EVENT/SUB-EVENT DETECTION FROM MICROBLOGS

By

Giannis Alexandrou

A thesis submitted to
the University of Cyprus
for the degree of
COMPUTER SCIENCE

Department of Computer Science
University of Cyprus
May 2020

ACKNOWLEDGMENTS

I would like to express my deepest gratitude to my supervisor Dr. George Pallis, Assistant Professor in the Computer Science Department of University of Cyprus, for his endless willingness and the opportunity that gave me to work my thesis in one of the biggest new-age problems. Furthermore, I thank him for all of his support and encouragement that provided me throughout the progress of this thesis project.

Moreover, I would like to thank the Ph.D. candidate Demetris Paschalides for his excellent guidance and the chance he gave me to learn an incredible amount of technologies.

Last but not least, I must thank my family and my best friends who have given me constant support and love during the completion of the thesis. In particular, I owe eternal gratitude to my mother, who has encouraged me at every step in life.

ABSTRACT

Event detection techniques in social networks have proposed as a way to sense and understand what is happening in the offline and online world. Despite being a challenging task, event detection has utilized in applications that target critical aspects of peoples' lives, like health, natural hazards, and sports events like football matches. In this thesis, we look into the different techniques of event detection in micro-blogs. Specifically, we focus on the Twitter platform due to the popularity it, as well as the ease-of-access to vast amounts of data, through their API. Therefore, to do so, we gather a large data-set, and we process in order to detect the events. In other words, by collecting our data-set, we clean our data using Natural Language Processing methods. After data preprocessing, we vectorize our data, creating a bag of words vocabulary for each time series and extracting features from them. The next step is to use our feature extraction to generate clusters and set them as candidate events. After collecting events, we need to process, organize, and make sense of them. We need to use algorithms that identify the topic of the event, where it happened, and whether it is newsworthy. Finally, we summarize each event that we detected.

DEDICATION

I dedicate this thesis to my parents Andreas and Elenitsa. I hope that this achievement will complete the dream that you had for me all these years ago when you chose to give me the best education you could.

Contents

List of Figures

List of Tables

Chapter One

Introduction

Contents

1.1 Motivation

It is a common fact that we live in a world where every single person globally, has the right to use social media platforms to express his feelings, emotions and even describe actions that happened all over the world. These actions in platforms often lead to an abnormal situation when something happens at a specific time and place. More specifically, Twitter is a platform that presents the above features and allows users to "See what's happening in the world right now.". Twitter's reputation for detecting events has surged over the years, evolved it from being an entertainment blog discussion to a primary source of news content.

Nowadays, big companies and organizations try to take advantage of the abilities that Twitter has to offer. Sometimes, by tracking a known event that it will occur at a particular time and location, and other times using tools to extract breaking news in real-time. Figure 1.1 indicates the number of tweets that were posted and

