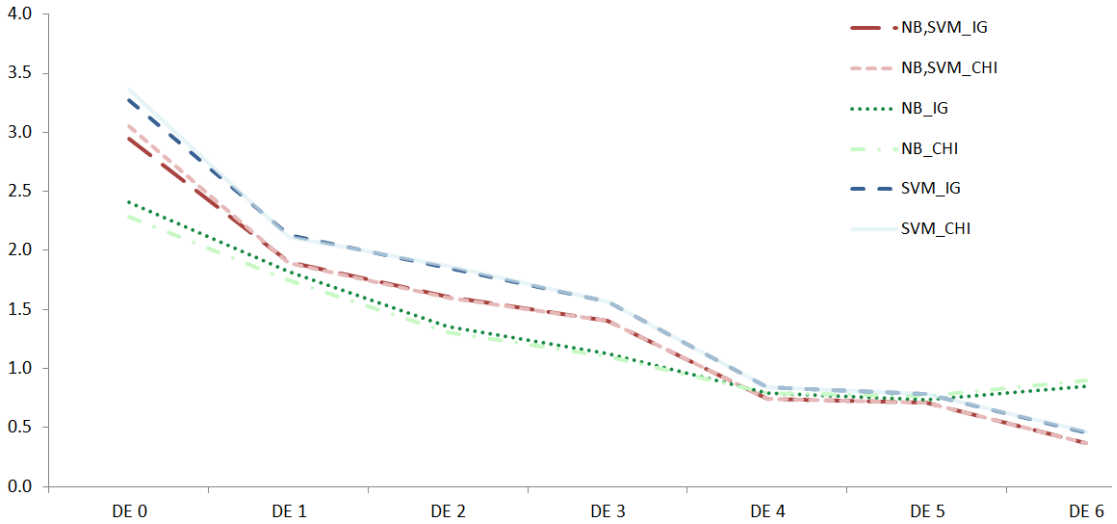


Figure 14.1: Distance errors of the hierarchical classification.

After the first step we further classify the 'before' class into the (to us) familiar seven categories. Besides using the same classifier for each of the two layers, we also investigate how the performance would be using Naive Bayes for the first layer and SVM for the second layer. The responsiveness of Naive Bayes was highest in the previous experiments, and the SVM classifier had best RMSE scores. So in this way we hope to combine the advantages of the two classification algorithms[1] which we write as 'NB,SVM'. The F1-scores are given in Table 14.2. All F1-scores of the before categories improved slightly (between 0.02 and 0.06), compared to their equivalent of the main experiment.

We see in Figure 15.1 that the distance errors reduced compared to the distance errors of the main experiment, and that the improvements were biggest for class 0 and class 1 (0.6 - 0.7). For the other classes, the improvement decreases

[1]We also did this the other way around, and indeed the scores (especially responsiveness) were worse compared to using the same classifier for both layers, and will therefore not be presented.

Table 14.2: F1-scores of the second step in hierarchical classification. The small numbers reveal the standard deviations.

		before							during	after	
		0	1	2	3	4	5	6	7	8	M-F1
NB	IG	0.24 .06	0.32 .07	0.27 .02	0.24 .05	0.48 .05	0.47 .04	0.54 .02	0.10 .01	0.34 .03	0.46 .01
NB	χ^2	0.24 .07	0.31 .08	0.27 .03	0.24 .04	0.48 .05	0.47 .03	0.54 .02	0.10 .02	0.33 .02	0.46 .01
SVM	IG	0.28 .07	0.40 .06	0.33 .03	0.32 .03	0.61 .05	0.46 .04	0.61 .02	0.52 .03	0.38 .04	0.53 .02
SVM	χ^2	0.28 .07	0.40 .07	0.33 .03	0.33 .05	0.61 .06	0.46 .04	0.61 .02	0.52 .03	0.38 .03	0.53 .02
NB, SVM	IG	0.29 .07	0.41 .07	0.35 .03	0.34 .04	0.62 .05	0.47 .04	0.65 .02	0.29 .03	0.31 .02	0.56 .02
NB, SVM	χ^2	0.28 .07	0.41 .07	0.35 .03	0.34 .05	0.63 .06	0.47 .04	0.65 .02	0.30 .03	0.29 .02	0.56 .02

continuously from class 2 in steps of about 0.15 to 0.1 for the last 'before' class. We again see that Naive Bayes has the best distance errors for the first 5 classes. For class 4 and 5 they are similar for all the classifications, and then the NB,SVM path has the best scores.

Looking at the RMSE for the NB and SVM classification, we see that all RMSE scores improved by approximately 1 hour compared to their equivalents in the main experiment, but that the responsiveness decreased by 0.4%, both significantly ($p < 0.01$). However, the standard deviations are a bit lower in the main experiment.

