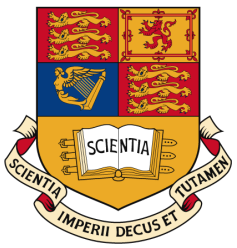
Understanding Revolution 2.0 - Mining the Arab Spring Social Network Data



George Eracleous

Department of Computing

Imperial College London

A thesis submitted for the degree of

MEng Information Systems Engineering

2012

I would like to dedicate this thesis to my loving parents.

Acknowledgements

And I would like to acknowledge ...

Abstract

TODO: WRITE ABSTRACT

Contents

Chapter 1

Introduction

In this chapter, we will explain the problem we aim to tackle in this project and what are its challenges. By breaking the problem down into smaller problems we will explain the steps we need to undertake in order to find the solution and the diffculties we expect to encounter in our endeavour.

1.1 Motivation

Social media have literally changed our lives in the recent years. Social media platforms such as Facebook, Twitter and many more not only offer their users a way to connect and communicate with their friends, colleagues and family but they have been used as a way to trigger and sustain revolutions and protests. Shirky in [20] argues that social media have the power to sustain political uprisings while others believe that the power of social media is overrated [14]. Nevertheless, there have been several examples of people using social media to oust dictators and protest against regimes. This is exactly what happened during the Iranian elections in 2009 and Egypt's uprising in the beginning of 2011. During these events people used social media and especially Twitter, to report news during protests and influence others to participate. The events that took place during the Egyptian uprising engendered an unprecedented flow of information and ideas on Twitter. People who were watching their Twitter timelines could literally see the events unfolding before their own eyes. One wonders what would

have happened if someone collected all this information and tried to make sense of them. What if we could be able to detect events and describe them as they are happening? These are the questions this project sets out to answer, not in a science fiction way but following a structured and scientific methodology. We have collected social media data from Egypt's uprising and we have tried to describe what happened. Using this case study as a starting point we developed a generic framework for detecting and describing events as they unfold. In the last few years a growing community of researchers started using Twitter to conduct research on social net- works and data mining. Recently, Lotan et al.[13] attempted to investigate the events that took place during the Egyptian revolution by collecting and analysing a large amount of tweets. They have identified different types of users (media organizations, bloggers, activists) in the dataset and they studied how each type was influencing the other by looking at the information flow between them. Another research project studied the information dissemination during the Iranian elections and offered important insight into the dynamics of information propa- gation that are special to Twitter [23]. Other studies have investigated event detection and summarisation during the Egyptian revolution as well as the different type of actors participating in the diffusion of news through Twitter. Mining and analysing Twitter data is not an easy task and research has revealed numerous challenges for researchers. The biggest challenge is the enormous amount of data flowing on Twitter every second requiring careful filtering in order to block unwanted tweets such as updates on someone's life or spam tweets. Additionally, an intrinsic problem of analysing on-line data is that they are not always related to real world occurrences. For example, Twitter specific memes such as #musicmonday(users tweet their music preferences) or #followfriday(someone suggesting to other users someone else to follow) produce a massive amount of tweets during a certain time interval but they do not correspond in real life events. Therefore, the motivation behind this project is firstly to overcome these diffculties and also contribute to the research community by providing robust and accurate tools for automatically describe and detect events from social media content.

1.2 Objectives

In this section we present our main objectives for this project by breaking down the main problems into smaller sub-problems. We believe that this breakdown will help us develop a robust framework for solving the problem by identifying the individual difficulties of each sub-problem.

1.2.1 Event detection

The main objective of this project is to develop a framework for event detection. We have collected over 100,000 tweets related to the revolution and our aim is to sift through them and identify the main events that took place. Below we present the formal definition of this problem along with the necessary definitions.

A **real-world event** identifies something non-trivial happening at a certain time and at a certain place **?**. We describe a real-world event to be a set of attributes describing that event such as keywords, geographic location, time of occurrence and the social actors (Twitter users) involved in it. It is possible to have incomplete or no information at all for an event therefore our system should take into consideration these possibilities.