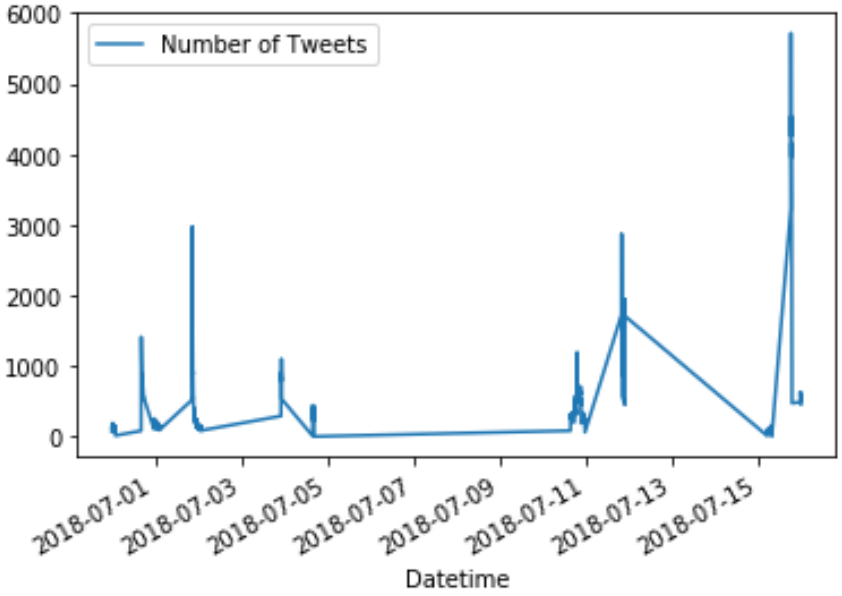


Figure 1.1 The frequency of tweets during the FIFA World Cup 2018

talked about the tournament FIFA World Cup 2018 over 17 days from 29/06/2018 to 15/06/2018. What stands out from this graph is the occurrence of spikes, which point out an anomaly on the track. Hence, it could reflect a high probability of an event occurring. Another interesting point is the peak of 15/07/2018, where it was the day when the final of World Cup 2018 conducted. As a result, the trend of the figure witnessed a rapid incline, which helps us to understand how Twitter users usually react to an event appearance.

Despite being an essential tool for event detection, the usage of Twitter should manage appropriately to extract features and make sense of them. Surely, in a real-time platform like Twitter, where users can share whatever they want, there is a high probability of many risks for anyone who wants to present events to the audience that are newsworthy. In this research, we focus on how Information Retrieval and Clustering techniques which will help us recognize candidate events.

In general, we can define events as real-world occurrences that unfold over space and time [4],[29]. Moreover, they can be divided into planned or unplanned events.

Planned events are the kind of events that are predefined with already known content, location, specific date and time, and with a group of people related to their preferences. On the other hand, an unplanned event occurs without any prior information about when, where, and what happened. Furthermore, event detection techniques can be classified into specified and unspecified techniques. Unspecified techniques are utilized by events that occur in real-time by focusing on the volume and the temporal signals of Twitter Streaming. In contrast, specified techniques rely on specific information about a predetermined event like location, type of event, and a particular time. Last but not least, taking advantage of the unique characteristics of a tweet like hashtags, mentions, and keywords in the text can extract useful features to track events.

1.1.1 Problem Formulation

We define the following terms in order to formulate the problem of event detection:

Message: A tweet message is consisted of id, user id, content, entities, timestamp and location. Each message has its unique id and a user who shared it in the Twitter platform at a specific timestamp. Moreover, the message contains content which, in our case, is a text in a tweet format. Also, a tweet can provide a plethora of entities, like hashtags, mentions, likes, etc. Besides that, each message can provide the location where was posted with different ways like coordinates(longitude, latitude).

Stream: Twitter platform can provide a Streaming API, which contains N messages for each time-window i. Moreover there are I time-windows. Therefore, a stream can be formalized as S_i = [m_1,m_2,...,m_N] where i $\epsilon_{1,2,...,I}$.

Event: An Event can be formalized as E = [T,M,C], with T defined by t_0 which

is the time of the first message shared in the event E occurrence, and t_1 the time of the final message related to the current event E. Furthermore, M is defined by the set of messages that represents the event E. Finally, C is defined by the representative content that can be used to summarize a specific event E.

Problem Statement: Given a stream S of messages for each time-window i, detect all possible events E_i within S.

1.2 Challenges

By definition, the ability to automatically detect and track ongoing real-world events has surely attracted big organizations' interests. Nevertheless, it is a challenging task from every perspective. For every journalist, it is tough to recognize the useful information on Twitter without being overwhelmed by an endless stream of irrelevant tweets. Another issue that a journalist has to face is the reliability of the information that will retrieve from Twitter. Sometimes, for companies like BBC, Reuters, and CNN exporting breaking news as fast as possible, they usually faced the challenge to prevent being fallen into the trap of non-reliable happenings.

Furthermore, using natural language processing tools, we can read, decipher, understand, and make sense of the human languages in a valuable manner. Despite all the comforts, NLP methods may fail in many circumstances, and sometimes we might not be precise as we wanted at the outcome. Specifically, NLP requires structured sentences for processing and extracting useful entities efficiently. The challenge that we need to face in our research is how we are going to process tweets that are not structured sentences with lots of spelling mistakes and with different disciplines. We had to study how to manipulate tweets features and use them as an advantage to our clustering system.

