NTNU
Norwegian University of
Science and Technology

Event Detection in Social Media

Detecting News Events from the Twitter
Stream in Real-Time

Øystein Kvamme Repp

Master of Science in Computer Science
Submission date: June 2016
Supervisor: Herindrasana Ramampiaro, IDI

Norwegian University of Science and Technology
Department of Computer and Information Science

Abstract

The proliferation of social media and user-generated content in the Web has opened new opportunities for detecting and disseminating information quickly. The Twitter stream is one large source of information, but the magnitude of tweets posted and the noisy nature of its content makes the harvesting of knowledge from Twitter very hard.

Aiming at overcoming some of the challenges and extract some of the hidden information, this thesis proposes a system for real-time detection of *news events* from the Twitter stream. The first step of our approach is to let a classifier, based on an Artificial Neural Network and deep learning, detect news relevant tweets from the stream. Next, a novel streaming data clustering algorithm is applied to the detected news tweets to form *news events*. Finally, the events of highest interest is retrieved based on events' sizes and rapid growth in tweet frequencies, before the news events are presented and visualized in a web user interface.

We evaluate the proposed system on a large, publicly available corpus of annotated news events from Twitter. As part of the evaluation, we compare our approach with a related state-of-the-art solution. Overall, our experiments and user-based evaluation show that our approach on detecting current (real) news events delivers state-of-the-art performance.

Sammendrag

Den hurtige utbredelsen av sosiale medier og brukergenerert innhold på nettet har åpnet nye muligheter for rask oppdagelse og spredning av informasjon. Twitter-strømmen er én stor kilde til informasjon, men et enormt omfang samt mye støy gjør den veldig vanskelig å hente ut.

For å dra nytte av noe av denne skjulte informasjonen, tar vi denne oppgaven sikte på å overvinne noen av utfordringene rundt informasjonsuthenting fra sosiale medier og foreslå et system for sanntidsoppdagelse av nyhetshendelser i Twitter-strømmen. Det første steget består i å anvende en klassifikator, basert på et kunstig nevralt nettverk og dyp læring, for å gjenkjenne Twitter-meldinger som omhandler nyhetshendelser. Videre tar vi i bruk vår egen, nye grupperingsalgoritme for å gruppere Twitter-meldinger som er relaterte til de samme nyhetshendelsene. Til slutt henter vi ut, presenterer, og visualiserer de nyhetshendelsene som er antatt viktigst, basert på en analyse av omfang og positiv endring i frekvens av relaterte Twitter-meldinger.

Det foreslåtte systemet blir evaluert ved å benytte et stort offentlig tilgjengelig datasett bestående av Twitter-meldinger med bekreftet nyhetsrelevans. Som en del av denne evalueringen, måler vi løsningen vår opp mot en ledende metode innen feltet. Samlet sett viser eksperimentene, samt en brukerbasert evaluering, at vår metode presterer bedre enn dagens løsninger.

Preface

This thesis is submitted to the Norwegian University of Science and Technology in Trondheim, as a final fulfillment of the requirements for a Master of Science in Computer Science with specialization within Artificial Intelligence. This work was conducted at the Department of Computer and Information Science during the spring semester 2016.

Acknowledgements

Firstly, I would like to thank my supervisor, Associate Professor Heri Ramampiaro at the Department of Computer and Information Science (IDI), for valuable discussions, excellent guidance, and continuous feedback on the thesis throughout the whole project. His faith in me have truly been inspiring when problems have seemed impossible.

Secondly, I would thank all my friends in the Sule Masters Hall at IDI for contributing to our little research community, as well as for all valuable (and less valuable) discussions throughout the spring.

Thirdly, all my friends, who spent almost an hour each on the user-based evaluation, deserve to be mentioned: Sindre M., Sindre B., Petter, Nemanja, Håvard, Synne, Ole Andreas, Marthe and Simen, *thank you!*

Finally, my girlfriend Marthe deserves the uttermost gratitude for her patience, as well as a big apology for all the late-night dinners. I recognize that I have some catching-up to do on housework…

List of Tables

List of Figures

Contents

Chapter 1

Introduction

This chapter introduces the main topics of the thesis. It focuses on its motivation, its context, and explanations of its goal. Section 1.1 explains the motivation behind information retrieval from Twitter in general, and behind the goal of the thesis in specific. The context of the project is presented in Section 1.2, and some important definitions used throughout the thesis are given. The research questions are presented in Section 1.3, while you will find in an outline of the thesis in Section 1.4.

1.1 Motivation

Social Media has become the human being's perhaps most important mean of communication during the last decade. One of the biggest contenders in the world of Social Media, in terms of user base and in information content, is the microblog service Twitter. Today, more than 500 million Twitter status updates, hereafter referred to as *tweets*, are sent every day[1], all over the world. The concise and informal expressional form that characterizes tweets makes the threshold for posting updates low, which may explain the enormous volume. With the increased popularity of smartphones in the past 10 years, almost everyone is nowadays able to communicate real-time information to the whole world the moment they witness or are involved in an event. Since Twitter's launch in 2006, common people have been using Twitter for anything from sharing today's dinner to knocking down totalitarian regimes, and it has grown to become the perhaps most up-to-date information channel of all.

Although there is no reason to doubt Twitter's value as a source of information regarding real-world events, it has turned out to be hard to successfully make use of all its hidden information. Due to the 140-character limit of Twitter and its informal nature, the posted tweets are very noisy in its nature. People tend to use abbreviations and "SMS language", emoticons, specially constructed hashtags (which often are concatenations of actual words that are rarely seen elsewhere), as well as less distinctive means as humor and sarcasm. As human beings we are usually able to grasp the meaning of a tweet, despite these noisy

[1] `http://www.businessinsider.com/twitter-tweets-per-day-appears-to-have-stalled-2015-6`

aspects, but the same task is very difficult for computers. The short format of tweets makes sparsity a problem, as the dimensionality of the "Twitter vocabulary" is enormous. Traditional text mining techniques has a tendency to fail on Twitter data, as they were typically designed for longer texts following strict grammatical rules.

By overcoming the challenges with Twitter, we can harvest of all its hidden information. Among this information we find news events, perhaps even reported in real-time by first-hand witnesses. Petrovic et al. (2013) reports that even in the mere 1% of the public Twitter stream, that is made freely available through Twitter's Streaming API, around 95% of all events reported in traditional newswire is covered. In addition, the Twitter stream also contains several other events that are either too small, too local or too domain-specific to be of any interest for the newswire, as well as events that have value for only a short period of time. If we are able to identify and separate news events from the flood of spam and everyday events that are reported on Twitter, we can create a system that displays only the "important stuff". Such a system can be of use for journalists who increasingly adopt social media as a professional tool (Jordan, 2013; Schifferes et al., 2014), as well as other parties interested in early detection of event throughout the world, such as governments and organizations of various kinds. We know for a fact that human rights organizations are already monitoring social media for identification of human right violence (Chen and Neill, 2015).

It is also likely that the average Twitter user would be interested in using such a system to keep up up to date on hot news, as an extensive study produced by the American Press Institute and Twitter shows that *nearly 9 in 10 Twitter users use Twitter for news*, and the vast majority of those (74%) do so daily (Rosenstiel et al., 2015). The same study also shows that 40% of the users us Twitter to be alerted to breaking news, while 40% say they use it to keep up with news in general. A system that is able to detect news events in the streaming setting of Twitter can be used to report breaking news (first-story detection for news), or it can be used to report the last events happening over a given time – i.e. as a news summary. Such systems would possibly help us detect, collect and disseminate new information even quicker than today, which really is the main motivation behind this thesis. The main contribution of this thesis is the proposition of a method to detect news events that are attracting interest in Twitter and to make this information available with a close to real-time latency. We also implement a prototype system to illustrate how the proposed method can be applied to create a newswire service.

1.2 Context

1.2.1 Event Detection

Generally speaking, an event can be referred to as a real-world happening that finds place in a given space at a given time. Detection of such events has previously been addressed in the field of information retrieval (IR), but the focus has mainly been on detecting events from conventional media sources, such as RSS feeds from news broadcasters. The Topic Detection and Tracking (TDT) (Allan, 2002) project is perhaps the most pronounced example, where the goal was to produce a system that was able to monitor broadcasted news and produce an alert whenever a new event occurred. Most TDT systems adopts a simple nearest neighbor clustering algorithm, and some reasonably effective systems have been produced. However, contents from news broadcasters deviate from typical tweets in the way that it is usually long, well-written and rich on semantics, while tweets are very short and often contain large amounts of abbreviated, informal and irregular words (e.g. hashtags), as well as grammatical errors and odd sentence structures. These are reasons that approaches used in traditional information retrieval often perform poorly when applied directly to Twitter data. With the last years' explosion in the use of social media and websites based on user-generated content, information retrieval within these domains has, however, received an increasing amount of attention from the IR community.

The event detection task is actually one of the most commonly studied tasks in the Twitter domain. It is related to *trending topics*, which is often mentioned when speaking of Twitter, but it differs in the way that the objective is to detect real-world happenings based on what is said in Twitter. Trending topics, on the other hand, is simply concerned about what attracts interest in the Twittersphere and is often limited to bursting terms and popular hashtags. Some of the literature addressing the challenges of working with content from microblog services such as Twitter, with a main focus on those works addressing event detection and/or extraction in particular, will be presented later in this thesis.

Despite the attention that event detection within microblogs has been subjected to through the last decade, we stress that it is a task that not should be regarded as solved just yet. The reason for this is very likely due to the lack of a common understanding of the task.

1.2.2 Defining Event

There exists no widely agreed definition of event. As a result, there is lack of a common understanding of the event detection task. This makes comparisons of different detection approaches very hard. It is especially the granularity level of events that differs from work to work. Where some may consider something a single event, others may break the same event into multiple events. This has caused researchers to pose any kind of restrictions and limitations on their systems, making comparison difficult. McMinn et al. (2013) addresses this problem in their work on creating a large event detection corpus on Twitter data, and they propose a new definition of event. The definition they propose fits the objective of this thesis perfectly and will therefore be used in this work. The definition requires the following two points to be fulfilled:

Definition 1. ***Event:***

1. *An event is a significant thing that happens at some specific time and place.*
2. *Something is significant if it may be discussed in the media. For example, you may read a news article or watch a news report about it.*

McMinn et al. (2013)

By using this definition, detected events will automatically be what we consider to be news events. However, even though a news event necessarily must happen at some specific time and place, it does not mean that tweets referring to it mention this explicitly. It is therefore important to emphasize that this is a definition of the *event itself*, and *not* the tweets referring to it. I.e. a tweet referring to a news event does not have to include information regarding time and place. As an example, let us look at the tweet:

“`Police officer shot in bank robbery`”

This tweet mentions neither time nor place, but if it refers to an actual episode, that episode is by no doubt a news event. For this thesis, which objective is real-time detection of news events, we will work out from the assumption that any mentioned event has news relevance at the time the tweet referring to it is posted. In other words, an event that has already taken place, or that will take place in the future, can still generate a *news event* in Twitter today.

Throughout this thesis we distinguish *news tweets* and *news events* by the following definitions:

Definition 2. ***News tweet****: A tweet that refers to a specific news event, or in any other way is directly related to such an event as described in Definition 1.*

Definition 3. ***News event****: A group of* news tweets *that are naturally connected to each other because of temporal and semantic similarities.*

A more plain explanation of Definition 3 is that it is a group of *news tweets* discussing the same topic at similar times. Any tweets that are not related to news are considered *irrelevant.*

1.2.3 Variations of Event Detection

Just as there are different definitions of event, there are different shades of event detection. There is detection of specified versus unspecified event types, where the question is whether we want to detect events of a given type (e.g. catastrophes, riots, music concerts) or just any kind of event. Another question is whether we want to detect events that have occurred in the past (retrospective event detection) or new events as they emerge (new event detection). The former has the advantage that an iterative approach can be taken, while the latter typically requires online processing of the data. The third and last obvious subdivision is related to what method that is used for detection – supervised or unsupervised. Examples of these differences will be shown in the literature survey in Chapter 2.

In this project the specific event type we address, is *news events*. Whereas news events may refer to news of any kind, the event type can be said to be more or less unspecified. The time of occurrence for the real-world event that generates an event *in Twitter* is not important, as an sudden increase in the interest it attracts in Twitter indicates that it has news value at the present moment – thus making it a news event. Nevertheless, it is reasonable to assume that Twitter users are more prone to tweet about recent events events rather than old ones.

1.2.4 Feature Engineering

Feature engineering is one great challenge when working with tweets. The textual content itself is likely to be an important feature for event detection from tweets. However, with tweets being as short as they are and their dimensionality so ludicrously large, sparsity becomes an issue that makes traditional methods such as bag-of-words and word vectors less suitable for representing the content in the text. It is therefore important come up with a way of representing the text that is able to capture the information necessary for recognizing news, while at the same time keeping the dimensionality low enough to make the application of machine learning or clustering algorithms computationally feasible. In addition to the text, tweets contain Twitter-specific meta data that may or may not be valuable for solving the task, such as user information (e.g. number of previous tweets, number of followers) and number of retweets[2].

1.3 Research Questions

To help harvesting some of the hidden and valuable information in Twitter by developing methods for real-time reporting of news events, the main research question we aim to answer is:

RQ *How can news events be detected in real-time from the Twitter stream?*

To do this, we will address the following subquestions:

RQ 1 *How to choose features from tweets and how should we represent them when using an artificial neural network to filter news tweets from non-news tweets?*

RQ 2 *How can we cluster semantically and temporally similar news tweets to form events in real-time?*

RQ 3 *How can we identify the important news events for presentation?*

1.4 Thesis Outline

This thesis begins with **Chapter 1** giving an introduction to its treated field of research. Herein the work is contextualized and the research questions are stated. Further, it continues with a survey in **Chapter 2** where the literature on *event detection* in Twitter is reviewed. **Chapter 3** provides some background on the problems being treated in the thesis, and typical challenges facing us when dealing with Twitter data is discussed. It also presents some information regarding text representation in information retrieval, as well as some insight to clustering and classification in the Twitter-domain. In **Chapter 4** things are really getting interesting, as our proposed system for detecting *news events* is presented. Some experiments are conducted, and choices for the final system are made based upon them. The experimental setup for evaluating our system is then explained in **Chapter 5**, before the results of the evaluations are presented and discussed in **Chapter 6**. Finally, in **Chapter 7**, conclusions are drawn, the contributions of this thesis is laid out, and its goal achievements with regard to the research questions are presented. The last chapter also includes propositions for possible future work.

[2]Tweets that are re-posted by other users.

Chapter 2

Literature Survey

This chapter contains a survey of works related to this thesis. Firstly, the most closely related works are presented in detail in Section 2.1, then other related work are presented in less detail in Section 2.2, and lastly, a summary of the survey is given in Section 2.3.

2.1 Closest Related Works

2.1.1 Real-Time Event Detection

Petrovic et al. (2010) present a system for First Story Detection (FSD) in Twitter that works in the streaming model, i.e. where data arrive continuously in a chronological order and every new instance must be processed in bounded space and time. FSD is not exactly the same as event detection, but it is very closely related. The objective of FSD is to, given a sequence of stories, identify the first story to discuss a particular event. This means that the event detection problem must be solved as a part of finding first stories, and as suggested in the introduction, a system for event detection (such as ours) can be exploited for detecting first stories. Petrovic et al. (2010) meet the space/time complexity requirement of the streaming model by keeping the amount of stories in memory constant, and by employing Locality Sensitive Hashing (LSH) for nearest neighbor search, respectively. They find that applying pure LSH to the problem of FSD yield poor performance, and therefore come up with a modification that virtually eliminates the variance and improves the performance.

Petrovic et al. initially test their system on the standard TDT5 dataset[1], where they report results comparable to state-of-the-art (SoA) systems in terms of accuracy, while obtaining an order of magnitude improvement in terms of speed. Given that the majority of tweets are not real stories, but rather spam and mundane updates from the users' everyday life, a direct application of a FSD system would result in an enormous amount of new stories every second. To overcome this problem and make it possible to apply their system on Twitter data, the authors come up with the concept of *threading*. They first run the FSD system on the Twitter stream and let it assign a ”novelty score” to each tweet. Secondly

[1]Dataset released in connection with the NIST Topic Detection and Tracking project. `http://www.itl.nist.gov/iad/mig/tests/tdt/`

they use the cosine similarity (which was already calculated as a part of the first step) between TFIDF weighted vector representations of tweets to analyze threads, i.e. subsets of tweets that all discuss the same topic. The threads are generated by introducing the ”links relation,” where they define that a tweet A links to tweet B if B is the nearest neighbor of A and $1 - cos(A, B)$ is below a specified threshold. Each tweet is then assigned either to the same thread as its nearest neighbor, or to a new thread, in which it is the first tweet, if the $1 - cos(A, B)$-value is below the threshold. In that way the granularity of the threads is easily controlled by changing the threshold value. Once the threads are generated, the authors use the growth of speed for each thread to measure its impact. For each time interval only the fastest growing threads are outputted.

For evaluation they let their system run on a corpus of 164.5 million tweets collected over a period of six months using Twitter's Streaming API[2]. They preprocess the data by removing words containing @ (user mentions) and # (topic tag). These features are removed to make the approach independent of stream type, although the authors claim that they probably would have been helpful for the specific Twitter case. They do not evaluate the system's ability to detect first stories in terms of recall, as this would have involved having human experts read through the whole corpus looking for first stories – a task so big that it is close to impossible. They do, however, let human experts manually label all the tweets returned by the system as either event, neutral or spam, and show that their system is able to detect major events with a reasonable precision. They also conclude that the number of people writing about an event is more indicative of the event's impact than the number of tweets written about it.

McMinn and Jose (2015) is another work that aim at real-time detection of events in the Twitter stream. The approach they take is to use named entities to detect and track events, on the grounds that named entities are the building blocks of events and they reason that the people, places and organizations involved are crucial in describing an event. Their proposed approach to detect events is to identify bursty named entities and then use an efficient clustering method to detect and break events into individual topics.

Following the alleged importance of named entities, McMinn and Jose (2015) filter all tweets that contain no named entities. They also remove re-tweets as well as tweets containing terms that are unlikely to be related to events or that are known to be associated with spam and noise. This is reported to reduce the amount of data that need to be processed by over 95%.

To cluster the tweets McMinn and Jose uses a method similar to the one found in Petrovic et al. (2010), but instead of using extensive tweet-to-tweet comparisons, they compare new tweets only to other tweets in which named entities co-occur. For each named entity in a new tweet, a list of tweets containing the same entity is looked up in an inverted index, and the maximum TF-IDF weight cosine similarity is calculated. If the similarity is above a set threshold, the new tweet is assigned to the same cluster as its nearest neighbor. To ensure real-time performance, the number of tweets to be retrieved from the inverted index is limited, and only the top-ten TF-IDF weighted terms is used per tweet.

To detect events, McMinn and Jose (2015) look for bursts in the frequency of an entity over a set of time windows, and define an entity's window as bursty if the number of

[2]`https://dev.twitter.com/streaming/public`

tweets in it is greater than the mean plus three standard deviations. Once a burst has been identified, an event is created and associated with the bursting entity, and entity clusters associated with the bursting entity is associated with the event. To avoid to have a single event for each entity involved in a real-world event, they merge events A and B if an entity e is mentioned in at least 50% of the tweets in A while at the same time being part of event B.

McMinn and Jose (2015) evaluate their method on the Event2012 dataset from McMinn et al. (2013) and are thereby, to the best of their knowledge (and our), the first to evaluate their system on a large-scale publicly available dataset. They do however use a rather low threshold for when an event should be considered detected, requiring only that 5% or at least 15 tweets in a candidate event must be relevant to a single ground-truth event for it to be counted as detected. Their rational for this is, and righteously so, that it is impossible that the the collection has judgements for every event which occurred over the 28 days it covers, and because the dataset was generated by using different automated methods each with different level of granularity, the events in the dataset vary between being sub-events and full events (for example, a statement made in a Primary Debate, rather the the Primary Debate itself). A high threshold would therefore make it difficult for an event to be relevant if the system detects a full event rather than the sub-event(s) that is recorded in the dataset. They also evaluate their system using crowdsourcing, which shows that the precision was not overestimated as a consequence of the low threshold value. They report (*precision*, *recall*, *F1*)-scores of (0.636, 0.383, 0.478) when using crowdsourcing for evaluation and (0.302, 0.310, 0.306) when evaluating using the *Events2012* dataset (best run, events with 75+ tweets).