A **tweet** is defined as a tuple (a, c, H, d, F_T) where $\boldsymbol{a}$ is the screen name of the author, $\boldsymbol{c}$ is the textual content, H is the set of hashtags associated with the tweet, $\boldsymbol{d}$ is the timestamp and F_T is the feature vector. The feature vector contains a set of attributes associated with the tweet such as the number of times this tweet has been retweeted, whether or not this tweet is a @-reply, the set of URLs and/or named entities if any.

A **tweet stream** is defined as $(t_1, t_2, t_3, ..., t_N)$ where N is the number of tweets in the stream, t_i is a tweet and $d_{t_i} < d_{t_{i+1}}$ (i.e. ti has occurred prior to t_{i+1}).

A **topic** is defined as $\{t_i | t_i \in T \wedge |A| \geq U \wedge F_P \wedge \forall t_i, t_j . t_i$ is similar to t_j with respect to this topic $\}$ where T is the tweet stream, A is the set of unique authors related to the topic and U is the minimum number of actors needed to be involved in the

topic. F_P is a feature vector containing attributes of the topic such as time, geographical location and keywords. The similarity of a tweet with respect to a topic is based on the tweet feature vector F_T as well as the textual content of the tweet.

Given a tweet stream T our aim is to partition it into a set of distinct topics $P = \{p_1, p_2, p_3, ..., p_{N_p}\}$ where N_p is the number of detected topics. Then based on the feature vector F_p we can associate each of the topics with one or more events.

1.2.2 Event summarization

When an event has been detected it would be interesting to be able to describe what happened by extracting a description in the form explained below. Therefore, we wanted to develop the framework for generating automatic summaries for an event. The formal definition of the event summarisation problem is given below.

Given a topic or topics associated with an event as described above, we aim to extract an **event label** which serves as a description for this event.

An event label is defined as a set of descriptors $D = \{d_i | i \in \{1, 2, ..., N\} \wedge |D| = X\}$ where N is the number of tweets associated with this event and a descriptor d_i is a tuple (c_i, d_i, H_i, s_i) where c_i is the content of tweet t_i, di the timestamp of tweet t_i, H_i is the set of hashtags associated with tweet t_i and s_i is the score of tweet t_i calculated by the summarization algorithm. Only the X highest scored tweets will appear in the event label.

The score is calculated based on several criteria such as quality, relevance and usefulness of a tweet. Quality refers to the textual quality of the tweet i.e. low quality tweets contain slang and short hand notation. Relevance is how well a tweet reflects information related to the associated event and usefulness is how well the tweet informs a human about the event.

1.2.3 Actor classification

In order to further understand the structure of a detected event we wish to identify the different types of users that are involved in the information dissemination on Twitter. Varying from media organisations and activists to normal individuals, these people were the people who ignited, documented and sustained the revolution. We aim to develop a tool for automatic classification of the users into different categories.

An **actor** is defined as $\{a, fr, fl, T, F\}$ where $\boldsymbol{a}$ is the actor's username, fr and fl are the sets of actor's followers and followees respectively, T is the set of tweets which belong to this actor and F is the feature vector. The feature vector contains the activity features of the actor such as the total number of tweets, the fraction of retweets among all the tweets from a user, the fraction of @-replies directed to other users and the fraction of tweets containing a URL.

An **actor label** is defined as $l_a \in \{c_1, c_2, c_3, ..., c_N\}$ where a is an actor, N is the number of existing classes and c_i is an actor type such as blogger, activist and media organisation. Given a training set of actors and their corresponding actor labels $\{(a_1, l_{a_1}), (a_2, l_{a_2}), ..., (a_N, l_{a_N})\}$ our aim is to find a mapping $l_a = f(a)$ which can predict the associated actor label l_a for an actor $\boldsymbol{a}$. The prediction will be based on the feature vector F and the other attributes associated with an actor such as the sets of followers and followees.

1.3 Contributions

In the process of developing the solutions for the problems described above we have come across several challenges as well as some opportunities. We have tried to tackle all the difficulties and also exploit the opportunities we were given. Our efforts have led to several contributions and the following list summarises the main contributions of this project:

- The solution of the event detection and summarisation problems led to the development of a data mining toolset which consists of several sub-

components. These components vary from data acquisition tools to clustering algorithms and text processing tools. These individual components can act independently to solve smaller tasks but most importantly they can be combined to solve the problems posed by this project.