For the hierarchical classification using both Naive Bayes and SVM, we see almost the same scores as in the case training with SVM in both layers. Comparing these RMSE scores to the SVM and Naive Bayes scores of the main experiment, reveals that (although the differences are rather small) they are significant in all cases, except comparing with SVM in case of Information Gain ($p = 0.119$). The differences in responsiveness are always significant.

Table 14.3: RMSE and responsiveness for the hierarchical classification. Responsiveness is given in brackets.

Algorithm	FS	RMSE k-means	σ	RMSE k-means main experiment	σ
NB	IG	55.5 (.78)	5.5	56.6 (.81)	5.3
NB	χ^2	57.3 (.78)	5.1	58.4 (.82)	4.8
SVM	IG	52.5 (.70)	5.6	53.3 (.74)	5.3
SVM	χ^2	51.4 (.69)	6.6	52.3 (.73)	6.4
NB, SVM	IG	52.0 (.70)	6.0		
NB, SVM	χ^2	51.7 (.70)	6.4		

14.3 Conclusion

For the F1-scores and especially the distance errors of the 'before' classes, the results were better. In terms of RMSE and responsiveness, the hierarchical classification did not make great improvements. The RMSE was somewhat better, but the results of the responsiveness were slightly worse. The scores of the responsiveness are mostly decided in the first layer of classification. If we can improve this layer of classification, then it might become useful to do the hierarchical classification.

Chapter 15

Characters as features

Unfortunately, the hierarchical classification did not lead to great improvements. So we start to think about other changes that we could make to the classification, that possibly could improve the classifications. In Twitter messages the use of acronyms, emoticons and misspellings is ubiquitous, so maybe we should not choose words as features. Instead, we could utilize character-based n-grams [20]. In order to do this, we extract all 2-,3-,4- and 5-grams that appear more than 50 times, which gives us 81,444 features. Then we apply the same procedure as with the 'normal' features, using the k-means segmentation with χ^2, and classifying with all three algorithms.

15.1 Results

Sadly, the training with the SVM classifier took over a month, and then we stopped it. Given that the other classifications never took more than 2 days, this was really surprising. Apparently, the SVM classifier could not find support vectors that could properly distinguish the classes. These problems did not arise for Naive Bayes and k-nearest neighbors.

15.1.1 F1-score and distance error

The F1-scores of the Naive Bayes and k-nearest neighbors classification are displayed in Table 15.1. The scores are worse for all classes (except for the 'during' class with k-NN). Moreover, the results for the 'during' and 'after' class are really bad in case of the Naive Bayes classifier. The distance errors for the k-nearest

neighbors classification display a similar result as in the main experiment. However, the distance errors for Naive Bayes are totally different; we no longer see the big influence of the 'during' class. Instead, all distance errors are around 2 for the classes at the edges, and around 1 for the middle classes.

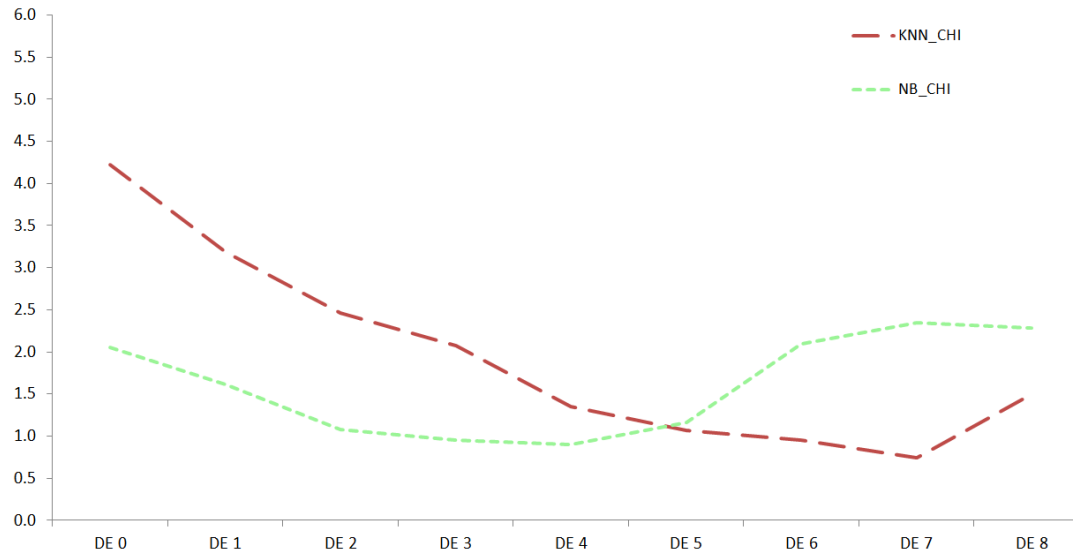


Figure 15.1: Distance errors of the classification using characters as features.