The work presented in Petrovic et al. (2010) is interesting for this thesis as it threats the handling of large amounts of Twitter streaming data in a very efficient manner, and more so, the authors claim to be the first doing it in such a scale. We have a common goal of event detection, although the new/first-aspect is not crucial for our task. The weakness of their method is that it does not take the content of tweets into account, so that all topics (or threads in their wording), for which a sudden rise in interest is detected, will potentially be returned as an event. They are able to filter out spam by taking entropy into account, but it is still likely that the system will return many events that are not related to real-world events. In that way one could say that their system detects emerging trends rather than new events.

McMinn and Jose (2015) is relevant as its objective is the same as this thesis – real-time news event detection in the Twitter stream. It is also the only, to the best of our knowledge, work where the evaluation is done using a publicly available dataset. This, besides state-of-the-art performance, makes it an evident candidate to measure our own work against.

2.1.2 Open Domain Event Extraction

Ritter et al. (2012) describe a system (TwiCal) that they claim is the first open-domain event-extraction and categorization system for Twitter. We have already seen that Petrovic et al. made a system for first-story detection in 2010, but they did, as mentioned, not focus exclusively on events, nor did they classify their extracted tweets. The objective of TwiCal is to extract structured representations of events from Twitter, represented as 4-tuples containing a named entity, an event phrase, a calendar date, and an event type.

The events are extracted after performing various pre-processing steps. The tweets are first POS tagged, and then name named entities and event phrases are extracted using supervised methods trained on in-domain Twitter data in a supervised manner. The authors report an considerable increase in performance using methods trained on Twitter data over traditional Natural Language Processing (NLP) tools typically trained on edited texts such as news articles. In the end temporal expressions are resolved, and the extracted features are used to classify the events into event types.

As there is hard to predict the set of event types that will be mentioned on Twitter in advance, the authors come up with a novel approach to discover important event types and categorize aggregate events within a new domain. They subsequently inspect the automatically discovered types to filter out incoherent ones and annotate the rest with informative labels. These labels are used for the previously mentioned classification.

The classification is done by adopting an approach based on latent variable models, where each event phrase is modeled as a mixture of the event types, and each type corresponds to a distribution over the named entities found in a specific instances of the type, as well as a distribution over all dates on which the event of the type occurs. Prediction of new events is done by using a streaming approach to inference.

To test their system the authors gather around 100 million tweets posted during a one-day timeframe by tracking temporal keywords (‘today’, ‘tomorrow’, weekdays, etc.) using Twitter’s Streaming API. The evaluation is done by manually inspecting samples of the top 100, 500 and 1,000 returned event-tuples for events apparently taking place within a 2-week future window starting the day after the data was collected. For the 100 highest ranked entries, they report a precision of 90%, which must be said to be good.

Zhou et al. (2014) also focus on extracting structured representations of events. They propose a simple, yet effective, Bayesian model, which they call Latent Event Model (LEM), to extract structured representations of events from social media, represented as (*named entity*, *time*, *location*, *keywords*)-tuples. The model is fully unsupervised and relies on traditional IR preprocessing steps such as time expression recognition, named entity recognition, part-of-speech (POS) tagging, and stemming. After pre-processing, non-location entities, locations, dates and candidate keywords are selected as input to the LEM model for event extraction. The authors utilize the fact that in social media the same event could be referenced by high volume of messages. Due to this property, they come up with their statistical model that groups similar events based on the co-occurrence patterns of the elements forming the event. By treating each event as a latent variable, they model its generation as a joint distribution of its individual elements.

Their work is very similar to TwiCal (Ritter et al., 2012) in terms of objectives, but the main difference is that Zhou et al. are able achieve their goals in an unsupervised manner, and even more so they claim even better results than Ritter et al. (2012). The data which they apply their system to is the First Story Detection dataset (Petrovic et al., 2013), which consists of 2,499 tweets manually annotated with one corresponding event from a total of 27 different events mentioned in the tweets. This is a very small dataset to benchmark against, when we know that over 500 millions tweets are produced every day, which demonstrates the problem of the lack of a good benchmark corpus for event extraction/detection from Twitter.

The work of Ritter et al. (2012) presented firstly seems to be one of the most cited papers on event extraction from Twitter, and is of course of interest for our own project. The authors bring up and discuss several of the challenges that Twitter poses for researchers trying to extract information from it, thus making it a valuable read. It does, however, focus on extracting structured representations of events mentioned in Twitter, as opposed to our own goal of detecting news events of current interest without concern for the whens and wheres of the actual real-life events.

The work of Zhou et al. (2014) is interesting because they obtain good results without using supervised learning, something that is beneficial considered the volume of unlabeled data available from the Twitter stream. Their learning approach, however, is not done in an online fashion and relies on some pre-configured parameters, such as number of events. This makes it unsuitable for a direct application to the real-time Twitter stream. It is also noteworthy that their system was not tested on larger amounts of data.

2.1.3 Event Detection Based on Term Pivoting

Kunneman and van den Bosch (2014) aims at detecting and separating tweets mentioning significant events from those mentioning everything mundane and insignificant. To do this they base their work on an event detection method previously proposed by Qin et al. (2013) which again build on the segment-based event detection method of Li et al. (2012). In short, what this method does is to split each tweet into unigrams and, for each time window, cluster segments that show a bursty frequency pattern and are similar in terms of content into one cluster. Kunneman and Bosch apply a Hidden Markov Model to uncover the most likely sequence of states for each unigram (bursty/not-bursty) and use the Viterbi algorithm to dynamically find the bursty states for each unigram. The clustering is done by applying Jarvis-Patrick clustering (with a small modification) to a similarity graph generated by taking the cosine distance between the unigrams represented as pseudo-documents containing TF-IDF weighted term vectors for all tweets in which the unigrams occurred.

The resulting clusters contain any types of events, so the significant events must subsequently be separated from the insignificant ones. To do this the authors represent the clustered events with a set of cluster-level features and presents event filtering as binary classification problem which is then solved using the Winnow classification algorithm (Littlestone, 1988). All event clusters that are positively recognized as ”true events” by the classifier form the returned news event set. Most of the features they use are taken from Qin et al. (2013), and divided into two subsets: Cluster Features and Tweet Features. Both subsets contains several numerical sub-features – some of them more interesting than the others, such as the wiki feature which represents the average newsworthiness of all unigrams in the cluster and is calculated by taking into account appearances of the unigram as anchor text in Wikipedia articles.

To evaluate the system they let human annotators label the 1,000 events that was most confidently classified as significant by the system. They then use this dataset to report extremely good results (F-score in the 85-95% range). It is, however, hard to say how good their system *actually* performs compared to similar systems, as long as the training and testing is done on an already ranked output of their own classifier. Nonetheless, the approach let them compare different feature selections for classification, and they find that the number of event tweets in the clusters and the number of URLs within these tweets are important features for recognizing significant events, while tweeters' tweet frequencies, newsworthiness and similarity values are not as important as reported in Qin et al. (2013).

For us, the most interesting part of this work is the part regarding classification of events as significant or not, as this is one of the problems we seek to solve in this thesis. The authors use the definition that *"something is significant if it may be discussed in media"*, which is in line with our own definition. The authors stress that this is a somewhat loose definition that is vulnerable to subjective interpretation. In addition they emphasize that different granularity levels for extracted events make comparison between previously proposed methods very difficult. Once again the lack of mutually agreed definitions, benchmark datasets and evaluation frameworks proves itself to be a problem for evaluating event detection in Twitter.

2.2 Other Related Work

Beside the aforementioned papers, many other works have also been done on event detection and/or extraction from Twitter and other microblogs. Agarwal et al. (2012) make a four-step method for extraction of local news events from Twitter, focusing on the specific event types *fire-in-factories* and *labor-strikes*. They combine regular expressions based on prior domain knowledge and supervised classification to detect candidate events, which in turn are clustered using LSH. Aspects of events are then extracted from each cluster of tweets and used to merge clusters based on properties such as time of occurrence and location, before information regarding the events is extracted using NLP-based information extraction techniques. Some interesting aspects of this particular work is that they manage to reduce tweet-to-tweet comparisons by 80% by using LSH in the clustering stage, and that they are able to detect sparsely reported events by using supervised classification of individual tweets.

Another work is Zhou et al. (2015) which builds on Zhou et al. (2014), but differs in the way that the LEM model is extended to also perform unsupervised categorization of the extracted events. This is done by grouping events into categories which all are assigned with an event type label without manual intervention. In addition they use a keyword lexicon built from news articles, published in the same period as the tweets, to filter out tweets not related to news events. Opposite to many other previous works, the authors actually apply their system to large datasets (60 millions tweets) and they are able to obtain good results, beating TwiCal (Ritter et al., 2012) by 6 pp.

The former work is interesting as it, in contrast to many other works, is focused on detection of sparsely reported events, while the latter is interesting as it poses some interesting ideas on how to make systems for event detection and categorization fully unsupervised.

2.3 Summary

As the review of related works in the preceding pages shows, there is a great deal of variation in the tasks the researchers of the field has aimed to solve. Petrovic et al. (2010), Agarwal et al. (2012), and McMinn and Jose (2015) are the only ones of the mentioned works that focus on new event detection (NED), i.e. the discovery of new events in near real-time. The other works are focused on retrospective event detection (RED), which typically involves iterative clustering algorithms, or other approaches that are dependent on seeing the full document (tweet) collection, before being able to group and detect events. Only Petrovic et al. (2010) explicitly apply their system to the real-time Twitter stream, while the others experiment with and evaluate their solutions on previously collected tweets. There is also a difference between the works regarding whether they address *detection* of events or *extraction* of events. Ritter et al. (2012) and Zhou et al. (2014) aim at extraction, which typically includes the extraction of structured representations of events taking place at specific times, as opposed to the other works which simply aim at detecting any mentioned event.

As our objective is detection of *news events* in the real-time Twitter stream, i.e. unspecified (as news events more or less are) NED, McMinn and Jose (2015) is the work most closely related to ours.

What our review shows is that there still is a great amount of research left to be done in the field of event detection from microblogs – especially when it comes to the real-time detection task. Based on the reviewed papers, as well as those we have read, but chosen not to include in this survey, it is clear that the literature contains large variations in both the objectives and in ways of evaluating the solutions. To conclude this survey, we would like to remark that the research community probably would have benefited from a commonly agreed research agenda for microblog IR, like the TDT project was for newswire IR. Table 2.1 shows a summarization of the reviewed works.

Table 2.1: Summary of related works reviewed in Chapter 2

Work	**Event type**	**Detection task**	**Magnitude**
Petrovic et al. (2010)	Unspecified	NED	∼164.5M tweets
Main idea(s):	Clusters “stories” by using LSH. Generate “threads” by using cosine similarity between tweets. Fastest growing threads are outputted in each time-interval.		
McMinn and Jose (2015)	Unspecified (news)	NED	∼120M tweets
Main idea(s):	Detects bursting named entities and creates events by clustering tweets where such entities co-occur based on cosine distance to nearest neighbor.		

Work	Event type	Detection task	Magnitude
Ritter et al. (2012)	Unpecified	RED	∼100M tweets
Main idea(s):	POS-tagging, NE extraction, temporal expression extraction, and event phrases extraction are done using supervised methods. These features are used to classify event type using an approach based on latent variable models. New events are predicted using a streaming approach to inference. Assumes that significant events have a common point in time to which multiple tweets refer explicitly.		
Zhou et al. (2014)	Unpecified	RED	2499 tweets
Main idea(s):	Non-location entities, locations, dates, and candidate keywords are extracted and used as input to an fully unsupervised LEM model for event extraction. Utilizes that the same event could be referenced by high volume of messages and group similar events based on co-occurrence patterns of the elements forming the event.		
Kunneman and van den Bosch (2014)	Unpecified	RED	∼65.2M tweets
Main idea(s):	Detects events by clustering bursty unigrams. The clustering favors unigrams that are mentioned with comparable content and are most bursty within the same time window. Insignificant events are filtered out from the significant ones by using a supervised binary Winnow classifier, taking both cluster-level and tweet-level features from the detected events as input.		
Agarwal et al. (2012)	Specified	NED	∼8M tweets
Main idea(s):	First classifies tweet as relevant or not to the specified target event using supervised classification and regex filtering. Relevant tweets are clustered in event clusters using LSH, before clusters with similar event aspects are merged together.		
Zhou et al. (2015)	Unspecified	RED	∼60M tweets
Main idea(s):	Filters tweets based on keywords found in lexicon built from the newswire around the same time the tweets are published. Groups events into categories, by using an LEM model, and automatically assigns event type labels.		

Chapter 3

Background

As presented in the introduction, the main contribution of this thesis is related to detecting *news events* formed by *news tweets*. This chapter provides some background information for the scope of this task, and presents some of the challenges that are posed by Twitter. Section 3.1 presents the main challenges of working with Twitter data, and why it is different from traditional text. Section 3.2 presents some different methods for representing text. Section 3.3 gives a brief introduction to tweet clustering, while lastly, Section 3.4 provides a short introduction to classification of tweets and some insight to artificial neural networks and deep learning.

3.1 The Challenges of Twitter

Information retrieval (IR) is a field of Computer Science that has been thoroughly studied for many years, dealing with search, manipulation and representation of electronic documents and other information-carrying medias. To be able to satisfy the information needs of the users, several sub-problems have arose, such as text classification, event detection, topic tracking, entity recognition and query expansion. And, of course, many approaches have been suggested to solve each individual problem. The research area perhaps most closely related to this thesis' objective of *news event* detection, is the Topic Detection and Tracking (TDT) project (Allan, 2002) mentioned introductorily. This initiative had for its objective to provide core technology for news monitoring tools, and one of the tasks was to detect new (unforeseen) events. The research fostered many good systems for detection and tracking of topics in full-length news texts, many of them presented in Allan (2002). Unlike tweets, the news texts that the TDT project was based upon were typically edited and content rich, something that made the application of traditional IR approaches such as *term vectors* or *bag-of-words* for data representation, or lemmatization and stemming, stop-word removal, and tokenization for document pre-processing effective. When dealing with tweets one can, unfortunately, no longer assume that words are spelled as in the dictionary, that capitalized words express names or that the writer follows strict grammatical rules. Neither do we know if a tweet is of importance to anyone except the followers of the tweeter, in contrast to the news wire, where we can assume that the documents must be of some importance to the public. All these aspects contribute to making event detection in Twitter much harder than in the classical TDT setting. In the following paragraphs some of the most pressing challenges of working with Twitter data will be explained in further depth, and a few approaches from the literature taken to overcome them will be presented.

3.1.1 Noise

Twitter users tend to use a language that is similar to the language known from SMS. Capitalization is often omitted when it should not have been, or it may be used to strengthen expressions by capitalizing whole words or sentences. Other times one can get the impression that it is used almost on random. The same goes for punctuation. Precise use of punctuation is almost non-existing in Twitter, and it is often used in unconventional ways, such as for emphasizing words or creating emoticons. The 140-character limit of tweets also causes the users to use incomplete sentences and untraditional sentence structures to fit their messages. In addition to this, abbreviations and slang are commonly used. This makes the use of dictionary based methods for extraction or representation of information difficult.

Some work has previously been done on normalization of SMS text, such as Kobus et al. (2008), where machine translation and phonetic similarity between words are used to transform SMS text into more standard form of English. Similar systems have also been investigated for Twitter, such as in Han and Baldwin (2011) where the authors propose a method for identifying and normalizing ill-formed words by using a classifier for the detection task and generate correction candidates based on morphophonemic[1] similarity, taking both word similarity and context into consideration. Although applying normalization techniques like these may make tweets more applicable as input to classical NLP tools, any system taking this approach will be more prone to error propagation, as the performance of the subsequent parts of the system always will depend on the normalization stage performing well.

3.1.2 Sparsity

Sparsity is another pressing matter when dealing with tweets. As tweets are limited to 140 characters and contains 10.7 words on average[2], while the English language contains over one million words[3], it is obvious that representing them as a simple term vectors will be unfeasible due to the enormous sparsity. Similarity between two such vectors would simply never be accurate because the term overlap would be too small. Denser representations of tweets are therefore needed to be able to apply e.g. machine learning methods to various information retrieval tasks in the Twittersphere. Problems like targeted event detection or monitoring of specific topics can be solved by hand-crafting feature vectors like occurrences of pre-specified words, but such approaches will not scale well to the streaming setting nor adapt to the ever evolving Twittersphere.

[1]"...the study of the relations between morphemes and their phonologicalrealizations, components, or mappings." (http://dictionary.reference.com/browse/morphophonemics)"

[2]http://blog.oxforddictionaries.com/2011/05/short-and-tweet/

[3]http://www.languagemonitor.com/number-of-words/number-of-words-in-the-english-language-1008879/

One approach that has been proposed to overcome the sparsity issue of short texts, where the contextual information is insufficient to provide enough semantic substance, is to use explicit semantic analysis (ESA) to enrich the text representation. Gabrilovich and Markovitch (2006) creates *bag-of-concepts* extracted from Wikipedia, instead of using only bag-of-words generated from the document, and compute the similarity between documents by using these enriched representations. Another potential approach to overcome the sparsity issues of short texts is to make representations based on word embeddings, a concept that will be covered later in this chapter.

In the specific case of news event detection in Twitter, sparsity also reveals itself as a problem considering how small the amount of tweets referring to news events is, compared to the tremendously high number of irrelevant tweets (spam and mundane events). Filtering irrelevant tweets from the stream could be one approach to counter this problem, and we will get back to this in the next paragraph.

3.1.3 Spam and Mundane Events

As mentioned earlier, there is no way to tell whether specific tweets are relevant to the public or only to the closest ones of the tweeter (or perhapsly not even to them). There is of course the possibility of manually specifying which users to monitor and assume that all tweets from these users are relevant. For the task of news detection, examples of aforementioned users could be news agencies, journalists, political activists and others, but by doing so, only news reported through already known news channels would be detected – i.e. old news. Filtering of spam and tweets describing mundane events are easier in the case of targeted event detection, where good results have been obtained by simply tracking relevant keywords (Sakaki et al., 2010). For news detection it is hard though, to know beforehand which words that will be used to describe future news events, and it is reasonable to assume that such as list of tracked words, if even possible to create, would become very big.

Twitter also presents us with extensive metadata for every tweet in the stream, such as number of retweets and mentions of other users. A plausible assumption is for example that a high number of retweets indicates relevancy to the public. In that way the metadata may be exploited to help filtering out irrelevant tweets. Unfortunately, the number of retweets can not be used in the real-time setting, as the tweets are received immediately upon publication and could impossibly have been retweeted yet.

Another problem with Twitter data, is the huge amount of spam. Forbes Tech analyzed the followers of the top 25 accounts on Twitter (roughly 2-7% of all users) in 2013, and found that 42.44% of the users were fake.[4] Several methods for filtering out spam among events in Twitter are found in the literature, such as taking entropy into account (Petrovic et al., 2010) or using strict filtering on number of mentioned users, urls and hashtags in tweets (McMinn et al., 2013).

[2]http://blog.oxforddictionaries.com/2011/05/short-and-tweet/

[3]http://www.languagemonitor.com/number-of-words/number-of-words-in-the-english-language-1008879/

[4]http://www.forbes.com/sites/tristanlouis/2013/04/07/twitters-growing-spam-problem/

3.1.4 Data Availability

Although Twitter makes their data very easily accessible through their Streaming and Search APIs, there are certain limitations. The Streaming API, which is the one relevant for this thesis, only provides roughly 1% of the total public Twitter stream. In addition to this, the 1% sample is not completely random. For instance, if sample *S* contains a tweet from user *A*, the probability that the sample contains other tweets from user *A* is reduced, as well as the probability that sample *S* contains tweets from user *A*'s network (i.e. friends, followers) (Efron, 2011). These are important aspects to bear in mind when using the Twitter Streaming API for research purposes. Fortunately, Twitter let us filter the stream on properties such as location and language, so that we at least can make sure that the data we get are as relevant as possible for the tasks we are chasing.