- In order to be able to develop our solution we have conducted an extensive literature survey about event detection, event summarisation and Twitter user classification. Our research was focused on these three areas in the context of social media and more specifically on Twitter content. The emergence of social networks has sparked the interest of the research community and numerous solutions have been proposed over the last few years. We have gathered the most prominent solutions in the literature and we have commented on their applicability and feasibility in our project.

- The core component of our solution for the event detection problem is text clustering. The challenging aspect of our work is that we had to deal with social media content which offers some advantages but at the same time it poses significant challenges. After implementing several clustering algorithms we have conducted a thorough evaluation study to assess their individual performance in several aspects that are unique to social media documents. Additionally, the event summarisation and user classification components have been evaluated as well and our results can be used as a guidance to future projects in this area.

- We have built a proof-of-concept web application which uses our algorithm to detect events and summarize them using Twitter data. The application provides a user friendly environment and data visualisations to allow the user to discover events in historic data.

- We have collected and analysed a large number of Twitter content related to the Arab Spring and we have used this dataset to conduct a case study on real data using our algorithm and web application.

1.4 Report Structure

TODO: Complete this section when the rest of the report is done.

Chapter 2

Background research

In this chapter we present the state of the art for the individual problems of this project. We discuss the most prominent solutions and methodologies and we comment on their applicability in our work.

2.1 Event detection

2.1.1 Traditional event detection vs social media event detection

Traditionally, event detection research has focused on detecting news events from continuous streams of news articles. The majority of the work done in this area does not account for any social effects but focuses only on the textual properties of a news article. Therefore, researchers have used natural language tools such as named entity extraction and part-of-speech tagging in order to perform event detection **?**. However, in this project our problem exhibits several differences compared to traditional event detection since we aim to use data from Twitter.

Twitter and more generally social media documents exhibits several advantages in relation to the event detection task but at the same time pose significant challenges for the researchers. More specifically, a large number of social media documents are irrelevant to real world events as they usually describe updates on

the user's life or information unconnected with an event. About 40% of all the tweets are pointless "babbles" **?**. Additionally, social media documents contain little textual content which is usually restricted to a few characters (140 in the case of Twitter) which renders traditional event detection methods undesirable. On the other hand, social media documents provide us with new prospects since they often come with additional context information such as tags, locations and network structures.

2.1.2 State of the art event detection in social media streams

One of the first research projects that tried to detect events in Twitter streams aimed to inform people about the occurrence of an earthquake **?**. The explicit assumption in their research is that a large number of Twitter users are tweeting when an earthquake happens and therefore it is possible to detect its existence. They have developed an earthquake reporting system which monitors Twitter streams and reports the occurrence of an earthquake promptly. In order to detect tweets discussing earthquake occurrences they have used a support vector machine to detect real occurrences. In order to train their classifier they have prepared a set of training data consisting of positive tweets (tweets which mention an earthquake occurrence) and negative tweets. For each tweet they used different types of features to be used by the classifier:

- statistical features such as the number of words in a tweet message and the position of the query word within a tweet.
- keyword features such as the words in a tweet.
- word context features such as the words before and after the query word. By treating users as sensors they have built a socio-temporal model which can predict the occurrence of an earthquake and estimate the location using Kalman filtering.

After evaluating their reporting system in Japan for a month, they reported that 96% of earthquakes larger than JMA seismic intensity scale 3 or more were successfully detected.

Researchers at University of Edinburgh **?** proposed a novel method for detecting news events in Twitter streams. The main objective in their work was the detection of a"first story" in a stream of tweets. The task of First Story Detection (FSD) was first coined in **?** and aims to identify the first story to discuss a particular event. The traditional method of First Story Detection (FSD) is to represent documents as vectors in term space. Then each document is compared to all the previous ones in the stream and a similarity value is produced. If this value is below a particular threshold it is declared to be a "first story". However, the challenge they faced was that Twitter provides a vast amount of data in real-time which renders the traditional method inefficient. Therefore, in order to alleviate this problem they implemented a novel algorithm based on locality-sensitive hashing. Their method uses the cosine between two documents as the similarity metric and a hashing scheme proposed by Charikar **?**. This scheme limits the number of documents to be compared and increases the performance of the algorithm. They reported that the hashing scheme alone yields poor results with a lot of variance. In order to solve the problem they used a variance reduction strategy. They have tried their algorithm on Twitter data collected over a period of six months and they have showed that their system is able to detect major events with reasonable precision.