Another challenging part of our research was the difficulties of the data-set that we utilized due to many reasons. Firstly, 48% of tweets deleted, with the result being unable for some events to be detected from our system. Another problem is the variety of languages of the tweets that we managed and overcoming this obstacle by focusing specifically on English tweets. The large scale of noise tweets in our data-set was another restriction for our event detection. Because our focus was to generate clusters with high purity and not for noise filtering, we concentrate on how to use clustering methods that can recognize the noise with a high probability. Lastly, collecting data was another part of the research that was challenging due to the difficulty of finding data with labeled tweets that have classified into an event. In the majority of studies, the data-set was not available publicly, and we need much investigation to find something relevant.

1.3 Contributions

The ultimate goal of our research is to examine how different features from a Twitter corpus can be utilized with the combination of natural language processing tools, to investigate the new age problem called Event Detection in Micro-blogs. Initially, we needed an appropriate data-set of tweets that can represent real-time data with a large amount of noise. Therefore, we collected large-scale tweets which contain a certain number of the relevant and large number of irrelevant tweets.

Moreover, we decided to focus on three primary features for our feature extraction; Textual features, Social features, and N-gram analysis. For textual feature extraction, we needed to clean our data by utilizing a variety of natural language processing tools. For social feature extraction, we needed to gather specific attributes for each tweet, such as hashtags, mentions, URLs. Finally, we utilized those two features to create n-gram keywords. Because by extracting the appropriate features

is one of the main contributions in our study, we examined each combination of n-gram analysis to find the best possible outcome.

Furthermore, we examined which is the most appropriate clustering approach regarding our requirements. We needed to generate pure clusters with arbitrary schemes that are related to a specific relevant topic and also minimize as much as we can the clustering of irrelevant tweets. After lots of experiments, we deduced that DBSCAN (Density-Based Clustering Algorithm) algorithm is the clustering approach that managed to provide the best possible outcome. We also present a visual representation of our experiment results and a table with specific values of each metric for every combination.

Last but not least, we decided to make a summarization of each cluster and derived the most characteristic and common features for every cluster. On this step, we managed to extract the most representative title for each cluster, including the most common social features. Therefore, this step will help us for further studies on this research, like merging clusters and monitoring detected events.

1.4 Outline Contents

Chapter 1: Introduction

Chapter 1 introduces the importance of social media platforms regarding the detection and tracking of real-time events. It also defines and formulates the problem of Event Detection from Twitter. Furthermore, it describes the challenges that we faced through the research and the contribution of this particular work.

Chapter 2: Literature review and related work

Chapter 2 focuses on an analytical review of the literature, which consists of work

related to Event Detection on Twitter. More specifically, we examine previous studies on event definition, topic detection and tracking, and feature extraction.

Chapter 3: Methodology

The methodology chapter defines our methodology and explains each step that we made in detail. We describe the collection of each data-set that we utilized and its role in our study. Moreover, we analyze how we process our data to extract them for our clustering method. Besides that, we describe how our chosen clustering algorithm works and the reason we select it. Finally, we analyzed the structure of how we summarized each cluster.

Chapter 4: Evaluation

Chapter 4 describes the metrics for our evaluation and focuses on presenting the results for each combination that we made. We present a visual comparison of our results, and we provide a table with specific values for each experiment. Moreover, we analyzed the tracking of a particular detected event and irrelevant clusters to draw some important conclusions. In the end, we summarize our findings from the evaluation metrics and give some insights and thought through the experiments we took.

Chapter 5: Conclusion

Chapter 5 defines our overall conclusions and summarizes them to provide a brief outcome for the reader. Finally, we propose some future works that will help in our system's improvement.

Chapter Two

Related work

Contents

2.1 Defining an Event

Digging into many articles, we realize that there was a variety of definitions about the term Event. Panagiotou et al. 2016 [23] investigated a series of definitions, according to event detection or similar problems. Beginning with the TDT project [3], they define an event as "something that happens at a specific time and place with consequences.". They explain the consequences of peoples' activities as reactions that will flash in the network activity. This definition is the foundation of the terms that were stated in the following years, especially events in social media. Hence,

Aggarwal et al. [2] defined the event as "something that happens at a specific time and place but is also of interest to the news media", which introduced the affection of social media platforms in the investigation of news events.