As for data quantity, the roughly 1% offered through the Streaming API, is still a large amount of data arriving at a high pace. It equals around 52 tweets/sec (taking for one's basis that 500 million tweets are posted each day and that 10% of them are public (Makice, 2009)) that have to be downloaded, read and processed in some way or another.

3.2 Representing Text

For retrieving information from text, documents are typically transformed into suitable representations using one of many available modeling approaches. The next paragraphs give a short introduction to traditional language models, typically applied in event detection tasks, as well as newer concepts like neural language models and word embeddings.

3.2.1 Traditional Language Models

Traditionally data is often represented as *term vectors* or *bag-of-words* for event detection tasks, and the terms in the vectors are typically weighted using *term frequency - inverse document frequency* (TF-IDF). This approach is unfortunately prone to the "curse of dimensionality", a problem that is even amplified when working with Twitter data, considered the vast amount of OOV (out of vocabulary) words and noise it contains. Some alternative data representations, found in the TDT literature, are the named entity vector (Kumaran and Allan, 2004) and the mixed vector (term + named entities) (Yang et al., 2002), as well as language models and other probabilistic methods incorporating both content and temporal information. Another problem with word-count-based approaches is that they do not take into consideration the context of the words, i.e. the spatial properties of the words in a sentence. Some have solved this by combining n-grams with count-based approaches to better capture the semantic meaning of both words and expressions in text, with the drawbacks it naturally carries in terms of even larger dimensions and increased computational costs.

3.2.2 Neural Language Models

During the last few years, however, a new generation of distributional semantic models have come into being, where optimization of the weights in a word vector is framed as a supervised task. I will refer to these models as *neural language models*[5] (NLM) Opposite to traditional construction of context vectors where context vectors are first created and then re-weighted using various weighting schemes and dimensionality reductions, the NLM models aim to optimize the weights to predict a target word's context. By exploiting the fact that similar words occur in similar contexts, the model learns similar vector representations of similar words. Baroni et al. (2014) perform the first (and only) extensive comparison of NLM to its count-based counterparts, and find that NLMs deliver superior performance compared to count-vector-based distributional semantic approaches.

Word2vec Mikolov et al. (2013) presents a method for efficient, unsupervised computation of continuous vector representations of words, and the result is the open-source toolkit named *Word2vec*. This toolkit is the one used for NLM in the comparison paper mentioned in the preceding paragraph. Mikolov et al. actually presents two different learning algorithms in their paper, namely *distributed skip-gram* and *CBOW*, and they are both implemented in the *Word2vec* toolkit. The CBOW model learns to predict a target word in the middle of a symmetric context window, while the distributed skip-gram model learns to predict a target context (i.e. the surrounding words) of a given word. According to Mikolov himself[6], Skip-gram works well with small amounts of training data, and represents even rare words or phrases well, while CBOW is several times faster than Skip-gram and obtains slightly better accuracy for frequent words. This makes sense as more training instances can be created from limited amount of data for the Skip-gram method (create contexts), while more data is needed for the CBOW method as it depends on surrounding contexts. Both models are "shallow" neural models, trained using stochastic gradient descent by backpropagation, with the output word probabilities computed either by *hierarchical softmax* or by *negative sampling*. We will not go in further depth on the details of the algorithms here, but a thorough explanation can be found in Rong (2014).

Mikolov et al. (2013) shows that the *Word2vec* models are able to recognize similarities among words, complex relationships (e.g. countries and capitals), analogies and word classes, as well as many linguistic regularities. They also discover that the employment of algebraic operations in the vector space produces meaningful results. For example will the simple algebraic operations `vector('king') - vector('man') + vector('woman')` result in a vector very close to the vector representation of the word `'queen'` (Mikolov et al., 2013). Perhaps is it not so strange that Jeremy Howard, previously President and Chief Scientist at Kaggle, proclaimed Word2vec to be *"the crown jewel of natural language processing"*. He also said that *"it's the English language compressed down to a list of numbers."*[7] And it is indeed impressive that we by using the super-efficient *Word2vec* toolkit can create such an information-rich, dense representation of words just by feeding it with sufficient amounts of (unlabeled) data. And, as we know, Twitter offers an extreme amount of unlabeled data.

Another great thing about *Word2vec*, and neural language models in general, is that they can be fed with any language, without being dependent on pre-built dictionaries or

[5]Also known as embeddings or context-predicting models.

[6]`https://groups.google.com/forum/#!searchin/word2vec-toolkit/c-bow/word2vec-toolkit/NLvYXU99cAM/E5ld8LcDxlAJ`

[7]`https://gigaom.com/2013/08/16/were-on-the-cusp-of-deep-learning-for-the-masses-you-can-thank-google-later/`

hand-crafted grammatical rule sets. This is very beneficial for use in the Twittersphere, as it means that one can quite simply input tweets from the Twitter stream straight into the learning algorithm of the NLM and it will automatically learn the "language" that characterizes Twitter. Let us look at the "abbreviation and slang problem" as an example: The word "birthday" can be expressed as "b-day," "bday" or "bd" in Twitter, but it will usually be used in the same contexts. This means that they will be assigned similar vector representations by *Word2vec*, and the model thereby learns that these words mean the same without any use of comprehensive human-made lookup tables or advanced NLP methods.

Paragraph Vectors While *Word2vec* is limited to representing only singular words, Le and Mikolov (2014) propose a method they call *Paragraph Vector* where the *Word2vec* model is adopted to create an unsupervised algorithm for learning a fixed-length feature representation from variable-length pieces of text. In addition to providing a real-valued dense vector representation of text, this method also solves the problem of varying length that we face whenever dealing with text, thus making it possible to apply algorithms that require fixed-length inputs such as SVMs or neural networks to the data. They evaluate the method on the task of Semantic Labeling, but the approach is likely to be applicable to other research areas as well. While this method looks interesting, it has been subject to some discussion as no one has seemingly been able to successfully reproduce the results from the paper. In fact, not even Tomas Mikolov, previous member of Google's Brain team, currently research scientist at Facebook AI Research[8], and co-author of the paper, was able to reproduce its results[9]. Even though no-one has been able to reproduce the exact results, the Paragraph Vector method still seems very promising for dense representation of short texts and the results other people report are usually in line with results from SoA solutions.

3.2.3 Another Approach to Word Embeddings

The publication of *Word2vec* lead to some follow-up works on word embeddings, with one of those attracting most attention being *Global Vectors for Word Representation (GloVe)* (Pennington et al., 2014). Despite Baroni et al. (2014)'s claim of the context-predicting models' thorough and resounding victory over their count-based counter-parts, Pennington et al. came up with *GloVe* and stated a 11% accuracy improvement over *Word2vec* in their initial publication. It was later shown that their evaluation section benefited *GloVe* and that when evaluated properly, paying attention to parameter settings, it shows performance similar to *Word2vec*.[10]

GloVe is, nevertheless, a kind of count-based model for word embeddings that is able to provide performance more or less (it depends on the task) as good as *Word2vec*. This is perhaps not so surprising, given that the two models in reality do almost the exact same thing, just by taking different approaches. Pennington et al. identified a matrix which

[8]https://research.facebook.com/researchers/643234929129233/tomas-mikolov/

[9]https://groups.google.com/forum/#!msg/word2vec-toolkit/Q49FIrNOQRo/ToAU2sYrPjYJ

[10]https://docs.google.com/document/d/1ydIujJ7ETSZ688RGfU5IMJJsbxAi-kRl8czSwpti15s/edit

when factorized using the SGD algorithm of *Word2vec* yields exactly the word vectors and context vectors that are found by the SGD backpropagation algoritm in *Word2vec*.[11]

3.3 Clustering

Clustering of tweets is a favorable task to solve, given the enormous amounts of unstructured information regarding all kinds of topics found in Twitter. As a consequence of this many works have been done on the topic. Clustering of the Twitter stream can be useful for a number of applications, such as sorting the tweets on topics so that a user can choose to see only the topics that interest him or her, finding trends, or as in our case, detecting events. Most approaches to tweet clustering uses a combination of traditional text clustering features and Twitter-specific features (hashtags, mentions, etc.)(Tsur et al., 2013), while others exploit information from elsewhere such as Wikipedia and the content of linked pages to add more substance to the tweets (Tang et al., 2012; Tripathy et al., 2014; Li et al., 2012).

Just as there is a wide range of features used for clustering, there is also a wide range of clustering algorithm that can be used. One of the most commonly used clustering methods is the *K-means* algorithm both in general and for the specific case of tweet clustering(Tsur et al., 2013; Tripathy et al., 2014). This is due to its relatively low computational cost and its good repute performance-wise. In the streaming setting of Twitter it is natural to use its descendant, the online K-means algorithm. These algorithms do however demand a pre-specified number of clusters, something that makes them less fit for clustering tweets on events, as there is no way to know in advance how many events that will occur within a certain time frame.

The clustering algorithm described in Petrovic et al. (2010), presented in Section 2.1, where tweets are clustered with its nearest neighbors within a certain distance, solves the problem of unknown number of events and gives an easy way of adjusting the granularity of the separate clusters.

3.4 Classification

Just as clustering is a favorable task to solve to group the information in Twitter, classification can be helpful to categorize all this information. Traditional classifiers like Random Forest has for instance been applied with good success to the problem of separating spam tweets from authentic tweets (McCord and Chuah, 2011). Another area of application is to use classification to decide topics of either single instances (Agarwal et al., 2012) or clusters of tweets (Hua et al., 2013). In the context of event detection, the former is typically used for specified event detection while the latter is typically used for unspecified event detection. The real-time aspect of this thesis call for a certain performance of a classifier when it comes to computation time. To be applicable to the Twitter stream, the classifier must be able to classify tweets at the same rate that they arrive. In our preliminary work (Repp, 2015), we found that a classifier based on an artificial neural network (ANN) offers

[11] `http://rare-technologies.com/making-sense-of-word2vec/`

this performance, as the classification can be done in constant time by using a pre-trained network.

3.4.1 Artificial Neural Networks

In short, ANNs are layered nonlinear models for machine learning, inspired by the workings of the neuron system of humans and other living organisms. The layers consists of neurons that are connected to the neurons in the other layers in some way or another, depending on the designer. Each neuron has an activation function that decides its output to the next layer based on the received input from its connected neurons in the preceding layer. The first layer, which receives the input, is known as the input layer, and the final layer, which outputs the processed data, is known as the output layer. Any layers between are known as hidden layers and can contain any number of neurons.

A Little History

The history of neural networking goes as far back as to the late 1800 with scientific studies of the human brain and William James' publication of the first work about brain activity patterns[12] . McCulloch and Pitts took this research further and produced a simple neural network using electrical circuits in 1943[13] . The evolution of artificial neural networks has come a long way since that time with some highlights worth mentioning: The invention of the perceptron algorithm (Rosenblatt, 1957); the introduction of recurrent neural networks with the invention of the Hopfield Network (Little, 1974; Hopfield, 1982); the application of the backpropagation algorithm to multi-layer neural networks (Werbos, 1974; Rumelhart et al., 1986); the invention of convolutional neural networks (Fukushima, 1980; LeCun et al., 1998); and most currently, the invention of the deep-belief net (Hinton et al., 2006).

Deep Learning

The term deep learning gained traction in the mid-2000s after the last-mentioned paper in the previous paragraph proved how many-layered feed-forward neural networks could be effectively trained using unsupervised layer-wise pre-training of stacked Restricted Boltzmann Machines and supervised backpropagation for fine-tuning. Today, when deep learning is spoken of in context of ANNs, it usually means neural nets with more than one hidden layer. Given enough training data, these models are able to learn internal features at different levels, a strongly desired ability in machine learning (ML), as it reduces the need for elaborate feature engineering.

The development of more powerful hardware, in particular GPUs to which the parallel nature of the deep neural network (DNN) maps naturally, has greatly accelerated DNNs success, and today top-performing systems are based on DNN in several machine learning tasks, such as image recognition and automatic speech recognition. Other mentionable

[12]`https://en.wikibooks.org/wiki/Artificial_Neural_Networks/History`

[13]`http://cs.stanford.edu/people/eroberts/courses/soco/projects/neural-networks/History/history1.html`

reasons for the success of DNN ML algorithms in later years are the ever increasing availability of data as well as the interest technology giants, like Google and Facebook, have taken in the methods.

When deep learning is applicable to a problem, it has been known to perform very well, often without the need for exhaustive parameter tuning. The exact reason why deep learning works as well as it does, is a frequently asked question whose answer is way beyond the scope of this thesis. A simple answer could, however, be that it is because of its ability to learn a better, internal representation of the data we feed to it. In this thesis we choose to investigate the application of deep learning to the problem of separating *news tweetes* from irrelevant tweets much because of its ease of use (little parameter tuning), but also because of its scarce presence in previous works on the matter. We saw from preliminary work (Repp, 2015) that a neural network is well suited for classifying tweets in a streaming setting, as prediction can be done in constant time with a pre-trained model.

Chapter 4

Approach

This chapter presents our proposed approach to detect news event in Twitter. Section 4.1 gives an overview of the approach. Section 4.2 explains how we use a classifier to separate news tweets from non-news tweets, and it shows experiments executed to find the best features and text representations. Section 4.3 shows how we use clustering to form news events, and it presents two novel clustering algorithms, as well as experiments conducted to find the best method and its corresponding parameter settings. Finally, the best approaches, revealed in the preceding sections, are assembled in a final system presented in Section 4.4.

4.1 Overview of Approach

The task of detecting news events in microblogs can be divided into three individual problems that each has to be addressed. The first problem is to detect the tweets that are relevant for one or another news event. The second problem is to group these news tweets into *news events*, i.e. detect clusters of tweets referring to the same event. The third and final problem is to identify which news events that are important enough to be reported and how to present them in a way that communicates the topics. Contrary to most other works being done on event detection in microblogs, we choose to firstly filter out irrelevant tweets (i.e. keep only *news tweets*) before forming *news events* by clustering the remaining *news tweets*. The rationale behind this is that the removal of all irrelevant tweets will lower the computational burden of the clustering step substantially. How to best do the classification and how to best do the clustering are, however, difficult research questions on their own. Therefore, in Section 4.2 and Section 4.3, we present experiments and results before giving an account for the choices made for the two components in the final system, respectively. The final system is described in its entirety in the last section of this chapter. Figure 4.1 shows a quick overview of the system and the components within it.

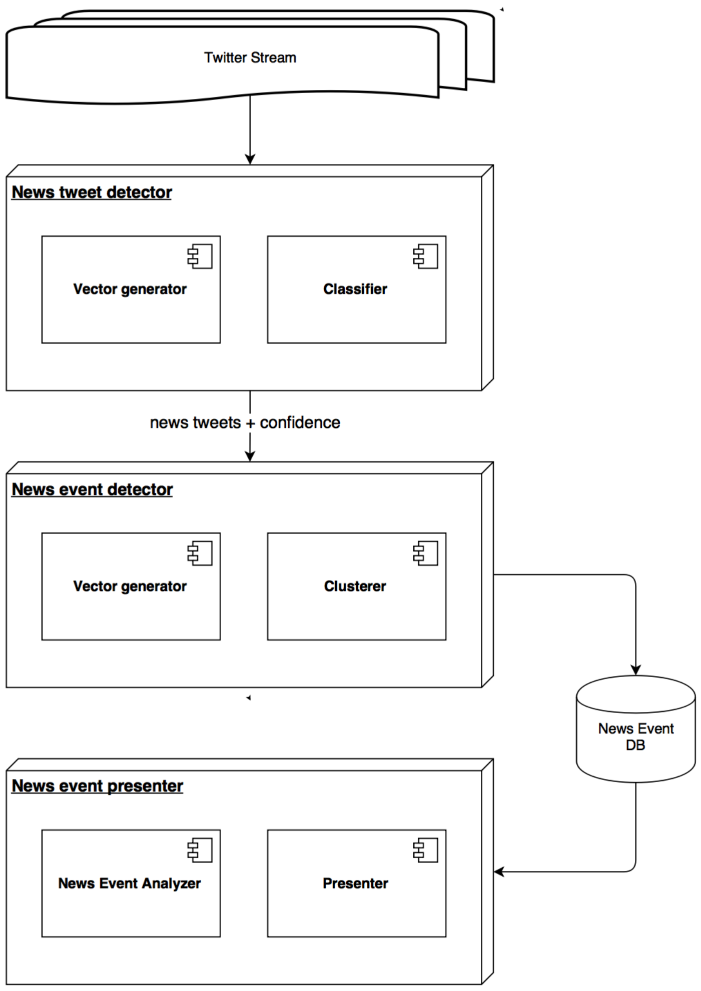


Figure 4.1: Overview of the components involved to detect news events from Twitter.

4.2 Detecting News Tweets

To detect which tweets that are actual news tweets, we choose an approach heavily leaning on machine learning and pose the problem as a binary classification task solved by using an artificial neural network (ANN). Preliminary work (Repp, 2015) showed that ANNs are well fit for classification in the streaming setting of Twitter, and we obtained good results by using average *Word2vec* vectors as input. Here we investigate other tweet text representations as well as the effect of including additional features for our news tweet classifier.

4.2.1 Tweet Text Representation

An important feature for deciding whether a tweet is relevant to a news event is the actual content of the tweet (i.e. the text). This text has to be represented in a way that an ANN "understands", or in other words, it must be transformed into a numeric vector. To do this we implement a vector generator interface that takes a pre-processed and tokenized tweet message as input and returns a real-valued, dense vector representation of the text. By doing so various ways of representing the tweets can easily be tested by using different implementations of the vector generator.

When dealing with Twitter short texts, we need a representation that is less vulnerable to noise than in the traditional IR setting. Our pre-study (Repp, 2015) showed that word embeddings were precisely that, when trained on in-domain data. A word embedding model trained on Twitter data is able to learn the special jargon often used in Twitter, and will produce similar representations of terms used in similar contexts. We believe word embeddings are the right mean to attack the noise challenge of Twitter texts, and therefore we base our suggested text representations on them. The specific vector generators we implement and investigate are as follows:

Average Word Vector (AvgW2V)

This method is based on the neural language model *Word2vec*, described in Subsection 3.2.2. We use a model pre-trained by Godin et al. (2015), consisting of a vocabulary of 3,039,345 words and 400-dimensional corresponding word embeddings. Godin et al. trained this model using Google's *Word2vec* toolkit on 400 million pre-processed tweets, collected between 2013-03-01 and 2014-02-28, using the Twitter Streaming API. The pre-processing included filtering out non-English tweets, and replacing user mentions, URLs and numbers with replacement tokens. Our proposed method to exploit word embeddings to represent tweets of arbitrary length is quite straightforward, as a tweet's text is represented simply as the mean of the word embedding vectors for the words in the text. The vectors are looked up in the dictionary of Godin et al.'s model. This means that a tweet text's vector representation will be of the same length as the word embeddings in the mentioned model, i.e. 400. Words that are not found in the dictionary are counted as zero-vectors, which means that they do not contribute to the mean vector.

Paragraph Vectors (D2V)

This method is based on the Paragraph Vector model presented in Section 3.2. We use the machine learning, NLP and data mining library *gensim*'s (Řehůřek and Sojka, 2010) implementation of the algorithm called *doc2vec*. In theory this representation should have an advantage over AvgD2V, as it captures the ordering of words in the text and therefore carries more information. We do however not have a pre-trained *doc2vec* model and therefore we have to train one ourselves, using Twitter data, before using it to generate tweet text representations. The training is done using gensim's implementation of the distributed memory algorithm (PV-DM) described in Le and Mikolov (2014), 20 full passes over the data (with randomization), and a learning rate starting at 0.025 and decreasing by 0.001 for each epoch. The dimensionality of the resulting vectors is 400, and a word's surrounding context is limited to the four closest words.

Average GloVe (AvgGloVe)

GloVe is, as mentioned in Subsection 3.2.3, very similar to *word2vec*. What is nice is that the GloVe developers offer freely available pre-trained word-vectors, generated from 2 billions tweets. The model vocabulary contains 1.2 million words and the embeddings are available as 25-, 50-, 100-, and 200-dimensional vectors. As opposed to the pre-trained model used for *AvgW2V*, here the words were lowercased at training time and hashtags and user tags were separated from their belonging words, which probably are the main reasons for the smaller size of this model's vocabulary compared to the one from Godin et al. (2015). We use the 200-dimensional vectors and average them in the same way as with the *AvgW2V* method. Words not found in the dictionary are represented as zero-vectors, which means that they do not contribute to the mean vector, just as in *AvgW2V*.