A different approach is proposed by Sayyadi et al. **?** in their paper "Event Detection and Tracking in Social Streams". The main idea behind their proposed algorithm is that documents describing the same event will contain similar keywords, and the graph of keywords for a document collection will contain clusters which are effectively events. They extract keywords from the documents and they construct a graph whose nodes are the keywords and edges between the nodes are formed when those terms co-occur in a document. Their algorithm uses betweenness centrality to assign scores to the nodes and the nodes with high betweenness centrality score are removed. This way they manage to reduce the

graph into several unconnected sub-graphs (communities). Then they consider each community of keywords as the key document/event with the keywords being a bag of words summary of the event. Documents in the original corpus which are similar to this key document can be clustered, thus retrieving a cluster of topical documents. Again in their methodology they used cosine similarity to discover document clusters for key documents. An important feature of their system is that they also took their approach one step further by proposing a sliding window approach for subsequent event detection in social media streams as it is considered impossible to apply the event detection algorithm to each new document arriving in the stream without performance degradation.

One of the best methodologies for event detection is proposed by Becker et al. **?**. In this study they focus on on-line identification of real world event content. A common problem of event detection which arises in Twitter is the false identification of non-event content as event content. For example, specific conversation on topics or memes (e.g., using the hashtag #followfriday) which do not correspond to real world occurrences might be incorrectly identified as events. In order to deal with this problem they decided to introduce a two stage process where initially each event and its associated messages are grouped together with an online clustering algorithm. In the next stage they calculate several features associated with each cluster in order to train a classifier to distinguish between event and non-event clusters. The main advantage of this method is that the features used to classify the clusters exploit the Twitter-specific features thus making the process of event detection much more robust. They evaluated their system by collecting 2,600,000 tweets posted during February 2010. They have used a nave Bayesian classifier as the baseline method and they showed that their method outperformed the baseline.

2.2 Event summarization

The first attempt to summarize events on Twitter was attempted by Chakrabati et al. **?**. Their research paper discusses the problem of extracting the tweets which summarize long-running events, such as football games and they propose

a two-step solution. The first part uses a Hidden Markov Model in order to partition the event time-line in sub-events (e.g, tweets before a goal and tweets after the goal) based on the burstiness and the word distribution in tweets. The second stage selects the key tweets of each segment and eventually all key tweets are combined to give the summary. The system takes as its input an event, which is detected using one of the existing event detection algorithms and it outputs a few tweets that best describe the occurrences in that event. Their proposed algorithm is able to isolate at most one sub-event in each time segment, which can then be used by the summarizer to output summaries. They use a variant of the Hidden Markov Model which models each event as a sequence of states. The tweets are the observations generated by the states and the transition between states models the chain of sub-events. They propose two other methods for summarization which are based on simple TF/IDF scores and segmenting the event time-line in equally spaced time intervals. They compare their algorithms performance in comparison to these baseline techniques for tweets pertaining to football games and they showed that their algorithm was performing particularly well. However, the authors state that they have not tested their system for one-shot long-running events such as revolutions which can be a significant problem in our project.

Becker et al. **?** proposed a different methodology for selecting quality tweets from the set of tweets related to a specific event. The goal of their summarization process is to extract tweets which meet the criteria of quality, relevance and usefulness. The extracted tweets should be comprehensible by a human (i.e. they do not contain short-hand notation), they must reflect the information related to their associated event and they must be useful in describing the main occurrences event. In order to satisfy these criteria the authors propose several methods for extracting these tweets. The main idea behind their approaches is that the tweets which are closer to the centre of a cluster (event) are more likely to reflect the key aspects of the event than other tweets. Therefore, their approaches are based on the notion of centrality. The first method, the centroid similarity approach computes the cosine similarity of the TF-IDF representation of each message to its event cluster centroid. Then the tweets with the highest similarity score are

selected. The second method involves message similarity across all messages in an event cluster. This approach, constructs a graph where a cluster message is a node in the graph, and there is an edge connecting a pair of nodes whose cosine similarity exceeds a threshold. This method selects the nodes with the highest degree centrality which is defined as the degree of each node, weighted by the number of nodes in the graph. Finally they use a state of the art method, namely LexRank which defines centrality based on the idea that central nodes are connected to other central nodes. They have evaluated their approaches alongside with some other baseline approaches such as selecting the newest tweets in the cluster and selecting tweets from popular users. The results revealed that the centroid method outperformed the others in all three selection criteria: quality, relevance and usefulness.