Table 15.1: F1-scores training with characters as features. The first and third row reveal the scores using characters as features, the second and fourth row the scores using words as features (as in the main experiment). The small numbers reveal the standard deviation.

		before							during	after	M-F1
Alg.	FS	0	1	2	3	4	5	6	7	8	
NB	χ^2	.10 .04	.17 .07	.11 .02	.09 .02	.40 .05	.41 .04	.23 .02	.12 .01	.29 .04	.25 .01
NB	χ^2	.21 .06	.27 .09	.24 .03	.21 .04	.44 .05	.45 .03	.48 .01	.58 .02	.53 .04	.46 .02
k-NN	χ^2	.10 .04	.18 .05	.16 .03	.16 .03	.28 .04	.30 .03	.40 .01	.68 .02	.32 .02	.42 .02
k-NN	χ^2	.11 .04	.23 .06	.20 .02	.18 .03	.34 .03	.33 .02	.48 .01	.63 .03	.41 .03	.44 .02

15.1.2 RMSE

The RMSE score for k-nearest neighbors actually improved, with only a small, but significant, decrease in responsiveness. However, the improvement is not significantly different ($p = 0.380$). In contrast, the responsiveness of the Naive Bayes classifier (significantly) increased to 0.89, bringing along a (significant) increase in the RMSE of 15.6, more than half a day.

Table 15.2: RMSE of the k-means segmentation with χ^2 as feature selection method, using characters as features.

Alg.	RMSE	σ	RMSE main experiment	σ
k-NN	57.3 (.75)	8.7	60.6 (.77)	6.6
NB	74.0 (.89)	6.3	58.4 (.82)	4.8

Adding up these results we see that k-nearest neighbors performs almost the same as in the main experiment, even having a slightly better RMSE, but this difference is not significant and the standard deviation is higher. In contrast, Naive Bayes performed totally different. The classifier is not skewed by the 'during' class anymore, instead it seems to have distributed the tweets approximately equally between the classes. Unfortunately, this did not results in a better RMSE, but in a much worse RMSE (74.0), although the responsiveness is higher (89%).

How can the scores be explained? The k-NN classifier finds the nearest neighbors of a test instances. Using characters as features, each instance will have more features. It could be the case that in this way k-NN classification can be improved, because better prototypes can be find in the training data.

On the other hand, because each tweet has so many overlapping features, the features are rather dependent instead of independent, which could be an explanation of the worse Naive Bayes scores. Naive Bayes is known to perform poorly when features are highly correlated.

Chapter 16

Humans vs computer

In earlier chapters we saw how the classification results are changed using different classification procedures. The best results in the main experiment were better than baseline results, but still not very satisfying. To be able to better judge the outcomes of the experiments, we investigate how well our systems perform in comparison to human judgments on estimating the time-to-event of single tweets. To have an indication, we carry out a small-scale experiment. We extract 500 tweets randomly, and from those we pick 80 tweets spread more or less realistically in time, with at least 9 instances in each of 6 selectively chosen categories: 7 days or more before event time, 4 to 7 days before event time, 3 to 1 days before event time, the day of the event, during the event and after the event. We choose these day-based categories because they would better align with human intuition. For example, a category like 9 to 12 days does not have any general temporal words in the Dutch language, whereas the category 'more than 7 days' has (f.e. next week).

The 9 participants[1] have to decide for each tweet to which of the 6 categories it belongs. We exclude one man of the participants because his answers do not adhere to the required format. We additionally ask the participants how certain they were about their answer on a scale from 0 (certain) to 2 (uncertain).

The human performance is scored by calculating the average percentage correct judgments for each category, the F1-score (based on the majority category given to each tweet), the distance error and average certainty of the participants for each category. Results are listed in Table 16.1.

The results show that humans have more difficulties (like computers) and are less certain in predicting the time-to-event when the event is still far away, espe-

[1]The group participants consisted of 5 men and 4 women, with an average age of 28 and median age of 24.

Table 16.1: Comparing the prediction accuracy of humans vs. the best system

		> 7	4-7	1-3	0	during	after	average
Humans	Percentage correct	.56	.30	.61	.74	.80	.65	.64
	F1-score	.56	.38	.64	.56	.77	.67	.69
	Distance error	.93	1.27	.53	.33	.22	.61	.58
	Certainty	1.22	1.14	.76	.37	.22	.43	.62
SVM,χ^2	Percentage correct	.27	.30	.46	.50	.96	.56	.51
	F1-score	.40	.32	.56	.62	.68	.56	.58
	Distance error	2.27	1.90	.73	.56	.04	.56	.84

cially for the penultimate time category. The distance errors are all below 1.3, which means that on average the predictions are maximally one category away from the correct category. Comparing the results to SVM with χ^2 feature selection (the best system in the main experiment) applied to these broad categories and tested on the same 80 tweets, we see that humans almost always outperform this system in terms of F1-score and distance error. Especially the category > 7 days before event time is better predicted by humans.