A similar explanation of event was stated by McMinn et al. 2013[20], where they focused on the influence of news media. They directed into the meaning that if something discussed in media, then undoubtedly is an event. Furthermore, they declared that the representation of the events is the entities that occurred during the detection and tracking of them. In other words, names of people, locations, and organizations describe a specific event.

Another interesting perception of the term event, and specifically in social media, is the observation by Hasan et al. 2018 [11]. They define an event as an appearance of topics that attract the attention of people in the real world, to discuss and express their opinion before, after, and during the occurrence of them. Dou et al. 2012 [7] declared a similar definition of 'event' as a situation that changes the volume of text data for a special topic at a specific time. Thus, this pair represented by entities like people and location with the result of generating an event.

Panagiotou et al. 2016 [23] concluded that there are a plethora of types explaining the term "event," and each one requires a different path for investigation. More particularly, they defined five different types of events; Planned, Unplanned, Breaking News, Local, and Entity related. Firstly, "Planned' are events that their time and location are predefined(e.g., a football game). Secondly, "Unplanned" are events that could happen at any time anywhere without being planned (e.g., disasters). Thirdly, "Breaking News," which are the type of events that are presented mainly in news media (e.g., the result of elections). "Local" are events that occurred in a specific geographical location. Finally, "Entity Related" is a type of event that talks about a specific entity (e.g., famous actors or the president).

All of the above definitions have a specific investigation and impact on the declaration of an event, though I am focusing on Unplanned detection of Events in this study as we mentioned in the section 1.1.

2.2 Event detection on Twitter

With the majority of benefits that social platforms have to offer, Event Detection on Twitter held the attention not only for researchers but also for international news organizations such as Reuters Tracer. [16] implemented a system to help their journalists discovering breaking news, verifying them, and informing the public before other news agencies. They divided their implementation strategies into two key system components, machine learning and data processing architecture. The system starts by pipelining real-time tweets to process large scale data. At next, it uses machine learning techniques by keeping English texts only and filtering the data from noises. Their hybrid approach of noise filtering is to categorize tweets into seven types; Spam, Advertisements, Mundane/everyday conversations, General information, Events, News, and Breaking News. After getting rid of noises, the system generates clusters based on the similarity of their content. Focusing on news discovery, they have to use FSD so that the system will be able to detect stories that are not breaking on social media. Subsequently, the system summarizes each cluster with its representative sentence so that users understand the description of it. Once the clusters being summarized, Reuter Traces use a classification model to categorize each of them into several topics. After being classified, Reuters wanted to recognize if a particular cluster is newsworthy. Hence, they have built a newsworthiness ranking model in order to understand if the cluster worth it as an event or not. Finally, they have used a prediction model in order to avoid rumors, fake news, and other misinformation. With this significant implementation, they managed to

discover real-time news faster than any traditional media.

Saravanou et al. 2018 [24], were concerned with the event delineation problem. It is an unsupervised technique that identifies the main event and captures the highlights of it at the same time. They construct a dynamic, heterogeneous network with graphs with both user and content nodes. The links between nodes are "user-user" edges, "user-content" edges and "content-content" edges. A graph in a particular time window contains several connected components that illustrate a discussion over a topic. Thus, the model creates a snapshot network, and if there is a sub-graph that contains a large number of connected components, then it shows a candidate event due to the majority of interest. Another similar model is by Meladianos et al. 2015 [21] and bases on a graph-of-words network. This network constructs a graph by the input set of tweets, and the detection is based on the tweeting rate. As long as each term weighting is becoming higher, the more representative is for the event detection on each time window. Thus, every 60 seconds, the system builds a weighted graph-of-words using the tweets posted during that period. Finally, a threshold value is set for every period, and when the number of tweets overcomes it, the graph is considering as a candidate event.

The usage of aggressive tweets and term filtering is another remarkable research that was introduced by the paper Ifirim et al. 2014 [12]. They combine an aggressive data preprocessing to eliminate tweets that were not aggressive under their investigation requirements; the number of hashtags, the number of words in the text, the appearance of n-gram words and the rank of resulting clusters. Following that, the model generates clusters of tweets by the hierarchical method using the value 0.5 as the predefined threshold. Their approach, although simple, shows encouraging results. However, they processed specific topics for their evaluation of the model focusing on the US presidential elections in 2012 and recent events that related the disasters in Ukraine and Syria in February 2014.