2.3 User classification

Twitter users have diverse backgrounds and they tweet for a variety of topics. There are many different types of users such as celebrities, bloggers, journalists, media, and organizations. Several studies have tried to identify different type of users on Twitter and explore their distinct characteristics. In this section we discuss various techniques which have been used to identify and classify users.

In a recent study of the Arab Spring, Lotan et al. **?** investigated how information was propagating during Egypt's uprising. In order to identify information flows they have collected a large amount of tweets during that period and identified the active users in the social network. They selected 963 users, from their dataset, who either were first to tweet, or were retweeted or mentioned at least 15 times. Then they manually classified these users in different categories:

- Mainstream media organizations (e.g., @AJEnglish, @nytimes).
- Mainstream new media organizations (e.g., @HuffingtonPost).
- Non-media organizations (e.g., @Vodafone, @Wikileaks).

- Mainstream media employees (e.g., @AndersonCooper).
- Bloggers (e.g., @gr33ndata).
- Activists: (e.g., @Ghonim).
- Digerati: (e.g., @TimOReilly).
- Political actors: (e.g., @Diego_Arria, @JeanMarcAyrault).
- Celebrities (e.g., @Alyssa_Milano).
- Researchers: (e.g., @JRICole).
- Bots: (e.g., @toptweets).
- Other: users that do not fall under any of the other categories.

It is apparent that since we are interested in automatic classification their method is not well suited in our case. However, their work has revealed several challenges of the user classification problem in Twitter which we must overcome in our implementation. The main challenge was that a number of users were very difficult to classify due to the fact that these users could be classified in multiple classes. For example, some users are bloggers but they are also activists and this led to ambiguities during the classification process. Furthermore, a number of accounts were deleted or suspended during the data collection process which inevitably introduced missing nodes in the social network graph.

Leaders, lurkers, associates and spammers are some of the type of users active in Twitter and another research project aimed to automatically identify them **?**. They designed two different methods for user classification: a context-dependent method and a context-independent method. They classified the users in four types: leaders, lurkers, spammers and close associates. Their research is based on several assumptions on these types of users. More specifically, they assume that spammers follow many users but they are followed be a few people and they

make on average 1,000 tweets per day. On the other hand, leaders are identified by the high rate of tweeting and a large number of followers but almost no followees. Closes associates have strong connectivity to their followers and low rate of tweeting. The final class, lurkers, follow many people, but they rarely post any tweets. When there is no contextual information about a user, such as their followers and @-replies they use the context-independent method which examines the tweeting pattern, otherwise they exploit these contextual information by employing the alternative context-dependent method. The context-dependent method employs a fuzzy logic approach in the first stage to estimate inter-actor relationship strengths and then they remove the links with strong relationships because they naturally represent close associates. Then, in the second stage, they use a linear classifier, using the number of tweets and the followee-follower ratio as two features. The context-independent method uses traditional classifiers and generic actor tweet patterns. They collected the tweets of particular types of tweeters over a period of ten days and then they used these to train three classifiers: nave Bayesian, multi-layer perceptron (MLP) and a random forest (RF) classifier. In their evaluation the MLP classifier has been found to outperform the naive Bayesian and Random Forest classifiers on the more challenging problem of classifying actors with limited data. Their research gives strong indications that user classification with Twitter data is possible with high performance.

A similar approach was taken by Choudhury et al. **?** in their research which aimed to automatically classify Twitter users in three categories: organizations, journalists and ordinary individuals. Their work offers two significant advantages compared to other methods in the literature. They used Twitter specific features to classify the users which make their system much more robust. They use standard features from previous studies such as network features (followers and followees) and activity features such as the number of tweets for a user. However, they added some additional ones including interaction features (number of retweets, @-replies and tweets containing URLs) and named entities features which capture the presence/absence of named entities in the tweets of a user. Finally, another important feature used in the classifier is the topic distribution which associates the user to a set of 18 broad themes from the IPTC Media Topic

News Codes. They have used 8 different classifiers in order to evaluate their design and the results indicated that the best performing classifier was a k Nearest Neighbors classifier with $k = 10$. Another important contribution of their work is that they investigated the participation of different categories of users in different events. They have showed that different events gather different degrees of participation (number of users from each category) and attention (proportion of posts from each category).