Looking at Table 16.2 and comparing human predictions to the predictions of our best system we can see that the RMSE of the human performance is considerably worse than the performance of the system, but that the responsiveness is much higher. Thus, humans are able to distinguish 'before' tweets from 'during' and 'after' tweets more accurately than SVM, which overpredicts the 'during' class. The worse RMSE does not automatically mean that humans did much worse. They classified the category > 7 days before event time more accurately, but this class has a very broad range (all tweets before 7 days or 168 hours) so the RMSE remains high.

Table 16.2: RMSE and responsiveness of the human experiment

	RMSE	responsiveness
SVM, χ^2	**62.1**	.58
Humans	95.6	**.92**

Summarizing, the results of this experiment show that there is much room left for computers to improve their responsiveness, and their predictions of the tweet posted long before the event.

Part V

Conclusion

Chapter 17

Discussion

From the results presented in previous chapters we can conclude that our approach to the classification of tweets over time outperforms baseline methods, but that there still is much room for improvement. The best RMSE in the main experiment is 52.3 hours, which amounts to just over two days. To predict an event with an error of 2 days that is still far away is pretty good, but for an event that starts within a couple of hours (which most event mentions in the tweets did) this RMSE is not useful. We will now discuss some points that emerged from the investigations.

It is hard to predict an event that is still far away, which is indicated by the evaluation measures. This can be explained in several ways. One is the fact that there is considerably less training data for these categories, which may bias the classifier to opt for the majority classes. Second, these tweets may be more diverse in content. Third, temporal words are less precise when they refer to longer time spans.

The regression-based study by Hürriyetoglu *et al.* [11] found a best RMSE of 43 hours, which is almost 10 hours better than the RSME of 52.3 hours of our study. However, Hürriyetoglu *et al.* only used events (football matches) with the same starting day and time, whereas our football matches started at different days (Friday, Saturday and Sunday), and times (from 12:30 pm to 8:45 pm). On top of that, only the tweets of 8 days or less before the event were collected. This could explain the better RMSE. All in all, both a regression based as the classification based procedure applied in our thesis are interesting for further research.

The calculation of the RMSE that excludes from the RMSE calculation of tweets classified as 'during' or 'after' is a decision we made, and we had some clear reasons therefore. First of all, we wanted to predict the time-**to**-event, and this study was not interested in the hours that the classification of the 'during' and 'after' tweets is off. Secondly, the 'after' class is not further divided into classes, so the RMSE of instances will still be high, even if they are classified correctly. Thirdly, it is harder to know what caused the better (or worse) RMSE. Is it because the large (and rather small) 'during' class was better classified? That is not what we are primarily interested in. Next to the RMSE we introduced the responsiveness, since only reporting the RMSE of the 'before' classes is not sufficient, because then we are not informed about the amount of 'before' tweets that is classified as 'during' or 'after'.

However, classifying a tweet from 1 week before the event as 'during' is worse than doing so for a tweet from 1 hour before the event, but these tweets are 'punished' equally in this way.

Current study was an exploratory study, and for this purpose current evaluation is sufficient to give a general view of the performance of the classification procedures. For a deeper analysis other or adapted evaluation measures should be thought of.

The standard deviations revealed that not all 10 folds performed equally. We chose to create the folds based on, as we called it, 'leave-six-events-out' cross-validation. As a result not every training and test fold had an equal amount of tweets, because not every match produced an equal amount of tweets. This fact can explain the differences in the evaluation, knowing that the amount of training data has influence on the results.

The performance of the informed baseline suggests that if real effort is put into the selected features and accompanied categories, it could be competitive with the fully automatic methods discussed above. Especially compared to the Naive Bayes and k-nearest neighbors classifications, the RMSE is less than 5 hours off, combined with a better responsiveness. However, the informed baseline has some drawbacks. Firstly, improving the performance of this classifier (i.e. selecting more and better features), will cost a lot of time and effort of human labor. Secondly, possibly underlying characteristics that are hard for humans to discover can be missed. Thirdly, the better responsiveness is mostly explained by our choice of classifying all tweets that do not contain any of the selected features as class 6. Lastly, it is expected to generalize poorly to new event domains.

The method of segmentation of the tweets over time can make big differences. The k-means segmentation has a significantly better RMSE for all procedures, with roughly the same responsiveness for both segmentation methods. It appeared that this was mainly due to the better classifications of the tweets posted far away from the event.

Training with different classifiers reveals that the choice of classification algorithm is important. The SVM classifier, which has been recognized as one of the most effective for text classification, had generally lowest RMSE rates. On the other hand, the Naive Bayes algorithm - although being a relatively simple algorithm - shows pretty good results as well, separating the best 'before' tweets from 'during' and 'after' tweets. On top of that, Naive Bayes generally had lower RMSE scores for tweets that are posted far away of the event, and is really fast. The k-nearest neighbors classifier showed worst performance, indicating that this classifier is not appropriate for this high-dimensional classification. Using less data to train with, similar patterns were observed. In contrast, using less features, the decrease in responsiveness for SVM is higher than for the other two classifiers, and k-NN has much lower RMSE scores than in the main experiment. Thus, especially SVM cannot handle classification with few features.