IDF-Weighted Word Vectors (IDFWV)

Term frequency–inverse document frequency (TF-IDF) weighting is used in the classic vector space model to incorporate the information value of words in the modeled text. In this method we combine this classical approach with word embeddings. Only IDF weights are used though, as TF is unlikely to provide any extra value given that tweets are so short (most words occur only once). The IDF weights are calculated from a training set of tweets and used to calculate a weighted mean over the word embeddings for the words in the tweets. Both Word2Vec and GloVe can be used as a basis for the word embeddings.

4.2.2 Pre-processing

The pre-processing of the tweets must be in accordance with the pre-trained models for representation. For *AvgW2V* this means that the steps taken in Godin et al. (2015) must be replicated. This includes replacing user mentions, URLs and numbers with replacement tokens. We use identical pre-processing steps for *D2V*, both in the training stage and when generating new vectors (the inference stage) as initial experiments showed us that including hashtags and capitalization yielded slightly better *paragraph vectors* than when omitted. We also believe that they could be important features for news tweets. The tokenization of the tweets are specialized towards twitter for both methods and retains tokens such as hashtags, urls, and emoticons. The specific implementation used is a Python port[1] of the Twokenize class from the CMU ARK Twitter Part-of-Speech Tagger (Gimpel et al., 2011).

The developers of the *GloVe* twitter model gives no clue towards what is done to the tweets before they are used to train the model, but fortunately GitHub user "manasRK"[2] has figured this out and offers a python script for transforming the *GloVe* model into a format usable by gensim's *word2vec* class as well as a tokenizer to tokenize tweets accordingly. We do nothing beside tokenizing the tweets using this tokenizer before feeding them to the *AvgGloVe* vector generator, leaving it to the pre-trained model to decide which tokens that will contribute to the resulting average vector. Note that in the *GloVe* model hashtags and user tags (@) are separated from their belonging words and constitute tokens on their own, something that may be unfortunate for our area of application.

Experiments conducted as a part of the preliminary work (Repp, 2015) showed that performance went down if we removed stop-words before creating the text representations for the ANN, an therefore we leave them in for all representations. This is probably because the ANN is able to utilize the stop-words, which are generally regarded as noise, to create better internal features, and with that they become an asset for solving the classification task.

4.2.3 Additional Features

Twitter includes extensive meta data in addition to the textual content of the tweets, containing information both about the tweet itself and its creator. We try to generate features out of the information that we think may be indicative of the tweet's news relevancy and append them to the text representation vector before feeding them to the ANN. The values of these additional features are scaled before they are inputted to the classifier, i.e. they are standardized to zero-mean and unit variance. The additional features investigated are shown and explained in Table 4.1.

Table 4.1: Explanations of features extracted from tweet meta data.

Feature name	Description
friends-followers ratio	Number of user's friends divided by number of user's followers.
followers count	Number of user's followers.
friends count	Number of user's friends.
statuses count	Number of user's previous postings.
hashtags count	Number of hashtags in tweet.
mentions count	Number of user mentions in tweet.
urls count	Number of urls in tweet.
text length	Number of words in tweet.

[1]`https://github.com/myleott/ark-twokenize-py`
[2]`https://github.com/manasRK/glove-gensim`

4.2.4 The Classifier

As mentioned, we treat the *news tweet detection* task as a binary classification problem (*news relevant/not news relevant*), which we aim to solve by using an ANN classifier. This classifier has to be trained before it can be used to predict classes of new data.

Training

The multilayer ANN is trained in a supervised fashion using tweets that are labeled either *event* or *not-event* and have been processed into one of the the previously described vector representations. While the Weka[3] tool was used for this step in the preliminary work, it is replaced with Keras[4] for this thesis for fast and easy testing and implementations of neural networks of various kinds as well as for code consistency reasons (to keep it pure Python). Keras is a minimalist, highly modular neural networks library capable of running on top of either TensorFlow[5] or Theano[6]. We use the Theano backend for this thesis as initial experiments showed that it performs marginally better than TensorFlow. A fully connected neural network using dropout (Srivastava et al., 2014) and rectified linear units (ReLu) constitutes the architecture of the classifier. Dropout is a regularization technique applied to artificial neural networks to prevent overfitting and speed up training by randomly leaving out units from the network throughout the training phase, while ReLU refers to units using the computationally cheap Rectified Linear Function as an activation function. The ANN consists of four hidden layer containing 400, 400, 200 and 100 neurons, respectively, each with a droput rate of 0.5. This architecture was found to work good in the preliminary work, although the exact depth and width of the network did not impact the performance to a very large degree. It should also be noted that when using a input layer based on word embeddings, it is basically like adding an extra hidden layer beneath the input layer which then becomes the input of words.

Prediction

Prediction is the task of “guessing” the class of previously unseen data, based on a previously learned model. We use the model trained as described above to predict whether or not an arriving tweet is a *news tweet*. The prediction step outputs a confidence score alongside the prediction itself, which makes it possible to easily adjust how strict the classifier should be. One may for instance limit the system to only return tweets that it predicts to be *news tweets* with a confidence higher than 90%.

4.2.5 Choosing Suitable Text Representations and Features

To decide what text representation and features that should be used for the classifier in the final system, we run a series of initial experiments using the dataset from McMinn et al. (2013). This dataset was downloaded as part of the preliminary work and consists of

[3]`http://www.cs.waikato.ac.nz/~ml/weka/`
[4]`http://www.keras.io`
[5]`https://www.tensorflow.org/`
[6]`http://deeplearning.net/software/theano/`

roughly 83 thousand tweets labeled as event tweets (there were more originally, but some tweets have been deleted from Twitter since the corpus was created), as well as about same amount of irrelevant tweets (i.e. *not* news tweets).

Each experiment is run using the same setup for the ANN: For the training the learning rate is set at 0.01 and the training is stopped when no change in error when evaluating on a 5000 tweet validation set has occurred over the last 10 epochs; stochastic gradient descent is used as an optimizer; and categorical cross-entropy is used for calculating the loss. Each of the experiments is run 20 times with the data set being randomly divided into training-, test- and validation sets for every new run. Each experiment is run on the same 20 selections, and performance is reported as a mean accuracy over all 20 runs. The sets of events represented in the training data and the test data are always disjoint, meaning that if the training data contains a tweet referring to a specific event, no tweets referring to that same event will be found in the test data, and visa versa. For the *D2V* method the *doc2vec* model used to generate text representations is pre-trained on the same data set that is used to train the ANN classifier.

Results and Findings

Text Representation: The results from training and testing the ANN using only tweet text as a feature, but with different ways of representing the text, are shown in table Table 4.2. The *D2V* representation is the lowest performing representation of the four. What this means is that the *D2V* representation is not particularly good at capturing the information necessary to separate a news tweet from a irrelevant tweet. It is hard to say for sure whether this is due to problems with the Paragraph Vector algorithm itself or due to insufficient amounts of training data. A potential problem with the algorithm could however be the fact that vector inferring (creating new vectors from previously unseen text) is not deterministic, i.e. if the same tweet is fed to the algorithm several times a different representation will be produced each time. The vectors should, however, be close to each other in vector space, so it should in theory not present itself as a big problem.

Next, we see that there is a large gap between the results from using the *AvgGloVe* representation and the *AvgW2V*, despite the theoretic similarity of these models. It is natural to think that *AvgGloVe* should have an advance over *AvgW2V* since its underlying model was trained on more tweets (2B vs 400M), but it is possible that this advantage is “eaten up” by its lower dimensionality compared to *AvgW2V* (200D vs 400D). It is also very possible that the inclusion of casing in *AvgW2V* works for its advantage when separating news tweets from irrelevant tweets, and the same goes for the inclusion of hashtags and user tags. Although the experiments conducted here is inadequate to say whether one model is generally better than the next one, *AvgW2V* works far better for capturing information necessary for the classifier to separate news tweets from irrelevant tweets than *AvgGloVe*, with the boundaries here imposed by using the underlying pre-trained *Word2vec* and *GloVe* models, respectively.

The experiments also show that the addition of IDF-weights to the *word2vec* embeddings yields no gain in performance compared to the considerably simpler *AvgW2V* method. We can actually observe a substantial performance drop. It is likely that the ANN is able to capture the importance of different words in the news detection task better than the IDF model, and that we by weighting the terms on beforehand only clutter up the

Table 4.2: Results from classifying event tweets using tweet text represented by different methods.

Method	Mean Accuracy
$AvgW2V$	**0.886**
$D2V$	0.623
$AvgGloVe$	0.684
$IDFWV_{w2v}$	0.847

classifier's input. It must also be said that this weighting slows down the vector generator substantially, as for each word in a tweet a weight must be looked up and element-wise multiplied with a 400-dimensional vector of floats.

Additional Features Table 4.3 shows that no single additional feature concatenated with the text representation helps on the classification task. In fact they all make the classifier perform worse. Only when applying them all at once can we see a slight improvement in accuracy. To investigate whether or not this difference is statistically significant we use a *McNemar's Test* on the predictions produced, as recommended by Dietterich (1998). We call the classifier using all the additional features C_+ and the classifier using only the text representation C. The predictions produced by these two classifiers give the following contingency table

$$\begin{array}{c|c} nn_{00} = 4613 & nn_{01} = 686 \\ \hline nn_{10} = 740 & nn_{11} = 43893 \end{array}$$

where n_{00} is the number of tweets misclassified by C and C_+; n_{01} is the number of tweets misclassified by C but not by C_+; n_{10} is the number of tweets misclassified by C_+ but not by C; and n_{11} is the number of tweets misclassified by neither C nor C_+.

Using the McNemar test statistic

$$\chi^2 = \frac{(|n_{01} - n_{10}| - 1)^2}{n_{01} + n_{10}}$$

which is distributed approximately as χ^2 with 1 degree of freedom, we get that

$$\chi^2 = \frac{(|686 - 740| - 1)^2}{686 + 740} \approx 2.12\,.$$

To be able to reject the null hypothesis, which in this case is that the two classifiers have different performance, at a 95% confidence level χ^2 must be ≥ 3.84 (Dietterich, 1998). This is not the case for our statistic, and therefore we can conclude that the better performance of the classifier taking the additional features versus the one taking only the *AvgW2V* representation, is not statistically significant.

Table 4.3: Results from classifying event tweets using *AvgW2V*-vectors concatenated with additional features.

Classifier input	**Mean Accuracy**
friends-followers ratio + AvgW2V	0.886
followers count + AvgW2V	0.886
friends count + AvgW2V	0.887
statuses count + AvgW2V	0.886
hashtags count + AvgW2V	0.888
mentions count + AvgW2V	0.886
urls count + AvgW2V	0.887
text length + AvgW2V	0.888
all of the above + AvgW2V	0.890
Only *AvgW2V*	0.889

Chosen Text Representation and Features

As a consequence of the results found in the above experiments, **we choose to use *AvgW2V* to represent the textual content of tweets** simply because it has the best performance on our test data, and because its relatively high accuracy indicates that it is well able to capture the important semantic of tweets. The fact that this representation scales linearly with the length of tweets (limited above by the number of words possible in a tweet), is also beneficial for our application. We also choose to **leave out all the additional features**, as the marginal increase in performance obtained by including them all was not statistically significant. Leaving out the Twitter-specific features also has the advantage of making the system less bound to Twitter, something that will make it easier to apply it to other stream types in the future.

4.3 Detecting News Events

A *news event* in the context of this thesis is a group of news tweets (as discovered by the classifier) that are referring to the same news-relevant topic. This makes detecting events a typical clustering task. The system must be able to perform this clustering in a more or less online fashion, i.e. it must be able to handle incoming tweets continuously and cannot be dependant on having all data before clustering. Another requirements is that it is sufficiently fast, so that it is able to keep pace with the stream of news tweets.

4.3.1 Clustering Methods

As the clustering must be done fast, and the number of clusters are completely unknown in advance, the clustering method proposed in Petrovic et al. (2010) is a good basis. We propose a couple of alternate variants of this method, one being completely online and the second being a mini-batch variant. We will refer to these methods as the online thread-cluster method (oTC) and the mini-batch thread-cluster method (mbTC) respectively. The thread wording is adapted from Petrovic et al. (2010) and refers to the fact that the methods grows threads[7] of tweets throughout time – threads which in our case correspond to *news events*.

Online Thread-Clustering (oTC)

This method performs online clustering of tweets by assigning the tweet to the same thread as its nearest neighbor if the cosine distance to that neighbor is below a given threshold, t. If the distance to its nearest neighbor is above the specified threshold, the tweet is assigned to a new thread. Only the last w tweets are considered when looking up the neighbors. The difference between this method and the one proposed in Petrovic et al. (2010) is that we do not use LSH to find the nearest neighbor. Because we only need to cluster tweets that have already been classified as *news tweets*, and thus heavily reduced the amount of data, we can afford to solve the true nearest neighbor problem instead of a relaxed version of it. Our implementation is done by using the *NearestNeighbor* class of the Scikit-learn package with cosine as distance function. Pseudo-code for the online thread-clustering method is shown in Algorithm 1.

Mini-Batch Thread-Clustering (mbTC)

What separates this method from the previous one, is that the tweets here are handled in batches of size b. For every tweet in a new batch, the tweet is assigned the same thread id as its nearest neighbor among the last w previously seen tweets, if the distance between them is below the threshold t. Next, another run is taken over those tweets in the batch that have *not* already been assigned to a thread. If they have a nearest neighbor within the batch that has already been assigned to a thread and the distance to that neighbor is below the threshold, the tweet is assigned to the same thread as the neighbor. If not, we create a new thread and assign the tweet to it. Nearest neighbors are found by using *NearestNeighbor* from Scikit-learn and cosine distance. The pseudo-code can be seen in Algorithm 2.

[7]Threads are synonymous to clusters, and both phrasings are used throughout this thesis

Algorithm 1: Pseudo code for the online thread clustering algorithm

```
input: threshold $t$, window size $w$
$T \leftarrow \{\}$;
while tweet in stream do
    $is\_duplicate \leftarrow$ False;
    if $T$ is empty then
        $thread\_id(tweet) \leftarrow$ new thread id;
    else
        $tweet_{nearest} \leftarrow$ nearest neighbor of $tweet$ in $T$;
        $d \leftarrow cosine_dist(tweet, tweet_{nearest})$;
        if $d \approx 0$ then
            $is\_duplicate \leftarrow$ True;
        else if $d < t$ then
            $thread\_id(tweet) \leftarrow thread\_id(tweet_{nearest})$;
        else
            $thread\_id(tweet) \leftarrow$ new thread id;
    end
    if $is\_duplicate$ is False then
        if $|T| \geq w$ then remove first tweet from $T$;
        add $tweet$ to $T$;
    end
end
```

Algorithm 2: Pseudo code for the mini-batch thread clustering algorithm

```
input: threshold $t$, window size $w$, batch size $b$
$T \leftarrow \{\}$;
$B \leftarrow \{\}$;
while tweet in stream do
    add $tweet$ to $B$;
    if $|B| == b$ then
        $duplicates \leftarrow \{\}$;
        if T is not empty then
            for tweet' in B do
                $tweet^{T}_{nearest} \leftarrow$ nearest neighbor of $tweet'$ in $T$;
                $d_T \leftarrow dist(tweet', tweet^{T}_{nearest})$;
                if $d_T < t$ then
                    if $d_T \approx 0$ then
                        add $tweet'$ to $duplicates$;
                    else
                        $thread\_id(tweet') \leftarrow thread\_id(tweet^{T}_{nearest})$;
                    end
                end
            end
        end
        for tweet” in B that has not been assigned to thread do
            $tweet^{B}_{nearest} \leftarrow$ nearest neighbor of $tweet''$ in $B$;
            $d_B \leftarrow dist(tweet'', tweet^{B}_{nearest})$;
            if $d_B < t$ and $tweet^{B}_{nearest}$ has thread id then
                if $d_B \approx 0$ then
                    add $tweet''$ to $duplicates$;
                else
                    $thread\_id(tweet'') \leftarrow thread\_id(tweet^{B}_{nearest})$;
                end
            else
                $thread\_id(tweet'') \leftarrow$ new thread id;
            end
        end
        if $|T| \geq w$ then
            remove first $|B - duplicates|$ tweets from $T$;
        end
        add all $B - duplicates$ to $T$;
        $B \leftarrow \{\}$;
    end
end
```

4.3.2 Time and Memory Requirements

For both of our previously described algorithms, the number of comparisons needed to assign a new tweet to a thread, is limited above by w. In addition to this, the dimensionality of each tweet's vector representation is constant. In sum, this means that the clustering algorithms will approach constant time complexity as the number of received tweets approach w – an important aspect when working in the streaming setting. The memory constraint of the streaming setting is also satisfied, as no more than w tweets will ever be contained in-memory at the same time. The clustering is a one-shot process for both algorithms, meaning that once a tweet has been assigned to a thread, it remains there forever.

4.3.3 Features

As explicit features for the clustering algorithms, only the textual content of tweets is used as it is here the semantic information lies. We use the same text representation as for the classifier, i.e. *AvgW2V*, for clustering, as opposed to Petrovic et al. (2010) who used TF-IDF term vectors. The time aspect is incorporated by only considering a limited number of the last received tweets when clustering a new one, as described by w in Algorithm 1 and Algorithm 2. One can regard this as a sliding window through time. What this means in practice, is that when no tweets have been assigned to a specific thread for the duration of one full window, that specific thread is "forgotten".

4.3.4 Choosing the best Algorithm

We run a series of experiments to decide which of the two algorithms that is best fit for the task of creating news event clusters. To do so we once again use the training corpus from McMinn et al. (2013), but this time we use only the tweets annotated as *news tweets* as this is what optimally should remain of the stream after passing through the classifier. In addition to testing the exact same representation that was found to perform best on the classification task, we try a version using a more aggressive pre-processing step that includes lowercasing the tweets and removing stop-words before the *AvgW2V* vectors are generated. The metrics that are used to assess cluster quality are *homogeneity*, *completeness*, and *v-measure* (Rosenberg and Hirschberg, 2007):

Homogeneity Satisfied if a clustering assigns only those datapoints that are members of a single class (event label ground truth in our case) to a single cluster. Bounded below by 0 and above by 1, where higher is better.

Completeness Satisfied if a clustering assigns all of those datapoints that are members of a single class (event label ground truth in our case) to a single cluster. Bounded below by 0 and above by 1, where higher is better.

V-Measure The harmonic mean of homogeneity and completeness.

(Rosenberg and Hirschberg, 2007)

Some of the advantages of using these specific metrics are that they are based on clustering "usefulness" and intuitively interpreted. They are also independent of the clustering algorithms, size of data sets, number of classes, number of clusters, and they not require us to map each cluster to a class. The latter means that only the quality of the clustering is evaluated, instead of a post-hoc class-cluster mapping.

All clusterings in the experiments are done using a window size of 2000, which means that the last 2000 tweets are considered when a new tweet is assigned to a cluster. The test set contains ∼54,000 tweets spanning 28 days, meaning that a window size of 2000 corresponds to roughly 25 hours worth of tweets (given an even distribution, which is unlikely).

Results and Findings

As we see from Table 4.4, both algorithm persistently perform better when the the more aggressive form of pre-processing is used. The reason that this is the case here, while opposite for the classifier (Repp, 2015), is probably that the filtered tweets that forms the input of the clustering step is of a higher quality (less noise) than the raw stream that the classifier receives, and therefore the dictionary based stop-word removal is more efficient. Also, the clustering algorithms do not learn which features that are important and not, like the classifier does, and therefore they are much more vulnerable to noise.

For *mbTC* there is a trend towards better performance for smaller batch sizes. The *mbTC* algorithm using a batch size of 50 does however yield better results than the purely online *oTC* algorithm for all threshold values except at $t = 0.25$, where *oTC* obtains the best observed score (0.774). We do however see in Table 4.5 that we can make the *mbTC* algorithm match the best performance of *oTC* by fine tuning its threshold value. Despite not having a lower complexity than *OTC*, the *mbTC* algorithm is also considerably faster due to less overhead in the implementation (e.g. lists expanded in batches and distances batch-wise calculated using fast matrix operations). The per-tweet clustering time for the two can be seen in Figure 4.2.