Chapter 3

A data mining framework for automatic event detection and summarisation

In this chapter we will define a theoretical framework for construction of a data mining system which can extract events from Twitter data. The system comprises of a number of individual components which when combined together construct a complete framework for event extraction. These components are explained in detail in the sections below and in Chapter 4 we discuss the concrete implementation of this framework as well as the development of a proof-of-concept application using this framework.
The first section of this chapter introduces the motivating work for the implementation of the system and we take a closer look at the concepts and key ideas that are necessary to understand the process we employ.

3.1 Methodology overview

The main idea behind our methodology for event detction is that similar events will be described by tweets having similar content. Therefore, the main task in our methodology is to cluster the tweets in different groups. Having done that we can then further investigate these clusters and identify which ones are events and

also extract useful information from them. This procedure is described in detail in the rest of this chapter and Figure 3.1 depicts an overview of the system. Initially, historical tweets from a service provider (Twitter API or another provider) should be retrieved and stored in an appropriate format in the database. Subsequently, the system receives a stream of tweets from the database and process and transforms them in a format that is appropriate for clustering. The next step in the pipeline is the actual clustering of the tweets in order to detect groups of tweets discussing the same topic. Then, the extracted clusters are processed in order to identify the events and generate their summaries. Finally, a visual representation of the results should be generated in order to aid understanding of the events.

In our discussion in this chapter we will leave out the data retrieval aspect since it is dependent on the actual implementation and it will be described in detail in Chapter 4. For now we'll assume that we have a source of tweets,such as the Twitter API which provides us a collection of historical tweets.

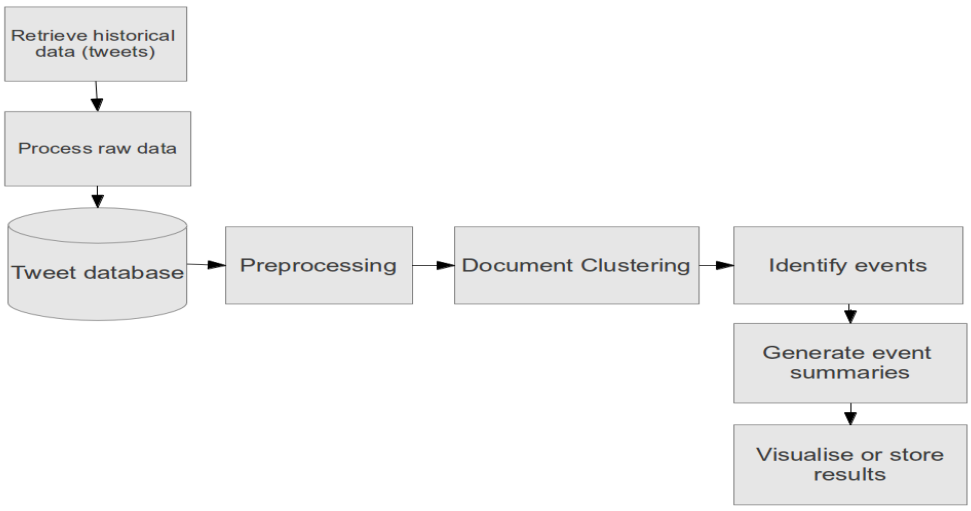


Figure 3.1: System overview - The event extraction system comprises of several independent components.

3.2 Raw data processing

Mining and analysing text corpora are two well studied problems but in our case we face a slightly different problem. Social media content and especially tweets are very short (140 characters) and usually contain slang phrases, abbreviations and irrelevant information. Therefore, is of foremost importance to ensure that clever data preprocessing takes place in order to help us in the subsequent tasks.