Training with less data showed that a training database of 140 thousand tweets is sufficient, but more data will aid the classifier in making better classifications, as we saw in the main experiment using 400 thousand tweets to train with. Using less than 20 thousand tweets is not sufficient. In general, the performance will plateau at some amount of data, which appears to be somewhere below a training set of 140 thousand tweets. After the plateau is reached, (small) improvements can still be made by only using more data.

Training with less features results in a big drop in responsiveness, and this went, surprisingly, hand in hand with better RMSE scores. However, analyzing these RMSE scores revealed that this was mostly due to the fact that relatively more tweets of the classes furthest away from the event were classified as 'during' or 'after', and in this way not included in the calculation of the RMSE.

Training with different feature selection methods has a mild influence on results. Training with 10,000 features, Information Gain gives better results if trained with Naive Bayes, whereas χ^2 gives slightly better results for the other two classifiers, and this pattern is observed for both segmentation methods. However, the differences between the feature selection methods are maximal 2 hours

only. Thus, our approach in the main experiment to the classification seems to be robust against these feature selection methods Investigating the same experiment and training with less features reveals that the choice of feature selection becomes more important, which is not surprisingly. It was however not the case that one of the feature selection methods consistently achieved the best results both in terms of RMSE and responsiveness.

The hierarchical classification was promising, but unfortunately results did not greatly improve. The best RMSE was lowered to 51.4, but this was at the cost of the responsiveness. The largest reduction in responsiveness emerged in the first step of the classification. If we can improve this layer of classification, then it might become useful to do the hierarchical classification. The study by Kunneman *et al.* [13] showed better results in the separation of the before tweets, so this is feasible if put more effort to it.

Using character n-grams as features caused opposite effects: the performance of the k-nearest neighbors classifier increased, whereas for Naive Bayes the RMSE worsened by more than 15 hours, but at the same time the responsiveness improved to 0.89. Thus, especially for the (first step of) hierarchical classification, the use of these features can be promising in combination with the Naive Bayes classification. Furthermore, a classification with a combination of the two kinds of features would be interesting to investigate.

17.1 Future research

Altogether, this thesis prepared the way for the adoption of further investigations, and pointed out which directions to take and which not. Some ideas for further studies will be presented.

Diverse events In order to be used in practice, more diverse events should be considered. In this thesis, only tweets that mentioned events that refer to football matches played in the Eredivisie are used to train and test with. The next step would be to test the system at different kinds of events, and ultimately at unscheduled events, in order to assist journalist, the police, etc. Classifying unscheduled events gives rise to the problem of collecting tweets about events. We collected tweets that referred to a specific hashtag, but then that is of course not possible anymore. In the section about related research the article about 'obtaining breaking news' a method is described to do this (or see article [29]).

Optimizing k-means clustering The k in k-means clustering was chosen so that both the left branching and the k-means clustering would have the same number of classes. This of course does not have to be the optimal k. Maybe it is better to segment the tweets in more, or indeed less segments. Having more classes can decrease the RMSE for the classes furthest away of the event, but then it can be expected that the F1-scores and distance errors will decrease. A proper balance between these two considerations must be stricken. Another point in optimizing the k-means clustering could be to begin the calculation of the clusters with different starting points, and see what kind of clusters then will be produced.

Hierarchical classification The influence of the 'during' class was very high in the experiments. A solution we came up with is the hierarchical classification. Indeed, the per class analysis showed an improvement, and the RMSE was lowered as well. Sadly, this was at the cost of the responsiveness, which is mostly decided in the first step of hierarchical classification. As the study by [13] showed, this responsiveness can at least be improved up to 0.88.
Another layer of classification could be a processing module that extracts specific date mentions. The inaccuracy of temporal words referring to longer periods, makes that people will use exact dates (*4th of November*) instead of phrases like *next week*. Besides that, in the data base we found some tweets containing messages like *'nog 15 uur en 17 minuten tot #psvaja'* (still 15 hours and 17 minutes to go till #psvaja). Seeing this tweet, humans know exactly when the event will take place, but the computer does not, unless language-specific rules are created that extract these time expressions and normalize them to a time-to-event estimate..