Table 4.4: Performance in terms of V-Measure for the different clustering methods by threshold value, t. The subscript of *mbTC* indicates the batch size that was used. The table includes results from the clustering run with and without the removal of stopwords. A window size of 2000 was used for all runs.

	Stopwords included					Stopwords excluded				
$t =$	.05	.15	.20	.25	.35	.05	.15	.20	.25	.35
oTC	.647	.721	.681	.594	.550	.646	.702	.748	**.774**	.665
$mbTC_{b=50}$	.648	.718	.589	.251	.105	.648	.710	.758	.763	.324
$mbTC_{b=100}$	.648	.709	.520	.274	.136	.647	.706	.749	.736	.295
$mbTC_{b=200}$	.647	.694	.512	.301	.176	.647	.699	.736	.721	.305
$mbTC_{b=400}$	.646	.678	.497	.307	.166	.646	.693	.721	.698	.274

Table 4.5: Performance of *oTC* and *mbTC* by threshold, t. The best results are boldfaced. A window size of 2000 was used and stopwords were removed before *AvgW2V* representations were generated.

		Threshold value						
Method	**Metric**	.22	.23	.24	.25	.26	.27	.28
	H	.958	.944	.929	.910	.888	.862	.833
oTC	C	.633	.648	.661	.674	.687	.695	.702
	V	.762	.768	.773	**.774**	**.774**	.769	.762
	H	.929	.912	.884	.843	.802	.762	.708
$mbTC_{b=50}$	C	.656	.672	.686	.696	.704	.717	.718
	V	.769	**.774**	.772	.763	.749	.739	.713

H = homogeneity; C = completeness; V = V-measure

Chosen Algorithm

Although both algorithms are well within the efficiency boundaries needed to handle the filtered Twitter stream, **we choose to use the *mbTC* algorithm in the final system**. By setting the batch size sufficiently small it will deliver close to immediate clustering of new tweets, and therefore correspond to only a slight yield in the requirement to assess events of new tweets continuously. It would in any event be natural to process the tweets batch-wise when writing them to persistent storage, so in practice, there will be little difference between using online- and mini-batch clustering in the final system.

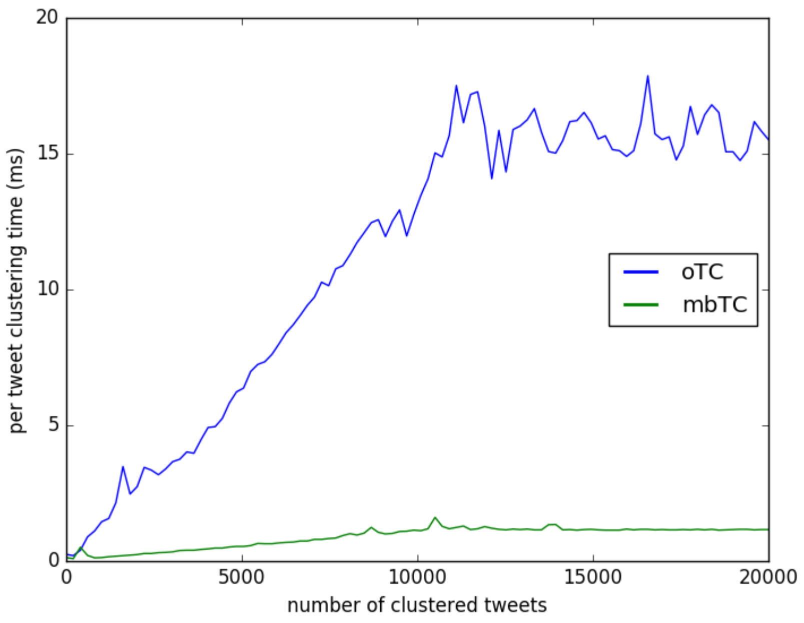


Figure 4.2: Per tweet clustering time for *oTC* and *mbTC* by number of tweets clustered. The window size is set at 10,000, which explains why the graphs flatten out at 10,000 clustered tweets.

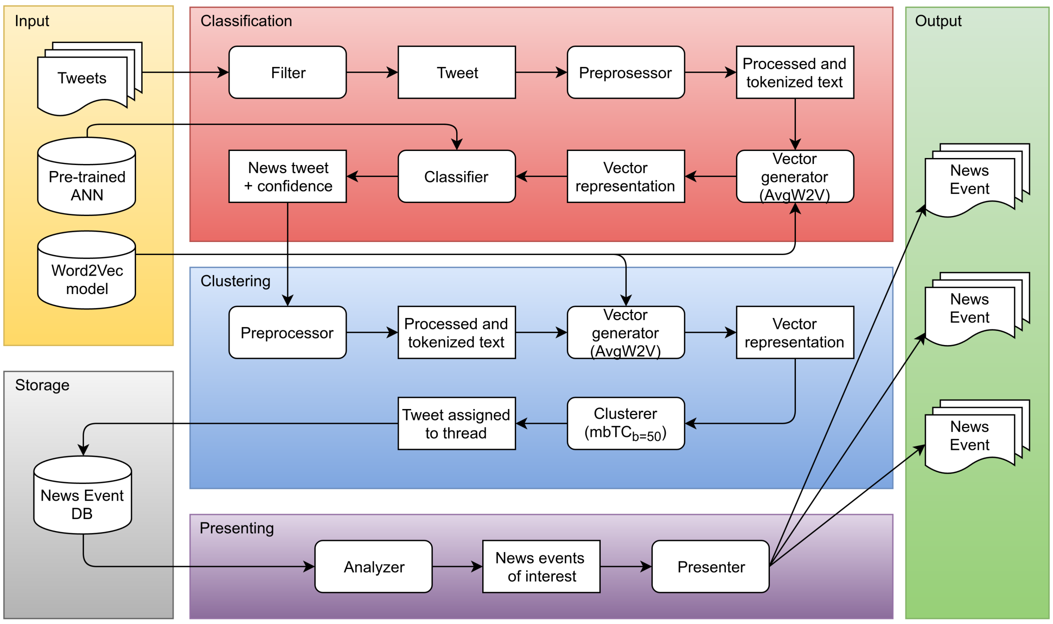


Figure 4.3: Overview of the final system.

4.4 Final System

Now that we have given an account for the methods, features and representations chosen, it is time to see how it all connects to form the final system for detecting and presenting *news events*. The complete system is displayed in Figure 4.3, and its workings are explained in the rest of this section.

4.4.1 Accessing and Filtering the Twitter Feed

To get access to the Twitter feed we use the tweepy[8] package for Python[9]. This is a Python wrapper of Twitter's APIs. The input of the system is the previously mentioned 1% freely available part of the public Twitter feed, often referred to as "the garden hose", which is easily accessible for everyone who signs up as a developer on `dev.twitter.com` and registers a new application. Fortunately, Twitter let us filter the stream on certain properties. As we limit the scope of this thesis to only deal with English tweets, the stream is set to only deliver tweets written in English. The stream is also limited to the geographical area that is the U.S. to make an eventual evaluation of the returned result easier, as one can assume that verification of real news can be done by inspecting the U.S. newswire only. The connection to the Twitter Streaming API is set up using personal Twitter Access Tokens for OAuth authorization[10], and the Twitter filter parameters that is

[8]`http://www.tweepy.org/`
[9]`http://www.python.org`

used (i.e. language and location) is set when the stream is initialized.

In addition we choose to discard some tweets in the filtering step to increase the quality of the tweets that are passed further down the system's pipeline. The discarded tweets are those that have features typically found in spam tweets, such as a large number of hashtags, URLs or user mentions, as proposed in McMinn et al. (2013), as well as retweeted tweets which only cause redundancy in the system. Like McMinn et al. (2013) we set acceptance limit for hashtags at 3, for urls at 2, and for user mentions at 3. All tweets that pass the initial filtering stage are passed on to the *news tweet* classifier.

4.4.2 Classification

Before the system starts to receive tweets from the stream, the *AvgW2V* model and the pre-trained ANN model is loaded into the system from disk. When ready, the system starts to process tweets from the stream. The text is tokenized as explained in Section 4.2 before fed to a vector generator to create the *AvgW2V* representation. The ANN then take this representation as input and predicts whether or not the tweet is an *news tweet*. If the tweet is recognized as an *news tweet*, the tweet as well as the prediction confidence is put in a queue, ready to be assigned to a *news event* in the clustering stage. If not, it is discarded without further action.

4.4.3 Clustering

When a *news tweet* arrives at the clustering stage, a new *AvgW2V* representation is generated, but this time we remove stopwords and urls, and we lowercase the text. The tweets are then clustered based on the *AvgW2V* vectors in batches of 50 using the *mbTC* algorithm. For every batch, every tweet in it is assigned either to an existing thread (cluster) or to a new thread in which it is the first instance. Every one of these threads correspond to a *news event*. The clustering process is run in parallel with the stream and classifier, and it picks and processes new tweets from the queue of news tweets populated by the classification module. In this way, the clustering process will never affect the system's ability to read the Twitter stream fast enough.

When all tweets in a batch have been assigned to a thread, the tweets, along with their classification confidence, thread id and meta data, are saved to an *SQLite*[11] database. The reason for choosing *SQLite* is simply because it is very easy to set up and use with Python, and for the time being, there is in reality no need for anything more advanced.

4.4.4 Presentation

When *news tweets* haven been detected and clustered together to form *news events*, we are still left with the challenge of choosing which events to display, for the system to have any use value. As all tweets are continuously stored to a database and therefore immediately available from outside the system, we decide to decouple the presenter completely from the rest of the system. In that way, the presentation layer can easily be tailored to suit different

[10]`https://dev.twitter.com/oauth`
[11]`https://www.sqlite.org/`

information needs. Different analyzer- and presenting modules for different objectives can be run at the same time from arbitrary servers and machines, as long as they have access to the database where the news events are stored.

Finding Events for Presentation

The presenter layer has an analyzer module that is used to find news events of interest by querying the database and analyzing the results. One can base these analyses on parameters such as time intervals, event impact (i.e. thread size), and the growth rate of threads. Petrovic et al. (2010)'s system for first story detection does for instance output only the fastest growing threads for each time interval. Our system can easily be set up to replicate this behaviour, or it can be set up to do something else, like returning the largest threads for the last 24 hours or the threads growing the most in the last 10 minutes, simply by modifying the analyzer.

Presenting the Events

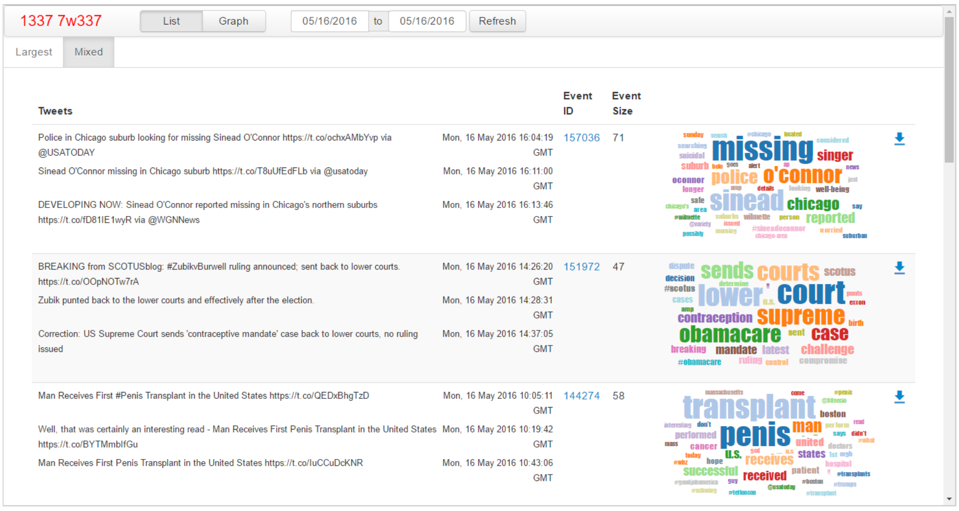
To display the detected *news events*, we use a web interface served by a backend built on the Flask[12] microframework for Python. *News events* are presented as a list of all the tweets constituting them as well as word clouds based on the content of all the tweets (see examples in Figure 4.5). The word clouds let the user quickly spot the most dominating terms in an event and thereby allows him or her to form an overview its topic(s). The user interface is not really a priority in this thesis, so, rather than a finished product, the proposed application should be regarded as a proof-of-concept and an example of how our method can be applied to create a news service for end-users.

Nevertheless, our application has a simple interface: It has the possibility to select a time range for which news events should be returned; the possibility to switch between two different views, where one presents a list of all events with corresponding word clouds and the other presents a line graph showing the events' evolution in the set time range; and it has the possibility to choose the type of analyzer used when fetching events for presentation. The line graph has perhaps a somewhat limited utility value, but we thought it was a cool addition. Examples of how our web application (codenamed *"1337 7w337"*) looks like can be seen in Figure 4.4.

4.4.5 Efficiency

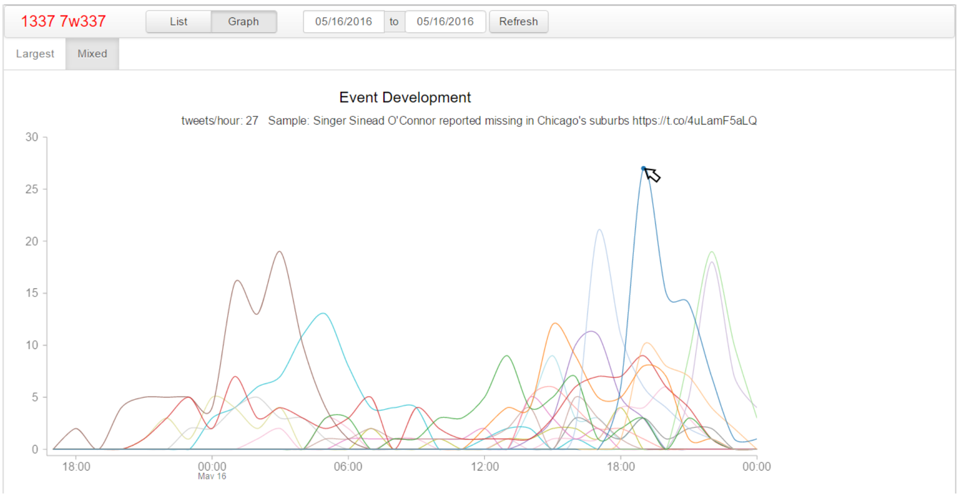
Figure 4.6 shows the efficiency of the the proposed approach, when the system is run on an ordinary desktop computer. The graph show that the classification is performed in constant time, while the clustering time grows linearly with the number of processed tweets until reaching the window size limit where it stabilizes. With a window size of 5000, as depicted in the graph, the average per tweet processing time is less than 4 ms. This is well within the boundaries required to handle the estimated average 52 tweets/sec (Section 3.1) delivered by the Twitter "garden hose". In a very unlikely scenario, where *all* tweets are detected as news tweets, the system will still be able to process over four times the average volume of the "garden hose".

[12]`http://flask.pocoo.org/`



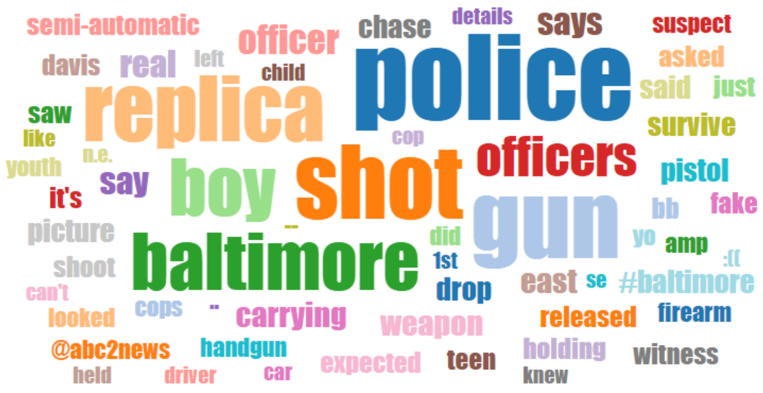
Tweets		Event ID	Event Size
Police in Chicago suburb looking for missing Sinead O'Connor https://t.co/ochxAMbYvp via @USATODAY	Mon, 16 May 2016 16:04:19 GMT	157036	71
Sinead O'Connor missing in Chicago suburb https://t.co/T8uUfEdFLb via @usatoday	Mon, 16 May 2016 16:11:00 GMT		
DEVELOPING NOW: Sinead O'Connor reported missing in Chicago's northern suburbs https://t.co/fD81lE1wyR via @WGNNews	Mon, 16 May 2016 16:13:46 GMT		
BREAKING from SCOTUSblog: #ZubikvBurwell ruling announced; sent back to lower courts. https://t.co/OOpNOTw7rA	Mon, 16 May 2016 14:26:20 GMT	151972	47
Zubik punted back to the lower courts and effectively after the election.	Mon, 16 May 2016 14:28:31 GMT		
Correction: US Supreme Court sends 'contraceptive mandate' case back to lower courts, no ruling issued	Mon, 16 May 2016 14:37:05 GMT		
Man Receives First #Penis Transplant in the United States https://t.co/QEDxBhgTzD	Mon, 16 May 2016 10:05:11 GMT	144274	58
Well, that was certainly an interesting read - Man Receives First Penis Transplant in the United States https://t.co/BYTMmblfGu	Mon, 16 May 2016 10:19:42 GMT		
Man Receives First Penis Transplant in the United States https://t.co/luCCuDcKNR	Mon, 16 May 2016 10:43:06 GMT		

(a) Events presented as lists of belonging tweets and word clouds. The first three tweets of each event is displayed in the list, but expanded inline to display more tweets if an event's row is clicked. The download buttons are to export each word cloud (SVG) as images (PNG), a feature that comes in handy when writing reports and creating questionnaires.

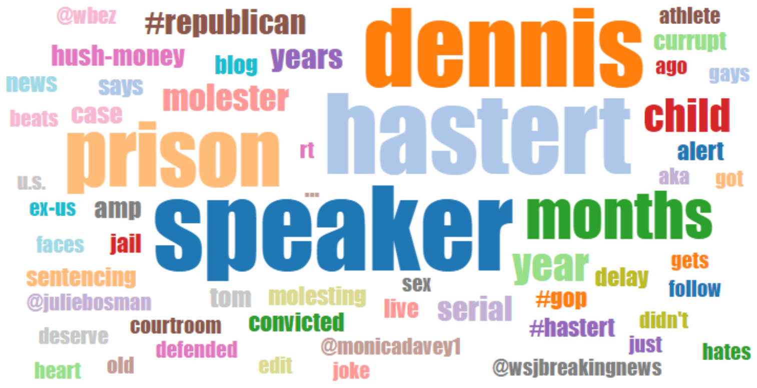


(b) Graph showing each events' development in time as posted tweets per hour. The number of tweets per hour as well as an example tweet from that hour is displayed when hovering over a line.

Figure 4.4: Screen captures showing our proposed web application for presenting detected news events.



(a) Related to an incident were a 13-year old boy with a replica weapon was shot by Baltimore police (April 27th).



(b) Related to former U.S. House Speaker Dennis Hastert being sentenced to 15 months in prison for illegally structuring bank transactions in an effort to cover up his sexual abuse of kids (April 27th). Not surprisingly, the focus in Twitter was on him being a child molester, although he was actually never charged with sexual abuse.

Figure 4.5: Examples of word clouds for news events detected by our system.