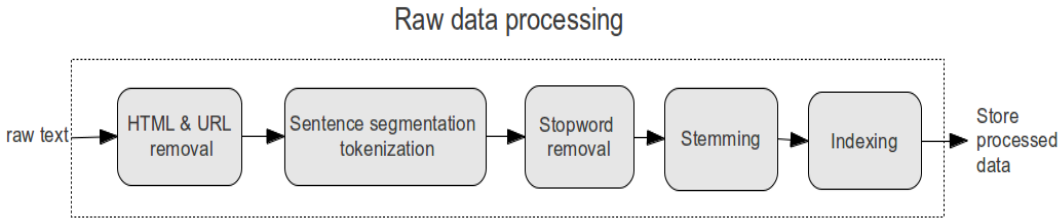


Figure 3.2: The raw data processing module - All the steps neccessary to convert raw documents to a format suitable for storage in a database.

3.2.1 Natural language processing

Natural language processing (NLP) is a field of computer science and linguistics which is concerned with the extraction of information from a human language input. NLP is used extensively throughout this project since we deal with human generated content and we use several of its applications. NLP in the context of raw data processing is used to clean and normalise our initial input which is a tweet containing raw text.
Since tweets usually contain URLs and HTML code one of the first priorities is to remove these elements. Then, the following NLP algorithms can be used on our data:

- Sentence segmentation and tokenization: Raw text is split into sentences and then each sentence is further subdivided into words using a tokenizer.
- Stopword removal: Some of the words occuring in the documents are common English words such as 'the', 'and' and 'a'. These words convey almost

no information but most importantly, they can reduce the clustering accuracy. Two documents can be erroneously considered related if they contain common English words like the ones we have already mentioned. Therefore, it is important to remove these stopwords. Usually, a dictionary of the common English stopwords is used to filter them out.

- Stemming: This is a mechanism for reducing English words to their stem form. For example, the stem form of the words 'connections' and 'connected' is 'connect'. This mechanism is particularly useful in the field of information retrieval and indexing. M.F Porter designed the most widely used stemming algorithm in 1980 [Insert ref here]. This algorithm works by applying a set of different rules, which after a number of iteration yield the final resut which is the stem of the word. Porter has developed 62 rules which may or may not apply to a given word.

In section **??** we explore more applications of NLP and how we can use them to extract automatic summaries, named entities or sentiment from documents.

3.2.2 Inverted index

An inverted index, also known as an inverted file, is a data structure central to text-based information retrieval. The name is derived from its purpose and design which is to map key-value pairs, where a key is a term in a document and the value is the list of documents that contain this term. For example if we have two documents:
Document1: The cat is on the tree.
Document2: The cat sat on the mat.
then the inverted index will look like:

Key	Value
the	{Document1, Document2}
cat	{Document1, Document2}
is	{Document1}
on	{Document1, Document2}
tree	{Document1}
sat	{Document2}
mat	{Document2}

The main reason for using an index is to increase the speed and efficiency of searches of the document collection. In our system the inverted index is vital component since it allows us to construct term-documment vectors easily and also filter terms and documents. For example, using our index we can find the words that appear either too often or less frequently and filter them out. This is used to reduce the dimensionality of our dataset by removing unneccesary words. Alternatively, we can remove documents/tweets which contain keywords that appear too often or less frequently.

3.3 Clustering

Clustering is the process of grouping a set of data objects into multiple clusters where all the objects in a cluster have high similarity but they are very dissimilar to objects in other clusters. Dissimilarities and similarities are calculated based on the attribute values of each data object and they often involve distance measures. Clustering is an unsupervised learning method since the object labels are not known beforehand. In our project we are interested in clustering because we would like to cluster tweets which are similar in terms of both textual content and social content. Additionally, due to the fact that we have to deal with vast amount of data we are interested in scalability and performance.

3.3.1 Vector space representation

Find a suitable place for this paragraph: ——————————— Preprocessing is also required to transform the documents (tweets) in a form that will

Document	team	ball	football	countries	world	england
Document1	3	0	5	1	3	1
Document2	2	1	2	0	8	2
Document3	1	5	3	0	1	3
Document4	5	0	1	2	5	4

Table 3.1: Term-frequency vector representation of documents

allow us to perform clustering on them. Therefore, we define 'preprocessing' as all tasks which take place before transforming the documents into the vector space representation. Figure 3.2 show the sub-components of the preprocessing module.