Evaluation in terms of RMSE As said in the discussion, an RMSE of two days can be pretty good in cases of events far away, whereas it is not useful for events that happen within a few hours or days. The introduction of a relative RMSE, that in some way gives higher punishments to events that are closest to event, could be a better evaluation measure. This has the further advantage that the RMSE cannot simply be lowered by classifying relatively more tweets that are far away of the event as 'during' or 'after'. As we saw for 100 features, this decreases the RMSE (since these are not counted anymore in current calculation of the RMSE). This is not what we want to achieve, since in many real-world applications the prediction of events some time ahead (weeks or more) will be of special interest, more than events that are about to happen. Besides that, as said in the discussion, the current calculation of the RMSE fails to give higher penalties in some cases. So for a more fine-grained analysis this should be taken into account to improve the evaluation.

Features One of the characteristics of words in tweets is that they often contain misspellings, abbreviations, etc. In the experiments we just used all words as features, without applying tokenizaton. So for example the features 'vnvnd' and 'vnvd' (both meaning 2nite), are treated as two different features, whereas they actually mean the same. However, we should not 'overfit' the tokenization: it could, for example, be possible that tweets that are posted further away of the event contain less misspellings than tweets that are posted right before the event, because people have more time to write the tweet. Apart from that, a combination of the bag-of-words features and character features could be used to train the classifier. As an extra feature the frequency of a character in a tweet could be added, because for characters it is more plausible that certain features will appear more than once in a tweet.

Individual tweets Even improving the classification, it seems reasonable that there will always be tweets that do no contain any information – in whatever form – about the time-to-event. Instead of classifying individual tweets, classifying groups of tweets that are posted in the same time span could circumvent this problem.

Chapter 18

Main conclusions

The scores for the different classes indicate that it is hard to predict an event that is still far away. There is a tendency to classify tweets with the most frequent 'during' class, which is reflected in the increasing distance error with tweets posted longer before the event. There may be several factors at play. First, the fact that there is considerably less training data for these categories hinders their learnability and may bias the classifier to opt for the majority class in case of doubt. Second, earlier tweets may be more diverse in content. Tweets posted closer to the match often focus on the match, whereas tweets posted earlier deal with various other aspects of the event, such as buying tickets, logistics, inviting friends, etc. Third, temporal words are less precise when they refer to longer time spans.

The method of time segmentation is important. We saw that the categorization using the k-means method produces better results than the left branching method. We explained this by the fact that k-means clustering takes into account content-based characteristics of event-related tweets over time, instead of assuming some mathematical distribution of occurrences. On top of that, the distance errors of the experiment with humans showed that there is room for improvement for these tweets long before the event. The k-NN classification algorithm is not appropriate for current experiments. On the other hand, SVM performs the best in terms of RMSE, but Naive Bayes has a better responsiveness.

As expected, a decrease in training instances leads to an overall lower performance, whereas using less features generally leads to a better RMSE. However, this better RMSE is mainly due to the fact that relatively more instances of the classes longer before the event are classified as 'during' or 'after', and therefore not included in the calculation of the RMSE. Hierarchical classification and the use of characters as features are methods beginning to address improvements to the system, but work in these methods should be fine-tuned.

Part VI

Appendices

Appendix A

Tables

Table A.1: Top 30 of selected features for each of the two segmentation and the two feature selection methods. The features '@omaamomentjes hoeveel' and '@omaamomentjes' for the χ^2 feature selection method are very surprisingly. The other features seem to be very reasonable.

ranking	Left branching IG	Left branching χ^2	k-means IG	k-means χ^2
1	morgen	zondag	morgen	morgen
2	zondag	volgende week	zondag	zondag
3	volgende week	volgende	volgende week	volgende week
4	volgende	morgen	vandaag	volgende
5	naar	week	naar	week
6	week	kaarten	volgende	vandaag
7	kijken	was	kijken	vanavond
8	was	zaterdag	week	was
9	vandaag	naar	was	naar
10	1-0	kijken	vanavond	kaarten
11	*6-10 woorden*	vanavond	1-0	zaterdag
12	vanavond	@omaamomentjes hoeveel	vanmiddag	kijken
13	wat een	@omaamomentjes	*6-10 woorden*	vanmiddag
14	*1-5 woorden*	1-0	wat een	1-0
15	uur	vandaag	*1-5 woorden*	voor morgen
16	straks	weken	straks	matchday
17	vanmiddag	*6-10 woorden*	uur	morgen naar
18	rust	slapen	*21-25 woorden*	@omaamomentjes hoeveel
19	*21-25 woorden*	voor zondag	rust	@omaamomentjes
20	klaar	matchday	klaar	*6-10 woorden*
21	klaar voor	wat een	matchday	straks
22	zin in	straks	klaar voor	weken
23	zin	hoeveel word	zin in	voor zondag
24	slapen	uur	zin	slapen
25	matchday	rust	zaterdag	wat een
26	zaterdag	*21-25 woorden*	slapen	uur
27	*16-20 woorden*	klaar	*16-20 woorden*	klaar
28	van	*1-5 woorden*	van	klaar voor
29	kaarten	kaarten voor	kaarten	*1-5 woorden*
30	wat	vanmiddag	wat	*21-25 woorden*

Table A.2: Words selected for the informed baseline.