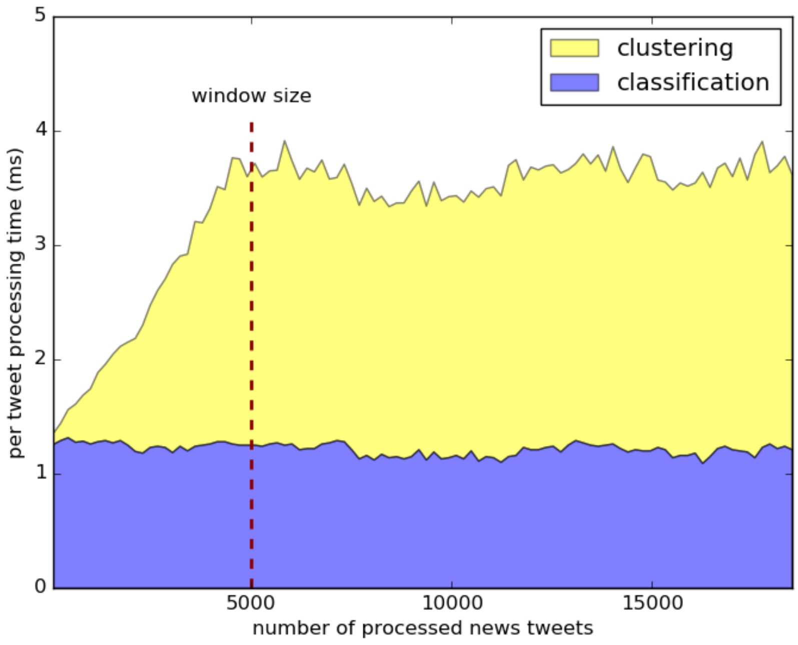


Figure 4.6: Per tweet processing time by number of news tweets processed.

Chapter 5

Experiments and Evaluation

This chapter describes the experimental setup used to perform an automated as well a user-based evaluation of the system proposed in Chapter 4. Section 5.1 describes the dataset that is used for the automated evaluation and as a basis for the classifier in the final system, and how this is obtained, while Section 5.2 describes in detail the system setups and how the evaluations are carried out.

5.1 Dataset

Obtaining a Twitter dataset is a challenging task, due to the restrictions that Twitter Terms of Services (TOS) impose on the re-distribution of tweets. This is one of the reasons why there are almost none good, publicly available datasets for the Twitter event detection task. Most of the previous works use datasets that they have downloaded through the Twitter APIs and labeled themselves. Twitter TOS do allow the re-distribution of tweet IDs and their belonging user IDs, so it is possible to publish labels and tweet IDs and leave it up to the next user to download the actual data, but even so, very few researchers choose to do this.

5.1.1 A Large-scale Corpus for Event Detection

Fortunately there are exceptions. McMinn et al. (2013) address the lack of publicly available corpora for evaluation of event detection approaches in Twitter. They created and published a tweet corpus with relevance judgements aimed specifically at the task of event detection. It was this dataset that we used to test the different components and setting of the system in Chapter 4.

McMinn et al. (2013) created the corpus by downloading a collection of 120 million English tweets using the Twitter Streaming API for 28 days, starting on 10 October 2012. They then retrieve candidate events from the collection using two different SoA event detection techniques (Petrovic et al., 2010; Aggarwal and Subbian, 2012). In addition they use events from Wikipedia's Current Events Portal[1] and retrieve tweets related to

[1] `https://en.wikipedia.org/wiki/Portal:Current_events`

Table 5.1: Event categories used in McMinn et al. (2013) and events per category in original corpus.

Categories	**Events**
Armed Conflicts & Attacks	98
Arts, Culture & Entertainment	53
Business & Economy	23
Disasters & Accidents	29
Law, Politics & Scandals	140
Miscellaneous	21
Science & Technology	16
Sports	126
Total	506

these from the tweet collection by using traditional query expansion and search techniques. The relevance of the resulting candidate events are then verified by having five annotators inspect them by using the crowdsourcing platform Amazon Mechanical Turk[2].

The resulting labeled corpus of McMinn et al. (2013) contains over 150,000 labeled tweets covering more than 500 different real-life news events. The event clusters are further divided into news categories that are generated based on the categories defined by Wikipedia and in the TDT project, but tuned to cover a much broader range than either of those two. The categories can be seen in Table 5.1, as well as the distribution of events over them. Although the *Events2012* dataset contains a very large number of events, we must point out that it does not guarantee full coverage of events nor that all tweets of every event is annotated.

McMinn et al. make their corpus publicly available by publishing the tweet IDs along with the news event relevance judgements, making it available for everyone for further research and development. The corpus will henceforth be referred to as *Events2012*.

Important Remark The tweets in *Events2012* are tweets related to any news event (Definition 1). This means that a tweet labeled as an event can be for instance a contribution to a discussion thread, or a single user's expression of opinion, regarding the corresponding event. More concretely, *the tweets labeled as events are not necessarily describing news events explicitly*. In the context of this thesis we regard all tweets labeled as events as *news tweets* (Definition 2).

[2]`https://www.mturk.com`

Obtaining the Corpus

The *Events2012* dataset was collected in the preliminary work (Repp, 2015) of this thesis. To retrieve the tweets in it (often referred to as "hydrating" the dataset) we implemented a tweet downloader in Python by using the *tweepy package* and the Twitter REST API's *lookup* call.[3] This allowed us to retrieve fully-hydrated tweet objects for up to 100 tweet IDs per request. The *lookup* resource is limited to 180 requests per 15-min window, which makes the download of large collections a tedious task. Because of this we did not attempt to download the full corpus (120 million tweets) and confined instead to the 150,000 labeled event tweets as well as roughly the same amount of unlabeled tweets (mostly not related to news events).

As *Events2012* was created in 2012, a large amount of the tweets in it is no longer available. The reason for this may either be that the tweets have since been deleted, or that the users originally posting them have now deleted their Twitter accounts. Due to this, the resulting hydrated *Events2012* dataset unfortunately contains only 82,887 of the original 152,950 labeled tweets. The per-category numbers of successfully retrieved news tweets can be seen in Table 5.2.

Table 5.2: Number of successfully retrieved tweets in *Events2012* by category.

Categories	**Retrieved Tweets**
Armed Conflicts & Attacks	8,176
Arts, Culture & Entertainment	6,529
Business & Economy	3,448
Disasters & Accidents	4,064
Law, Politics & Scandals	31,261
Miscellaneous	3,020
Science & Technology	1,958
Sports	24,414
Total	82,870

Merging Labels

As the scope of this thesis is limited to detecting news events without categorizing them into topics, we merge all the news tweets in *Events2012* into one class labelled *Event*. The exception is tweets labelled as *Sport*, which we label *Not event* together with the rest of the irrelevant tweets. The reason for this is two-fold. Firstly, we observed that sports is a very big topic in Twitter and that sports events generate an enormous amount of tweets. We are therefore afraid that the inclusion of sports events will affect the system's ability to detect more "important news" negatively. Secondly, we observed that the sport events very seldom were *breaking news*, i.e. unexpected events of some importance. A great share of the tweets were of the type "GOOOOAAAAL!" and "Matchday! Go <insert team here>!" Not only are these tweets very uninformative, but the little information they

[3]`https://dev.twitter.com/rest/reference/get/statuses/lookup`

contain are also easily found elsewhere, as there usually is no lack of live coverage of sports events. Because of this we mean that the inclusion of sports events does not add much value to our system. Still, sports events can easily be included in our system at a later time, simply by including them as news tweets in the training data. The final distribution of our dataset, after merging categories and adding irrelevant tweets, is shown in Table 5.3.

Table 5.3: Distribution of our final dataset after adding irrelevant tweets and merging them with sports tweets.

Type	Number of tweets
Event	58,456
Not-event (incl. sports)	114,433
Total	172,889

5.2 Evaluation

To perform a good evaluation of event detection in Twitter and other microblog services has shown to be so difficult that that coming up with good measures for quantitative and qualitative performance is itself a challenging research task. Weiler et al. (2015) address the lack of a common evaluation method in the literature, and they argue that a single event detection technique only can be evaluated "against itself", e.g. with respect to different parameter settings. This alongside the fact that many previous works publish neither their datasets nor the source code of their systems, makes comparison against previous systems extremely hard and some times even impossible. Some solve this by re-implementing previous solutions found in the literature, and applying them to their own datasets. However, the lack of necessary details of the implementations found in previous works are often pointed out as a reason that the performance may be dubious in the re-implementations. Because of the limitations of this thesis in terms of time and resources, we see it as more useful to put or effort in making our own system perform as good as possible, in preference to implementing earlier solutions just to compare it against. As far as evaluation metrics go, we keep it simple. What is important is that the system detects events, and that the detected events are in fact news events. This translates to *recall* and *precision*.

5.2.1 Automated Evaluation

We wish to perform an evaluation that is repeatable for the public, and as *Events2012* is the only publicly available large annotated corpus of news event from Twitter, we will utilize it to do so. However, with it being the only ground-truth dataset we have, we must also use it to train the classifier of our system. To perform the end-to-end evaluation of the system, we therefore split the dataset into distinct training- and test sets. The set of events for the training- and the test sets will always be disjoint, meaning that if tweets referring to event e occur in the training set, no tweets associated with event e will be found in the test set, and visa versa.

As for McMinn and Jose (2015), which to our knowledge is the only published work that performs an automated evaluation of a similar system against this same dataset, we will compare our own results against theirs. McMinn and Jose (2015) do however use an unsupervised approach, which let them use the full corpus for testing, and they have all the tweets of the original corpus at their disposal, so the comparison can unfortunately not be made on fully equal terms. For their evaluation they include only events with at least 75 tweets and they require that at least 5% or at least 15 tweets in a candidate event must be relevant to a single event from the ground truth for it to be considered detected (we presented their rationale for this choice in Section 2.1). For our own evaluation we feel that this threshold is a bit low, as it in many cases will be difficult for a user of the system to identify one specific news event if so few of the tweets in the candidate event are actually relevant. For our system we want it to be easy to spot a news event when presented with the corresponding tweets, and therefore demand that the candidate event must contain at least 5 tweets and that most of the tweets in a candidate event must be relevant to the ground truth news event for it to be considered detected. We define "most of" as at least 80%.

Another difference from McMinn and Jose (2015), is that we include all events with 10 tweets or more in the test data and we leave out events larger than 400 tweets. The rationale behind using 10 as a threshold, instead of McMinn and Jose (2015)'s 75, is that our test set is already limited opposed to theirs and we do not want to reduce it further. The reason for cutting off events larger than 400 tweets is that we see that these events in *Events2012* typically are news events in a wider definition than what we want for our system. As an example, we observe very large events that span several different debates and incidents in the 2012 US Presidential Election.

We create in total 20 different splits of *Events2012* where the distribution of training–test data is roughly 70-30 (it will vary because we pick events and belonging tweets on random until the test set is at least 30% of the full corpus). This corresponds to roughly 118,000 tweets for the training set and 54,000 tweets for the test set. For each split we train the ANN classifier until convergence using the training set, before the trained model is loaded into the system. We then use the corresponding test set to simulate a Twitter stream which delivers event tweets in the order they were posted originally, admixed with non-news tweets. Our system processes this stream until all the tweets in the test set have been classified and possibly assigned to an event. After each run, we analyze the detected events according to our detection criteria and calculate precision, recall and F1-score for the split. The final result is presented as the mean values of these metrics over all 20 runs.

For the clustering step we use 0.23 as a distance threshold for assigning a tweet to the same cluster as its nearest neighbor, as using this value obtained the best result in the experiments presented in Section 4.3. The window size is set to 2000 tweets.

5.2.2 User-based Evaluation

Since we have no guarantee about the pureness and completeness of *Events2012*, it is always a slight possibility that our system over-perform when using it for both training and testing. We therefore evaluate our system by letting human annotators evaluate detected events from today's Twitter stream as well. For this we use the full *Events2012* corpus to train the classifier of the system. This will let us see how our system, with a classifier trained on data from 2012, performs when released on today's Twitter stream.

Picking Candidate Events

To collect candidate events we run the system for a period of 4 days (May 17th to May 20th). Since our system is targeted towards English tweets and our training data is American, we limit the Twitter stream to English tweets from the geographical area that constitutes the US. We use no threshold on the confidence of tweets detected as news tweets by our classifier, and for the event detection stage (i.e. the clustering stage), only the last 5000 tweets are considered ($w = 5000$). The threshold value for assigning a tweet to the same event as its nearest neighbor is set at 0.20. The reason that we use a lower distance threshold than for the automated evaluation using *Events2012*, is that we expect a larger amount of tweets and more noise in the real-time Twitter stream. The lower threshold will give less noise in the final events, but it will also cause them to have a lower granularity. Hopefully the larger amount of data will provide sufficient similar tweets for the events to still be detectable.

Our system is expected to return a tremendous amount of news events, as all tweets that are classified as *news tweets* (correctly or not), but do not have a nearest neighbor closer than the threshold, will be considered new events. As opposed to when performing the automated evaluation against *Events2012*, we can not evaluate all detected events, as it will include having the annotators assess the correctness of probably tens of thousands of candidate events. This is were the analyzer component of the system comes in. We work out from the hypothesis that large events (i.e. events with a large number of tweets), as well as rapidly growing events are important news events. To pick candidate news events for our user-based evaluation, we query the database for the 100 largest events as well as the 100 events with the fastest increasing positive growth rates in the chosen time range. The former will let us pick up large events with a steady growth, while the latter will ensure that also smaller events can be detected as long as they are targets for a sudden rise in interest.

When picking the 100 largest threads we consider only those where the average classification confidence of all tweets are above 0.85, the average timestamp is within our chosen time window, and the size is larger than 5. To avoid events consisting of tweets produced by Twitter-bots, typically weather reports and traffic updates, we say that at least 85% of the tweets in an event must be posted by unique users. To reduce the amount of spam and events lacking in information (typically "I vote for <someone>" or "Happy birthday <celebrity>") we follow the suggestion in Petrovic et al. (2010) and remove events with entropy below a certain threshold. Entropy of an event is computed as

$$Entropy_{event} = -\sum_i \frac{n_i}{N} \log \frac{n_i}{N},$$

where n_i is the number of times term i appears in the event and N is the total number of terms in the event (Petrovic et al., 2010). We empirically found that setting this threshold at 5 worked pretty well, testing it on a different set of data. When picking the final candidate events for user-based evaluation, we use time windows of 24 hours, and select the 20 events with highest entropy from those resulting from following the above strategy.

Gathering Relevance Judgments

We end up with a total of 65 candidate events (the system was started in the evening on May 17th and only 5 candidate events that fulfilled all of the above requirements were detected that day). Although it would have been desirable to have the annotators judge an even larger number of events, the task would simply have become too extensive. Pilot evaluations showed that the assessment of one event typically took between 40 and 60 second, so we feel that the size of the task is already on the borderline. We could have let different annotators evaluate different sets of candidate events to make it possible to collect more relevance judgments. The reason that we do not do this, is that we want the judgments to be as consistent as possible across all candidate events, something that is better facilitated by letting all annotators evaluate the same candidates.

The evaluators are presented with a list of tweets and a corresponding word cloud for each candidate event, and asked to assess the relevancy of each one. The list of tweets for each event is limited to a maximum of 20 to prevent that our evaluators become fatigued from reading to many tweets. These tweets are the 20 firstly occurring tweets of the given event. Three response options are given when assessing the relevancy of a candidate event:

1. *Yes, the majority of the tweets are relevant to one specific news event*
2. *No, the majority of the tweets are not relevant to one specific news event, but they are related to a specific news topic*
3. *No, the majority of the tweets are relevant neither to a specific news event nor a specific news topic*

The reason for adding a third option, instead of just asking for a yes/no reply to the news event question, is that we want to know whether the candidate events not assessed to be news events actually have some news relevance or simply are irrelevant. Possibly, it will also give an idea about the fitness of the chosen granularity level.

A description of the task ahead of them is given to the respondents on beforehand, in which an (hopefully) comprehensible definition of news events is given (Appendix A). Despite us asking the evaluators to follow the same definition, we do expect differences in the answers, as the perception of news is, and always will be, to some degree personal. The actual survey is carried out by using a questionnaire created in Google Forms[4], and an example of the evaluation of one event can be seen in Appendix B. As the candidate events are from the US and the evaluation of their relevance therefore call for a certain insight in the American society as well as adequate language skills, we hand pick people that we believe meet these requirements to perform our evaluation – 9 in all. It may be mentioned that McMinn et al. (2013) use 5 annotators per candidate event when creating the *Events2012* corpus.

Ideally we would have measured both precision and recall for our system, but since the calculation of recall is practically impossible as it would have involved having the evaluators go through millions of tweets looking for news events, we confine ourselves to using only precision as a performance measure. In the literature, this has become the customary way of evaluating event detection in social streams when using human evaluators.

[4]`https://www.google.com/forms/about/`

Chapter 6

Results and Discussion

In this chapter we present and discuss the results from the experiments described in Chapter 5. Section 6.1 contains the explicit results of the automated and the user-based evaluations, while these results are analyzed and discussed in Section 6.2.

6.1 Results

6.1.1 Automated Evaluation

As shown in Table 6.1, our approach significantly outperforms McMinn and Jose (2015) on the *Events2012* dataset, with and F1-score of **81.8%**. Although the results are not entirely comparable due to the differences explained in Section 5.2, based on our observations, our method can safely be assumed to perform much better than the state-of-art entity-based approach of McMinn and Jose (2015). The average purity for the detected events is 96.8%, meaning that, on average, only 3.2% of the tweets in an event is erroneous.

Table 6.1: Results from McMinn and Jose (2015) (events with 75+ tweets, best run) and our own system (events with 10+ tweets, mean over 20 runs) evaluated using *Events2012*

	McMinn and Jose (2015)	**Our system**
Precision	0.302 (181/586)	**0.901** (271/300)*
Recall	0.310 (159/506)	**0.749** (112/150)*
F1	0.306	**0.818**

* *The numbers in the parentheses are the rounded mean counts over all 20 runs, and are not necessarily in perfect accordance with the corresponding precision and recall values as these are calculated for each run individually and then averaged.*

6.1.2 User-based Evaluation

As shown in Figure 6.1, 68.8% of the candidate events are evaluated to be *real news events* by the majority of the annotators, and we see that this number is only slightly reduced if we raise the annotator agreement threshold to 75%, i.e. 65.6%. At a 100% agreement level, 43.8% of the candidate events are evaluated to be real news events. By the majority (>50%), 18.8% of the candidate events are classified not to be news events, but to be related to a specific news topic, while only 9.4% is regarded neither news events nor news topics. This means that an impressive 90.6% of the events detected by our system have some news relevance. By leaving out the candidate events where the judgments are inconclusive (no class has majority) we are left with 62 events in total, whereof 44 are real news events, resulting in a **precision** of **71.0%**. The inter-annotator agreement among the human evaluators is 0.76, using Free-marginal multirater kappa (Randolph, 2005), which indicates strong agreement and is commonly regarded more than adequate.

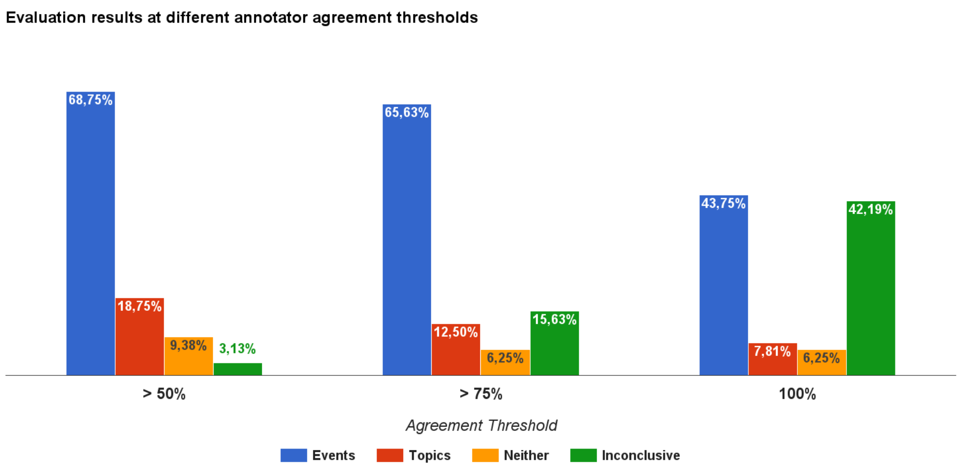


Figure 6.1: The figure shows the percentage of all candidate events being placed in each category by the share of annotators given on the x-axis. For instance, it shows that 68.25% of the candidates were regarded *news events*, 18.75% *news topics*, and 9.38% *irrelevant*, by the majority (>50%) of the annotators. The 'inconclusive' bars show the amount of candidate events for which a category could not be assessed on the given agreement threshold. We see for instance that neither of the categories was chosen by the majority for 3.13% of the candidate events.