We can represent each document in a dataset by a vector of identifiers. Usually, these identifiers are the distinct words in the document and the resulting vector is called term-frequency vector. If we combine all the vectors for the documents in our dataset we will end up with an $\boldsymbol{m} \times \boldsymbol{n}$ matrix $\boldsymbol{A}$. Each of the m documents in the document collection are assigned a row in the matrix, while each of the n unique terms in each document are assigned a column in the matrix. A non-zero element a_{ij} in $\boldsymbol{A}$, indicates not only that term j occurs in document i, but also the number of times the term appears in that document. Since the number of terms in a given document is typically far less than the number of terms in the entire document collection, A is usually very sparse.
For example in Table 3.1 we see that *Document*1 contains three instances of the word team, while football occurs five times. We can also infer that the words ball and world are missing for the entire document, as indicated by the value of zero at those entries of the matrix.

The main advantage of transforming the documents in the vector space is that we can define vector-space similarities between documents and therefore we can apply clustering algorithms on the document collection. In section we provide a more detailed discussion on similarity metrics and the vector space representation is used in clustering documents.

3.3.2 Assigning weights to terms with TF-IDF weighting

Once a document is transformed in its term-frequency vector we can assign weights to each term in the vector. So far the term-frequency vectors treat all the

terms as equal but this may not be the case. For example, a word that appears more frequently than others in a document could be considered as an important word. At the same time words that appear frequently in the corpus, such as 'a', 'the', 'and', are not very useful and their importance must be discounted.
The most common method to solve this problem is the TF-IDF weight (TF-IDF stands for term frequency-inverse document frequency) which quantifies the importance of a term in a document of a document collection. More specifically, the more a word occurs in a document, and the less it occurs in the rest of the coprus, the higher its TF-IDF weighting will be. Mathematically, TF-IDF is expressed as:

$$tf - idf_{t,d} = tf_{t,d} \times idf_t \tag{3.1}$$

where $tf_{t,d}$ is the importance of term t in document d and idf_t is the importance of term t relative to the entire corpus. $tf_{t,d}$ is higher when the term occurs many times in the document and idf_t is higher when it occurs rarely in the dataset. Therefore, the TF-IDF weighting for a term is very high if the term occurs frequently in a single document but very rarely in the entire corpus and it is low when the term either occurs rarely in a document or frequently in the entire corpus. TF-IDF is widely used to compare the similarity between documents and a common use case is for search queries where the similarity of a query q with a document d is calculated using TF-IDF, providing a sorted list of the most relevant documents.

3.3.3 Feature selection

TODO: Discusss feature selection methods used in our system.

3.3.4 Kmeans algorithm

3.3.5 DBSACN algorithm

3.3.6 Non-negative matrix factorisation algorithm

3.3.7 Online clustering algorithm

3.4 Automatic text summaries

3.4.1 Background

3.4.2 Generating automatic document cluster summaries

3.4.3 Detecting sentiment, named entities and locations in documents

3.5 Twitter user classification

3.5.1 Background

3.5.2 Decision trees

3.5.3 Neural networks

3.6 Summary

Show a simple diagram with a lot of tweets becoming vectors and then those becoming clusters and then some of them events. This is the outout of the algorithm.

Chapter 4

Developing a proof-of-concept event detection web application

4.1 Motivation

Why did we created this web application?

4.2 System architecture

Explain briefly in several subsections the individual components of the GUI tool such as how do we retrieve data (Topsy API), what framework are we using (Tornado) and how we produce the visualizations (d3.js)

4.3 Graphical User Interface

Explain and show screenshots from the prototypical application.

4.4 Summary

Chapter 5

Evaluation

5.1 Document clustering evaluation

5.1.1 Background

Explain the metrics and the method used.

5.1.2 Results and discussion

Present and discuss the results

5.2 Cluster labelling (summarization) evaluation

5.2.1 Background

Explain the metrics and the method used.

5.2.2 Results and discussion

Present and discuss the results

5.3 Twitter user classification evaluation

5.3.1 Background

Explain the metrics and the method used.

5.3.2 Results and discussion

Present and discuss the results

5.4 Summary

Chapter 6

Conclusions

Here I put my conclusions ...

Appdx A

and here I put a bit of postamble ...

Appdx B

and here I put some more postamble ...

References