Feature	**Translation**	**Class Left branching**	**Class k-means**
morgen	tomorrow	5	4
overmorgen	the day after tomorrow	3	3
straks	later	5	5
zo	soon	6	6
zometeen	in a little while	6	6
gisteren	yesterday	8	8
volgende week	next week	2	1
vanavond	tonight	5	5
vandaag	today	5	5
vanmiddag	this afternoon	6	5
nu	now	7	7
begonnen	started	7	7
klaar	ready	6	6
zondag	Sunday	3	2
zaterdag	Saturday	3	2
vrijdag	Friday	3	2
komend weekend	this weekend	2	1
volgend weekend	this weekend	2	1
zodirect	in a little while	6	6
1 (een) dag	1 (one) day	4	4
2 (twee) dagen	2 (two) days	4	3
3 (drie) dagen	3 (three) days	3	2
4 (vier) dagen	4 (four) days	3	2
5 (vijf) dagen	5 (five) days	3	2
6 (zes) dagen	6 (six) days	2	1
10 (tien) dagen	10 (ten) days	1	1
twee weken	two weeks	0	0
een week	one week	2	1
2 uur	2 hours	6	6
1 uur	1 hour	6	6
dag van	day of	24	5
spannende dag	exciting day	5	5
paar dagen	couple of days	3	2
afgelopen	finished	8	8
kaarten	tickets	2	1
kaartje	ticket	2	1
kaartjes	tickets	2	1
tickets	tickets	2	1
kijken	watching	7	7
naar huis	going home	8	8

Bibliography

[1] Sitaram Asur and Bernardo A. Huberman. Predicting the future with social media. *Computing Research Repository (CoRR)*, abs/1003.5699, 2010.

[2] Baharum Baharudin, Lam Hong Lee, and Khairullah Khan. A review of machine learning algorithms for text-documents classification. *Journal of Advances in Information Technology*, 1(1), 2010.

[3] Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, COLT '92, pages 144–152, New York, NY, USA, 1992. ACM.

[4] John D. Burger, John Henderson, George Kim, and Guido Zarrella. Discriminating gender on twitter. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 1301–1309, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics.

[5] Cheyanne Church and Mark M. Rogers. *Designing for results*. Search for Common Ground, Washington, 2006.

[6] Paul Earle, Daniel Bowden, and Michelle Guy. Twitter earthquake detection: earthquake monitoring in a social world. *Annals of Geophysics*, 54(6):708–715, 2012.

[7] Paul Earle and Peter Shearer. Characterization of global seismograms using an automatic-picking algorithm. *Bulletin of the Seismological Society of America*, 84:366–376, 1994.

[8] Daniel Gayo-Avello. "I wanted to predict elections". *Computing Research Repository (CoRR)*, abs/1204.6441, 2012.

[9] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. Prototype methods and nearest-neighbors. In *The Elements of Statistical Learning*, Springer Series in Statistics, pages 459–483. Springer New York, 2009.

[10] Richard Holts, mar 2013. `http://www.telegraph.co.uk/technology/twitter/9945505/Twitter-in-numbers.html`.

[11] A. Hürriyetoglu, F. Kunneman, and A. van den Bosch. Estimating the time between twitter messages and future events. In *Proceedings of the 13th Dutch-Belgian Workshop on Information Retrieval (DIR 2013)*, number 986, pages 20–23, 2013.

[12] Andreas Jungherr, Pascal Jürgens, and Harald Schoen. Why the pirate party won the German election of 2009 or the trouble with predictions: A response to Andranik Tumasjan, Timm Sprenger, Philipp Sandner and Isabell Welpe: Predicting elections with Twitter: "What 140 characters reveal about political sentiment". *Social Science Computer Review*, 30(2):229–234, 2012.

[13] F. Kunneman and A. Van den Bosch. Leveraging unscheduled event prediction through mining scheduled event tweets. In *Proceedings of the 24th Benelux Conference on Artficial Intelligence*, pages 147–154, Maastricht, The Netherlands, 2012.

[14] S. Landau and B. Everitt. *A Handbook of Statistical Analyses Using SPSS*. Statistics (Chapman & Hall/CRC). Chapman & Hall/CRC, 2004.

[15] Shoushan Li, Rui Xia, Chengqing Zong, and Chu-Ren Huang. A framework of feature selection methods for text categorization. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, volume 2 of *ACL '09*, pages 692–700, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics.

[16] Christine Liebrecht, Florian Kunneman, and Antal Van den Bosch. The perfect solution for detecting sarcasm in tweets #not. In *Proceedings of the 4th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 29–37, Atlanta, Georgia, June 2013. Association for Computational Linguistics.

[17] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA, 2008.