6.2 Discussion

6.2.1 Automated Evaluation

The high F1-score shows that our proposed method has an overall very good ability to detect news events in a stream of tweets. The extremely high average purity as well as the precision score indicate that our $mbTC$ algorithm produces very good events (clusters), although the slightly lower recall rate may indicate that the 0.23 distance threshold for the clustering algorithm is a bit strict or that a window size of 2000 is too low.

Comparison Comparing solutions for event detection in Twitter is, as mentioned several times already, often problematic. The perhaps biggest issue with our own comparison against McMinn and Jose (2015), is that we use substantially fewer irrelevant tweets in our emulated stream. We expect that the use of more irrelevant tweets and spam tweets would have caused a somewhat lower precision. This would not necessarily have given a more correct picture, though, as *Events2012* is guaranteed to contain many real news events that were not detected and given relevance judgments when the corpus was created. As a consequence, actual news events would likely have been detected, but counted as "misses" because they were not present in the ground truth. McMinn and Jose (2015) verifies this by obtaining a significantly better precision (0.636) when letting human evaluators judge the relevance of candidate events detected in *Events2012* as opposed to when performing the automated evaluation using the original relevance judgments of the corpus (0.302). We also believe that our system would have been able to filter most of any additional spam tweets in the stream, as our news tweet classifier shows good ability to separate irrelevant tweets from news tweets with a mean precision of 0.907.

Our recall measurements, on the other hand, is less likely to have suffered any loss if we had increased the amount of spam. The reason for this is that the same events would still have been present in the stream, and our relatively strict threshold for assigning two tweets to the same event would have prevented these from being polluted by the spam tweets, so that they still would have been detected. The high average purity of the detected events shows that we could have endured significantly more noise in our events while still being able to successfully detect them at the 80% threshold.

6.2.2 User-based Evaluation

The results from the user-based evaluation show that our system is capable of detecting real *news events* in the real-time Twitter stream at a **71.0% precision rate**, which is very good for this specific task. By way of comparison, do McMinn and Jose (2015), who claim to outperform current state-of-the art approaches, obtain a precision score of 63.6% when using crowdsourcing to assess detected events in *Events2012*. They perform the evaluation on 1210 detected events from the dataset spanning 28 days, which corresponds to around 14 events per day. This is close to our own user-based evaluation where we pick a maximum of 20 candidate events per day. Because of this, we mean that these evaluations are comparable.

Annotators and Evaluation Although our inter-annotator agreements is good, we see by analyzing the the answers of each respondent individually, that one of them significantly stands out from the rest. He or she did, for example, consider five of the candidate events to be news topics while all the other annotators evaluated them to be news events. The number of candidate events that this respondent regarded as news events is almost two standard deviations lower than the mean for all respondents, and the inter-annotator agreement increases significantly if we leave this respondent out, resulting in a kappa of 0.79 (which is close to outstanding). The reason for this could be that this respondent had another interpretation of the definitions compared to the rest, or it could be that he or she was a lazy-respondent that simply chose the most general alternative whenever the "correct" answer was not immediately clear. Such respondents generally become very visible when using a relatively small number of evaluators, and they may affect the outcome substantially.

As long as we choose to assess the class of a candidate event based on what the majority of the annotators mean, our results are, however, not very affected by this one respondent. To keep our findings as unbiased as possible, all answers were included in the results presented. We would however like to point out that if we ignore the respondent discussed above, 53.2% of the candidate events would be judged to be news events by *all* annotators.

Cases of Disagreement The typical disagreements between annotators are whether a candidate event is related to a specific news event or a topic, and whether a candidate event is related to a news topic or not related to any news at all.

An example of a candidate event where disagreement arose between news event and news topic, was one related to "THE UPCOMING CLOSING OF THE KENTUCKY PRIMARY ELECTION POLLS". The same number of annotators chose both categories, while one meant that this was an irrelevant event. It is hard though, to say which is right, but those who did not consider this a news event probably found the news value to be too low to fit the category. It is, after all, rare to see it announced in mainstream media that the polls in a state are closing in an hour. However, such events are examples of information with a narrow spatial and temporal relevance that can be found in Twitter, but perhaps not elsewhere, and thus it illustrates a valuable property of social medias.

Other examples with large disagreement are three events related to "THE RELEASE OF THE NEW CAPTAIN AMERICA - CIVIL WAR MOVIE", one related to "DANIEL CRAIG TURNING DOWN A MULTI-MILLION-DOLLAR CONTRACT FOR NEW JAMES BOND MOVIES", and one related to "THE SPREAD OF THE ZIKA VIRUS IN THE US". For the CAPTAIN AMERICA events we suspect that the respondents started to see them more as a news topic than individual news events, as these events continued to appear several days after the release of the movie. This interpretation is not surprising, and we are inclined to agree that CAPTAIN AMERICA could be viewed as a news topic.

The last two, on the other hand, should by our definition stand out as obvious news events, but we believe that the annotators may have been confused by different angles and facts represented in the belonging tweets. In the DANIEL CRAIG event, some discuss that he turned down a lot of money, some praise the job he has done in the role, while others discuss possible successors. It is probably because of all these sub-events that many of the annotators consider it a news topic, although the actual breaking news is that *"Daniel*

Craig will not play Bond again." With the ZIKA VIRUS event, we observe that the tweets for instance reports different numbers of pregnant women that have tested positive mixed with the number for patients being monitored. It is likely that some of the annotators regard the different numbers as individual news events, and the ZIKA VIRUS itself as a news topic. We should probably have instructed the respondents to tolerate fluctuating facts in the candidate events, as this is very typical for Twitter – especially in the early tweets regarding an event (number of deaths, money figures, locations, etc.)

Events Assessed to be Not-news Of the candidate events judged to be related neither to news events nor news topics by the majority, two were "LOCATION UPDATES FROM NEW YORK", two were related to different "SCHOOL ELECTIONS" and were typically calls for votes, one was "CONGRATULATIONS" of various types, and the last one was related to "PEOPLE SEEING THE NEW CAPTAIN AMERICA MOVIE IN THEATER." The candidate event containing location updates from NEW YORK is by no doubt irrelevant, and we believe that the reason that these passes through or news tweet filtering step is that one or more events in the training data causes the phrase 'New York' to be strongly associated with news. For the one regarding SCHOOL ELECTIONS and the one regarding CAPTAIN AMERICA, there was some disagreement on whether it was news topics or not. This is understandable, as the former is related to a matter that may have some news value, although perhaps not enough to be mentioned in mainstream media. It is also debatable whether one should regard SCHOOL ELECTIONS at different schools as one topic or not. For the latter, we suspect that not all respondents were aware that a new CAPTAIN AMERICA movie was just released, and therefore failed to identify it as a news topic. Another possibility is that those who did not regard it a news topic simply did not recognize any news value in the tweets, despite them actually being related to a news event.

All the edge cases discussed above reflect the difficulties of defining and interpreting news events and news topics. They may also indicate weaknesses in our definitions or in our evaluation criteria, although the strong inter-annotator agreement suggests that this did not manifest itself as a big issue.

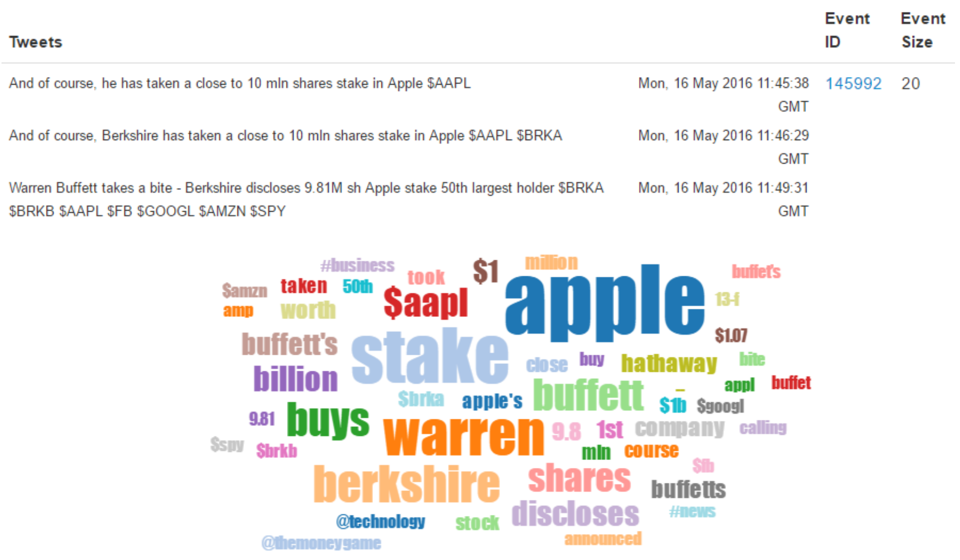
Events Assessed to be News After going through all candidate events that the majority judged to be news events and assigning them to one of the news categories suggested in McMinn et al. (2013), we end up with the distribution shown in Table 6.2. This shows that our system is able to capture news events belonging to a wide range of news topics. We do however see that no event fitting in the *Business & Economy* category was detected. This could simply be because no big event (in the meaning that it attracts interest in Twitter) occurred for the time in which we gathered our candidate events. We have however seen our system detect such events, for instance the one shown in Figure 6.2 which was detected the day before we started our collection of candidate events for user-based evaluation, so we know that it is capable. That the *Law, Politics & Scandals* category is so big is not a surprise to us, after having witnessed throughout the work with this thesis how much attention American politics in general, and the Primary Election and its runners in particular, gets in Twitter. That the system is able to capture several events both in the *Disasters & Accidents* and the *Science & Technology* category is also very promising, given the relatively small amount these categories constitutes of the training data (Table 5.2 in Section 5.1).

Further does a qualitative inspection of the detected events shows that the events are very pure in general, while at the same time they cover good variety in tweets. It also shows the analyzer is able to detect large events (∼200 tweets) as well as smaller events (∼10 tweets). We would also say that most of the detected events, in our opinion, are important news events (think *breaking news*).

All in all, both the parameters of the clustering stage and our method for choosing the candidate events (i.e. the analyzer) seems fit for the task of presenting "daily news," as verified by the evaluation results.

Table 6.2: Assessed news events by the majority of annotators distributed over the news categories suggested in McMinn et al. (2013).

News Category	Assessed News Events
Armed Conflicts & Attacks	7
Arts, Culture & Entertainment	8
Business & Economy	0
Disasters & Accidents	7
Law, Politics & Scandals	16
Miscellaneous	3
Science & Technology	3
Total	44



Tweets		Event ID	Event Size
And of course, he has taken a close to 10 mln shares stake in Apple $AAPL	Mon, 16 May 2016 11:45:38 GMT	145992	20
And of course, Berkshire has taken a close to 10 mln shares stake in Apple $AAPL $BRKA	Mon, 16 May 2016 11:46:29 GMT		
Warren Buffett takes a bite - Berkshire discloses 9.81M sh Apple stake 50th largest holder $BRKA $BRKB $AAPL $FB $GOOGL $AMZN $SPY	Mon, 16 May 2016 11:49:31 GMT		

Figure 6.2: Example of *Business & Economy* news event detected by our system: *Investor Warren Buffett's Berkshire Hathaway buys 9.8M Apple shares.*

Duplicate Events We observe some duplicates in the detected events, and that duplicates typically arise for events that are very frequently tweeted about, such as "EGYPTAIR FLIGHT 804 CRASHES – 66 PEOPLE KILLED" and "SHOOTING REPORTED OUTSIDE THE WHITE HOUSE". It is, however, often a very fine line between what should be regarded duplicates and what should be regarded separate events. Our system did for instance detect "IRISH SINGER SINEAD O'CONNOR REPORTED MISSING" and "MISSING IRISH SINGER SINEAD O'CONNOR IS NOW FOUND" as two separate events. In this case, the elapsed time between the singer being reported missing and the singer being reported found, will typically decide whether it is perceived as two separate news events or not. If we had used a lower threshold in the clustering process, the result would have been a lower granularity for the detected events, and these two events may have been recognized as one event. A lower granularity would probably have removed some duplicates, but on the other hand, it may have made the evolvement of single news events appear less evidently. A higher threshold also would have caused more noise in the detected events. Instead of raising the threshold to counter detection of duplicate events and thus introduce more noise in the result, we believe that a good way to eliminate duplicates would be to introduce a cluster merging step either at the time the events are fetched from the database or as a periodic process.

Weak Points

We recognize that the comprehensiveness of our user-based evaluation is limited. This is because manual assessment of events from social streams is a tedious and labor-intensive task, and the circumstances around this thesis pose certain limitations on time and resources. Still, after observing the system for many days during this write-up, we truly believe that its behavior in the four days the candidate events were collected, reflects its long-run behavior. However, with a larger time frame and more resources than we have had available for this work, we would have performed a more extensive evaluation of the system's performance on the real-time Twitter stream, using more candidate events, spanning a longer time range. It would have been natural to use a crowdsourcing tool like, for example, *Amazon Mechanical Turk* for this work, following the approach taken in McMinn et al. (2013) and McMinn and Jose (2015). Although this would have made it possible to evaluate a larger amount of candidate events, crowdsourcing does, in general, not guarantee better evaluations in terms of quality.

Optimally, we also would have evaluated different parameter settings for our system, as well as different analyzers and ranking algorithm to pick the events of interest, to find the best possible setup. Our user-based evaluation was kind of a one-shot process and the setup used was based on empirical observations. It is unlikely that we have discovered the optimal parameters by taking this approach, and very likely, the system is capable of performing even better if the optimal parameters are found and used.

6.2.3 Significance of Results

The results of the automated evaluation using *Events2012* shows that our system detects news events with a performance that beats an alleged state-of-the-art solution by a great margin. Further does our user-based evaluation show that our system also performs good on today's real-time Twitter stream, and that the precision obtained in the automated evaluation was not greatly overestimated – especially when considering that the training data was collected in 2012.

The Power of Word Embeddings We believe that a part of the reason why the system performs so good on today's Twitter stream, despite the *news tweet* classifier being trained on old data, is owing to the power of the *Word2vec* model. The specific model we use was trained on the Twitter stream for 300 days by Godin et al. (2015) in 2013/2014, and therefore carries more extensive and newer information than the *Events2012* dataset. We believe this model helps generalizing the events of *Events2012* meaning that instead of learning the specific entities and actions found in the training data, the classifier will learn a generalization of them. We illustrate this by the following example:

Let us consider an organization that did not exist or was not mentioned in media when the training data was collected. Another, similar organization was, on the other hand, involved in news events at that time. If both organization then occurs in the Twitter stream when training the *Word2vec* model (unsupervised and super-efficient, remember?), the model will learn similar representations for them both. Therefore, the classifier, trained using these representations, will be able to recognize an event involving the new organization as significant despite its absence in the training data. It is also important to remember that this capacity to generalize extends to all kinds of terms like `extremist` $\approx$ `terrorist`, spellings like `misspelling` $\approx$ `mispelling`, abbreviations like `birthday` $\approx$ `b-day`, and capitalization like `America` $\approx$ `america`. Put somewhat extremely, we can say that word embeddings capture *"what words mean, not what they say"*.

Further, we believe that the reason why our rather simple *AvgW2V* representation works so well is due to the shortness of tweets. Its main weakness, that it does not take word order in to account, is probably not an big issue for tweets given that a tweet's average length is 10.7 words. After all, there are only so many ways of re-ordering 10 words to completely change the meaning of a sentence while still having it making sense. The expressiveness of *AvgWV* for tweets also reveals itself when we use it for clustering with very good results.

Chapter 7

Conclusion and Future Work

In Section 7.1 we conclude the work presented in this thesis, give an account for its contributions and summarize its goal achievements with regard to the research questions, while in Section 7.2, we make a proposal for future work.

7.1 Conclusion

Driven by the motivation to help disseminating important information in today's information society even quicker than today, the main focus of this thesis has been on detecting and presenting *news events* in *real-time* from the Twitter stream. A text representation based on *word embeddings* has been used as a base for both filtering *news tweets* and for forming *news events*. The task of filtering news tweets from the Twitter stream has been solved by using an artificial neural network classifier, trained on labeled news tweets. Next, we have proposed two novel algorithms for clustering tweets to form news events, both satisfying the constraints in the streaming setting of Twitter. We have also proposed an analyzer that is able to extract important, detected news events from a given time frame taking basis in events' sizes and their change in tweet frequency. Lastly, we have developed a web application that displays, visualizes, and facilitates the browsing of, detected news events.

The main conclusions that can be drawn from the work that we have presented in this thesis are as follows:

- Artificial neural networks are well fit for classification tasks in the streaming setting of Twitter, given the existence of training data.
- Word embeddings are powerful means to tackle the noisy language in Twitter messages, while they also enable generalization of knowledge.
- The simple approach of creating text representations by averaging word vectors is very apt for streaming short texts, like tweets, as it produces fixed-length real-valued dense vectors in linear time. The resulting vectors are remarkably information-rich, and captures sufficient information to separate news tweets from non-news tweets. They are also very fit for clustering of semantically similar short texts.

- News events can be detected in real-time from the Twitter stream, more accurately than current state-of-the-art solutions, by using an ANN classifier to detect news relevant tweets, and our online clustering algorithm to create news events, both taking tweets represented as average word embeddings as input.
- The size as well as an rapidly increasing frequency of related tweets for a news event in Twitter, are good indicators that a news event is of interest to the public. A low entropy, on the other hand, indicates that it is of less interest.

7.1.1 Contributions of the Thesis

The main contributions of this thesis are specifically related to online detection of news events from the Twitter stream, leaning on machine learning, modern NLP methods and unsupervised clustering. The specific contributions of this work can be summarized as follows.

Representing Short Texts of Tweets

Our experiments, conducted using McMinn et al. (2013)'s *Events2012* dataset, has revealed that our proposed *AvgW2V* model better captures information necessary to separate irrelevant tweets from news tweets, than *Paragraph Vectors* and *AvgGloVe*. However, once again, we feel obliged to point out that we can not in general say that one model is better than the next one, as the premises for each individual model is decided by its underlying pre-trained model. Nevertheless, we have showed that the averaging of word embeddings captures sufficient information for our application and argued that this is owed to the powerfulness of word embeddings and its resistivity against noise, as well as the shortness of tweets.

Although the idea of representing text by averaging word embeddings is not new per se, we have not found it applied to the event detection task anywhere in the literature. Thus, our approach can be considered to advance the state-of-art within detecting news events from short streaming texts.

Novel Approach for Detecting News Tweets by Classification

Preliminary work had shown that an artificial neural network (ANN) trained on labeled news tweets, taking tweets represented as averaged *Word2vec* vectors (*AvgW2V*) as input, was well suited to detect tweets with news relevancy in the Twitter stream. In this work we have tested alternative representations of tweet text as well as different additional Twitter-specific features. The results of extensive experiments have shown the effectiveness of the *AvgW2V* representation for the classification task compared to the other investigated representations, and they have shown that the addition of Twitter-specific features has a detrimental or statistically insignificant effect on the classification performance. We have also discovered that weighting average word embeddings using IDF-weights yields lower performance compared to using plain average word embeddings. Further we have discovered that the method performs best if casing and hashtags are included when generating the representations, arguing that this is because the ANN classifier is able to utilize these as internal features.

Novel Algorithms for Detecting News Events by Clustering

As many before us, we have based our event detection approach on clustering of tweets. We have proposed two novel algorithms for online clustering of the Twitter stream. Both algorithms have been based on *AvgW2V* representations of tweet texts and (cosine) distance to nearest neighbors. Experiments have shown that both algorithm perform well in terms of *homogenity* and *completeness* when applied to the task of clustering news tweets, and that their time complexity is well within the boundaries required to be applicable to the Twitter stream. We have chosen the most effective algorithm (*mbTC*) and applied it to the task of event detection. Extensive experiments, performed both on static collections of tweets and on the real-time Twitter stream, have shown the algorithm's ability to effectively produce correct news events.

Presenting News Events of Interest

As our suggested approach produces a very large number of events from the Twitter stream, we have proposed a novel approach to find the detected events that should be presented, based on event size, tweet frequency and unique tweeters. Event entropy has been employed to eliminate uninformative events. The effectiveness of our analyzer has been demonstrated by having users evaluate the news events produced over a $\sim$4-day run on the real-time Twitter stream.

In addition, we have developed a web application for displaying, visualizing and browsing detected news events, as a proof-of-concept for how our approach for detecting news events could be utilized to create a newswire service for the general public.

7.1.2 Goal Achievements

The following elaborates on how the research question presented in Section 1.3 have been answered:

RQ 1: *How to choose features from tweets and how should we represent them when using an artificial neural network to filter news tweets from non-news tweets?*

The experiments presented in Section 4.2 have shown that the textual content of tweets represented using the *AvgW2V* model constitutes features that very effectively capture sufficient information for an ANN to be able to separate news tweets from non-news tweets with a high precision. Further, it is shown that the addition of Twitter-specific features as input for the classifier has a detrimental or a statistically insignificant effect on its performance.

RQ 2: *How can we cluster semantically and temporally similar news tweets to form events in real-time?*

Two novel clustering algorithms have been presented in Section 4.3 and it has been shown that they are applicable to the streaming setting of Twitter, and that they produce good clusters in terms of homogeneity and completeness. One of the algorithms have further proven its ability to cluster news tweets into news events as shown by the results presented in Section 6.1.

RQ 3: *How can we identify the important news events for presentation?*
A strategy for identifying news events worth presenting, which is based on event sizes as well as rapidly increasing growth in tweet frequencies, has been presented, and it has been shown in Subsection 6.1.2 that this strategy produces good results when applied to news events detected directly from the real-time Twitter stream.

7.2 Future Work

To improve, or to extend, our proposed system, we make the following suggestions for possible future work.

Make it adapt to changes in the stream Ideally, the system should continue to learn as it runs. This would require a continuous update of both the *Word2vec* model, underlying the *AvgW2V* method, and the ANN model, in an online fashion using tweets from the stream. For the *Word2vec* model this should be fairly easy to accomplish, as its learning algorithm is completely unsupervised. The ANN classifier, on the other hand, is in its current supervised form dependent on tweets being verified as actual news tweets. One approach to this problem could be to simultaneously track the Twitter accounts of known news broadcasters and use tweets posted by these, combined with similar tweets retrieved using traditional search and query expansion techniques, as training data for the classifier.

Refine the selecting of interesting events It would be interesting to investigate the possibility to include a second classifier in the system, taking cluster-level (i.e. event-level) features, to select the the most interesting news events for presentation. It is also likely that many spam and erroneous returned news events could be eliminated by including an option for the users to mark reported events as non-news and then used this feedback to train the classifier – i.e. including *reinforcement learning* based on user feedback.

Make the system ”unsupervised” This is a very comprehensive improvement, and probably worthy of its own thesis, but ideally, the system should be trained in an unsupervised fashion, overcoming the demand for labeled training data. A quick suggestion could be to collect tweets from the Twitter accounts of known news broadcasters and use search to retrieve tweets regarding the same events from regular Twitter users. An idea would be to use query expansion to widen the result. The retrieved tweets could then be used to train the classifier. Although the learning still would be supervised (thus the quotation marks around unsupervised), it would eliminate the need for human annotators.

Predicting breaking news This is, from every indication, an extremely difficult task, but it would be very interesting to use our proposed method as a basis for a system able to *predict* breaking news. What we mean by this, is that one could use our system to detect news events, but extend it to also predict whether an detected event is likely to occur in the newswire in the immediate future. The main challenge here, we think, is that events are typically very sparsely reported in Twitter before reaching the newswire. If feasible, however, such a system would likely be of uttermost interest for everyone from governments and corporations to investors, journalists, and the general public.

Bibliography

Agarwal, P., Vaithiyanathan, R., Sharma, S., Shroff, G., 2012. Catching the Long-Tail: Extracting Local News Events from Twitter. Sixth International AAAI Conference on Weblogs and Social Media, 379–382.

Aggarwal, C. C., Subbian, K., 2012. Event Detection in Social Streams. In: Proceedings of the SIAM International Conference on Data Mining. pp. 624–635.

Allan, J., 2002. Topic Detection and Tracking: Event-based Information Organization. Vol. 12. Kluwer Academic Publishers, Norwell, MA, USA.

Baroni, M., Dinu, G., Kruszewski, G., 2014. Don't count , predict ! A systematic comparison of context-counting vs . context-predicting semantic vectors. Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics., 238–247.

Chen, F., Neill, D., 2015. Human rights event detection from heterogeneous social media graph. Big Data 3 (1).

Dietterich, T. G., oct 1998. Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms. Neural Computation 10 (7), 1895–1923.

Efron, M., jun 2011. Information search and retrieval in microblogs. Journal of the American Society for Information Science and Technology 62 (6), 996–1008.

Fukushima, K., 1980. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. Biological Cybernetics 36 (4), 193–202.

Gabrilovich, E., Markovitch, S., 2006. Overcoming the Brittleness Bottleneck using Wikipedia: Enhancing Text Categorization with Encyclopedic Knowledge, 1301–1306.

Gimpel, K., Schneider, N., O'Connor, B., Das, D., Mills, D., Eisenstein, J., Heilman, M., Yogatama, D., Flanigan, J., Smith, N. A., 2011. Part-of-Speech Tagging for Twitter: Annotation, Features, and Experiments. Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Shortpapers (2), 42–47.

Godin, F., Vandersmissen, B., De Neve, W., de Walle, R., 2015. Multimedia Lab @ ACL W-NUT NER Shared Task: Named Entity Recognition for Twitter Microposts using Distributed Word Representations. In: Workshop on Noisy User-generated Text, ACL 2015.

Han, B., Baldwin, T., 2011. Lexical Normalisation of Short Text Messages : Makn Sens a # twitter. Computational Linguistics V (212), 368–378.

Hinton, G. E., Osindero, S., Teh, Y. W., 2006. A fast learning algorithm for deep belief nets. Neural computation 18 (7), 1527–54.

Hopfield, J. J., 1982. Neural networks and physical systems with emergent collective computational abilities. Proceedings of the National Academy of Sciences of the United States of America 79 (8), 2554–2558.

Hua, T., Chen, F., Zhao, L., Lu, C., Ramakrishnan, N., 2013. STED: Semi-Supervised Targeted Event Detection. People.Cs.Vt.Edu.

Jordan, M., 2013. Poke me, i'm a journalist: The impact of facebook and twitter on newsroom routines and cultures at two south african weeklies. Ecquid Novi: African journalism Studies 34 (1), 21–35.

Kobus, C., Yvon, F., Damnati, G., 2008. Normalizing SMS: are two metaphors better than one? Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1 (August), 441–448.

Kumaran, G., Allan, J., 2004. Text classification and named entities for new event detection. Proceedings of the 27th annual international ACM . . . , 297–304.

Kunneman, F., van den Bosch, A., 2014. Event detection in Twitter: A machine-learning approach based on term pivoting. Proceedings of the 26th Benelux Conference on Artificial Intelligence, 65–72.

Le, Q., Mikolov, T., 2014. Distributed Representations of Sentences and Documents. International Conference on Machine Learning - ICML 2014 32, 1188–1196.

LeCun, Y., Bottou, L., Bengio, Y., Haffner, P., 1998. Gradient-based learning applied to document recognition. Proceedings of the IEEE 86 (11), 2278–2323.

Li, R., Lei, K. H., Khadiwala, R., Chang, K. C. C., 2012. TEDAS: A twitter-based event detection and analysis system. Proceedings - International Conference on Data Engineering, 1273–1276.

Little, W., feb 1974. The existence of persistent states in the brain. Mathematical Biosciences 19 (1-2), 101–120.

Littlestone, N., 1988. Learning quickly when irrelevant attributed abound: a new linear-threshold algorithm. Machine Learning 2 (4), 285–318.

Makice, K., 2009. Twitter API: Up and running: Learn how to build applications with the Twitter API. O'Reilly Media, Inc.

McCord, M., Chuah, M., 2011. Spam detection on twitter using traditional classifiers. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). Vol. 6906 LNCS. pp. 175–186.

McMinn, A. J., Jose, J. M., 2015. Real-Time Entity-Based Event Detection for Twitter 9283, 65–77.

McMinn, A. J., Moshfeghi, Y., Jose, J. M., 2013. Building a large-scale corpus for evaluating event detection on twitter. In: Proceedings of the 22nd ACM International conference on conference on Information; Knowledge Management. ACM, New York, NY, USA, pp. 409–418.

Mikolov, T., Corrado, G., Chen, K., Dean, J., 2013. Efficient Estimation of Word Representations in Vector Space. Proceedings of the International Conference on Learning Representations (ICLR 2013), 1–12.

Pennington, J., Socher, R., Manning, C. D., 2014. GloVe: Global Vectors for Word Representation. Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, 1532–1543.

Petrovic, S., Osborne, M., Lavrenko, V., 2010. Streaming First Story Detection with application to Twitter. Computational Linguistics (June), 181–189.

Petrovic, S., Osborne, M., McCreadie, R., Macdonald, C., Ounis, I., Shrimpton, L., 2013. Can twitter replace newswire for breaking news?

Qin, Y., Zhang, Y., Zhang, M., Zheng, D., 2013. Feature-Rich Segment-Based News Event Detection on Twitter. Proceedings of the Sixth International Joint Conference on Natural Language Processing (October), 302–310.

Randolph, J. J., 2005. Free-Marginal Multirater Kappa (multirater K [free]): An Alternative to Fleiss. Online Submission (October), 20.

Řehůřek, R., Sojka, P., May 2010. Software Framework for Topic Modelling with Large Corpora. In: Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks. ELRA, Valletta, Malta, pp. 45–50.

Repp, Ø. K., dec 2015. TDT4501 Specialization Project: Detecting News-Related Tweets in the Real-Time Twitter Stream.

Ritter, A., Etzioni, O., Clark, S., 2012. Open domain event extraction from twitter. Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '12, 1104.

Rong, X., 2014. word2vec Parameter Learning Explained. arXiv preprint arXiv:1411.2738, 1–19.

Rosenberg, A., Hirschberg, J., 2007. V-measure: A conditional entropy-based external cluster evaluation measure. Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning 1 (June), 410–420.

Rosenblatt, F., 1957. The perceptron, a perceiving and recognizing automaton Project Para. Cornell Aeronautical Laboratory.

Rosenstiel, T., Sonderman, J., Loker, K., Ivancin, M., Kjarval, N., September 2015. Twitter and the News : How people use the social network to learn about the world. Online at `www.americanpressinstitute.org`.

Rumelhart, D. E., Hinton, G. E., Williams, R. J., 1986. Learning representations by back-propagating errors. Nature 323 (6088), 533–536.

Sakaki, T., Okazaki, M., Matsuo, Y., 2010. Earthquake shakes Twitter users: real-time event detection by social sensors. WWW '10: Proceedings of the 19th international conference on World wide web, 851.

Schifferes, S., Newman, N., Thurman, N., Corney, D., Goker, A., Martin, C., 2014. Identifying and verifying news through social media: Developing a user-centred tool for professional journalists. Digital journalism.

Srivastava, N., Hinton, G. E., Krizhevsky, A., Sutskever, I., Salakhutdinov, R., 2014. Dropout : A Simple Way to Prevent Neural Networks from Overfitting. Journal of Machine Learning Research (JMLR) 15, 1929–1958.

Tang, G., Xia, Y., Wang, W., Lau, R., Zheng, T. F., 2012. Clustering tweets using Wikipedia concepts. In: Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14). pp. 2262–2267.

Tripathy, R. M., Sharma, S., Joshi, S., Mehta, S., Bagchi, A., 2014. Theme Based Clustering of Tweets. In: Proceedings of the 1st IKDD Conference on Data Sciences - CoDS '14. ACM Press, New York, New York, USA, pp. 1–5.

Tsur, O., Littman, A., Rappoport, A., 2013. Efficient Clustering of Short Messages into General Domains. International Conference on Weblogs and Social Media (ICWSM), –.

Weiler, A., Grossniklaus, M., Scholl, M. H., 2015. Evaluation Measures for Event Detection Techniques on Twitter Data Streams. In: Data Science - Proceedings of 30th British International Conference on Databases, BICOD 2015. pp. 108–119.

Werbos, P. J., 1974. Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences. Ph.D. thesis, Harvard University.

Yang, Y., Zhang, J., Carbonell, J., Jin, C., 2002. Topic-conditioned novelty detection. Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '02, 688–693.

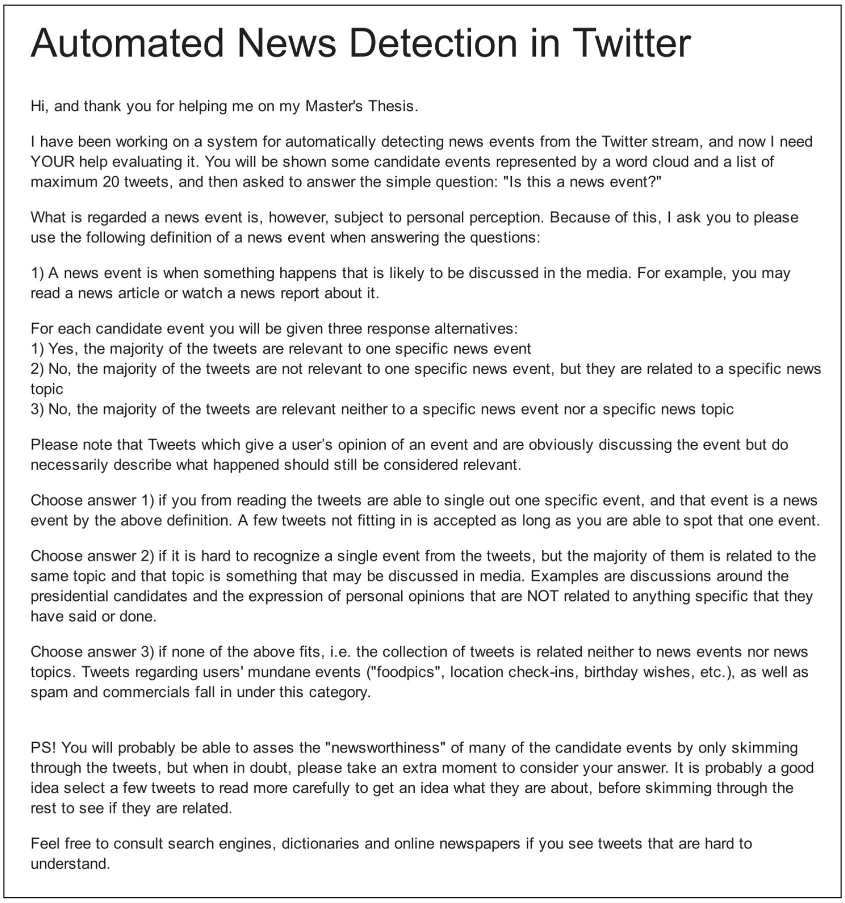
Zhou, D., Chen, L., He, Y., 2015. An Unsupervised Framework of Exploring Events on Twitter: Filtering, Extraction and Categorization. Twenty-Ninth AAAI Conference on Artificial Intelligence, 2468–2474.

Zhou, D., Chen, L., He, Y., Integration, I., Science, A., 2014. A Simple Bayesian Modelling Approach to Event Extraction from Twitter. ACL, 700–705.

Appendices

Appendix A

Instructions for Evaluators



Automated News Detection in Twitter

Hi, and thank you for helping me on my Master's Thesis.

I have been working on a system for automatically detecting news events from the Twitter stream, and now I need YOUR help evaluating it. You will be shown some candidate events represented by a word cloud and a list of maximum 20 tweets, and then asked to answer the simple question: "Is this a news event?"

What is regarded a news event is, however, subject to personal perception. Because of this, I ask you to please use the following definition of a news event when answering the questions:

1) A news event is when something happens that is likely to be discussed in the media. For example, you may read a news article or watch a news report about it.

For each candidate event you will be given three response alternatives:
1) Yes, the majority of the tweets are relevant to one specific news event
2) No, the majority of the tweets are not relevant to one specific news event, but they are related to a specific news topic
3) No, the majority of the tweets are relevant neither to a specific news event nor a specific news topic

Please note that Tweets which give a user's opinion of an event and are obviously discussing the event but do necessarily describe what happened should still be considered relevant.

Choose answer 1) if you from reading the tweets are able to single out one specific event, and that event is a news event by the above definition. A few tweets not fitting in is accepted as long as you are able to spot that one event.

Choose answer 2) if it is hard to recognize a single event from the tweets, but the majority of them is related to the same topic and that topic is something that may be discussed in media. Examples are discussions around the presidential candidates and the expression of personal opinions that are NOT related to anything specific that they have said or done.

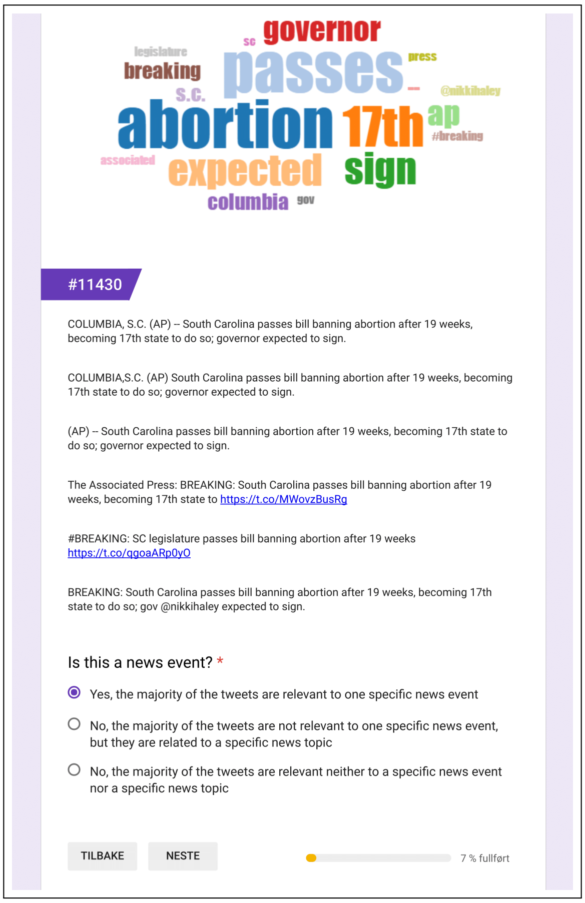
Choose answer 3) if none of the above fits, i.e. the collection of tweets is related neither to news events nor news topics. Tweets regarding users' mundane events ("foodpics", location check-ins, birthday wishes, etc.), as well as spam and commercials fall in under this category.

PS! You will probably be able to asses the "newsworthiness" of many of the candidate events by only skimming through the tweets, but when in doubt, please take an extra moment to consider your answer. It is probably a good idea select a few tweets to read more carefully to get an idea what they are about, before skimming through the rest to see if they are related.

Feel free to consult search engines, dictionaries and online newspapers if you see tweets that are hard to understand.

Appendix B

Example of Candidate Event



#11430

COLUMBIA, S.C. (AP) -- South Carolina passes bill banning abortion after 19 weeks, becoming 17th state to do so; governor expected to sign.

COLUMBIA,S.C. (AP) South Carolina passes bill banning abortion after 19 weeks, becoming 17th state to do so; governor expected to sign.

(AP) -- South Carolina passes bill banning abortion after 19 weeks, becoming 17th state to do so; governor expected to sign.

The Associated Press: BREAKING: South Carolina passes bill banning abortion after 19 weeks, becoming 17th state to https://t.co/MWovzBusRg

#BREAKING: SC legislature passes bill banning abortion after 19 weeks https://t.co/qgoaARp0yO

BREAKING: South Carolina passes bill banning abortion after 19 weeks, becoming 17th state to do so; gov @nikkihaley expected to sign.

Is this a news event? *

[✓ Yes, the majority of the tweets are relevant to one specific news event]

No, the majority of the tweets are not relevant to one specific news event, but they are related to a specific news topic

No, the majority of the tweets are relevant neither to a specific news event nor a specific news topic