[18] Viktor Mayer-Schonberger and Kenneth Cukier. *Big Data: A Revolution That Will Transform How We Live, Work, and Think*. Houghton Mifflin Harcourt, Boston, 2013.

[19] Simon McEvoy, jan 2013. `http://www.relativelystraightforward.com/post/40175883025/are-hashtags-the-new-sarcasm-mark-yeahright`.

[20] Z. Miller, B. Dickinson, and W. Hu. Gender prediction on twitter using stream algorithms with n-gram character features. *International Journal of Intelligence Science*, 2(4A):143–148, 2012.

[21] D. Nguyen, R. Gravel, D. Trieschnigg, and T. Meder. "How old do you think i am?"; a study of language and age in Twitter. In *Proceedings of the Seventh International AAAI Conference on Weblogs and Social Media*, 2013.

[22] Nelleke Oostdijk and Hans van Halteren. N-gram-based recognition of threatening tweets. In *Conference on Computational Linguistics and Natural Language Processing (CICLing) (2)*, pages 183–196, 2013.

[23] Irina Rish. An empirical study of the Naive Bayes classifier. In *First International Joint Conference on Artificial Intelligence workshop on "Empirical Methods in AI"*, 2005.

[24] Alan Ritter, Mausam, Oren Etzioni, and Sam Clark. Open domain event extraction from twitter. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '12, pages 1104–1112, New York, NY, USA, 2012. ACM.

[25] Kirk Roberts, Michael A. Roach, Joseph Johnson, Josh Guthrie, and Sanda M. Harabagiu. Empatweet: Annotating and detecting emotions on twitter. In *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, Istanbul, Turkey, may 2012. European Language Resources Association (ELRA).

[26] Takeshi Sakaki, Makoto Okazaki, and Yutaka Matsuo. Tweet analysis for real-time event detection and earthquake reporting system development. *IEEE Transactions on Knowledge and Data Engineering*, 25(4):919–931, 2013.

[27] Eric Sanders and Antal Van Den Bosch. Relating political party mentions on twitter with polls and election results. In *Proceedings of the 13th Dutch-Belgian Workshop on Information Retrieval (DIR)*, number 986, pages 68–71, 2013.

[28] Erik Tjong Kim Sang and Johan Bos. Predicting the 2011 dutch senate election results with twitter. In *Proceedings of the Workshop on Semantic Analysis in Social Media*, pages 53–60, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics.

[29] Jagan Sankaranarayanan, Hanan Samet, Benjamin E. Teitler, Michael D. Lieberman, and Jon Sperling. Twitterstand: news in tweets. In *Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, GIS '09, pages 42–51, New York, NY, USA, 2009. ACM.

[30] Rushdi Shams, aug 2011. `http://rushdishams.blogspot.nl/2011/08/micro-and-macro-average-of-precision.html`.

[31] Beaux Sharifi, Mark-Anthony Hutton, and Jugal Kalita. Summarizing microblogs automatically. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, HLT '10, pages 685–688, Stroudsburg, PA, USA, 2010.

[32] Andranik Tumasjan, Timm Sprenger, Philipp Sandner, and Isabell Welpe. Predicting elections with Twitter: What 140 characters reveal about political sentiment. *International AAAI Conference on Weblogs and Social Media*, 2010.

[33] C.J. Van Rijsbergen. *Information Retrieval*. Buttersworth, London, 1979.

[34] W. Weerkamp and M. de Rijke. Activity prediction: A twitter-based exploration. In *SIGIR 2012 Workshop on Time-aware Information Access*, aug 2012.

[35] Allan P. White and WeiZhong Liu. Technical Note: Bias in information-based measures in decision tree induction. *Machine Learning*, 15(3):321–329, 1994.

[36] Xindong Wu, Vipin Kumar, J. Ross Quinlan, Joydeep Ghosh, Qiang Yang, Hiroshi Motoda, Geoffrey J. McLachlan, Angus Ng, Bing Liu, Philip S. Yu, Zhi-Hua Zhou, Michael Steinbach, David J. Hand, and Dan Steinberg. Top 10 algorithms in data mining. *Knowl. Inf. Syst.*, 14(1):1–37, December 2007.

[37] Yiming Yang and Jan O. Pedersen. A comparative study on feature selection in text categorization. In *Proceedings of the Fourteenth International Conference on Machine Learning*, ICML '97, pages 412–420, San Francisco, CA, USA, 1997. Morgan Kaufmann Publishers Inc.

[38] Sheng Yu and Subhash Kak. A survey of prediction using social media. *Computing Research Repository (CoRR)*, abs/1203.1647, 2012.

[39] Siqi Zhao, Lin Zhong, Jehan Wickramasuriya, and Venu Vasudevan. Human as real-time sensors of social and physical events: A case study of twitter and sports games. *Computing Research Repository (CoRR)*, abs/1106.4300, 2011.