2.3 Topic Detection and Tracking

The TDT project began in 1997, and their motivation was to build a system that would be able to monitor news and could produce a signal when a new event appeared [3]. The project was organized by four methods, according to the technique they utilize.

2.3.1 Clustering Detection

Clustering Detection is an unsupervised technique that systems used at least in a first stage, as a Stream clustering task [23]. Grouping the texts based on topics they discuss, the system is cable of separate the clusters between "event clusters" and "non-event clusters". This kind of detection firstly adopted by the EventTweet system [1], which is base on location and keywords for each tweet. Keywords get a score according to their burstiness, which is measured to decide if a cluster is an event. At the end of each timestamp, clusters with high scores are marked as candidate events, while the other clusters filtered out as noise. A similar approach is in [22], where the authors utilize the semantic relationships of terms during the clustering procedure by using TF-IDF vectors. Cosine similarity is used as a distance metric in order to group tweets that discuss the same topic. [5] and [6] extend this approach by using density-based clustering methods OPTICS and DBSCAN, respectively. They also vectorize their terms for each tweet as N-gram sentences.

2.3.2 Anomaly Detection

Anomaly Based Event Detection focuses and tracks on abnormal observations [23]. In other words, if a group of unexpected terms or emotions demonstrates increased

usage in the last time window, then an anomaly is observed. According to [26] and [27] an event occurred when it affects the emotional state of a group of people that are close to the event.

2.3.3 First Story Detection

Another strategy of topic detection is First Story Detection. Systems using FSD, detect events when new texts that are similarly linked together to create "event threads", otherwise it is considered as a new event. A hashing technique such as LSH [25] is used in order to provide the approximate nearest neighbor in constant time. FSD is also called News Event Detection(NED), which targets at events from live streams in real-time. This approach is chosen by Reuters Tracer [16] so that the system will be able to detect stories that are not breaking on social media.

2.3.4 Topic-Specific Detection

Some other systems, follow the path of specific topic detection, by tracking events that are predefined such as disasters, music bands, football games, etc. After a topic is identified, systems are tracking the trend and try to detect anomalies of it. Take as an example in [30], where the authors monitored the events that are detected in the NFL 2010-2011 games. Although this type of method is not sufficient for real-time detection, it might be helpful for tools that are identifying real-time events and wanted to track the trend of them by using common terms such as hashtags, URLs, content words, etc.

All of the above techniques have a unique impact on topic detection and tracking. However, in this report, I am focusing on clustering processing.

2.4 Feature Extraction

Feature extraction constitutes one of the most crucial parts of the machine learning process because it reduces the data into more manageable groups of processing. In other words, it is cable of taking a set of data and convert it into vital and non-redundant information. Except for machine learning processing, feature extraction is also useful for other related fields in Artificial Intelligence such as pattern recognition and image processing alike, which is used mainly for dimensionality reduction. More specifically, there are a plethora of approaches to extract the features from a set of tweets from textual processing to social features.

2.4.1 Textual features

Textual features are based on similar words that users posted, but in order to extract them, we need to apply some data cleaning methods. According to TDT project [3], filtering out stopwords, stemming and tokenization techniques need to be subjected to be cable of extracting features of them. Tokenization is the process where a sentence or a paragraph of text is split into tokens of individual words or sentences. Stop words removal is the process where commonly used words (e.g we, these, your, his, through, me, were, her, more, himself, this, down, should, our, while, above, both) are eliminated from a list of tokens which is taken after the tokenization method. Stemming is the action of transforming every token into its root word by changing the form into its base. For instance, the words "Consultant", "Consulting" and "Consultative" will be reduced to the word "Consult".

2.4.2 N-gram features

An extending approach of textual extraction is the procession of N-gram sentences. N-gram analysis is a popular feature identification, especially for Twitter, where

each user can use a variety of text structures for a tweet. TwEvent and EventRadar are systems that implement this idea by using tweet segments(N-grams) instead of unigrams for their creation of bag-of-words [14], [6]. Then, features are extracted by the TF-IDF procedure, and tweets are considered as event candidates if they are close in space and time.

2.4.3 Parsing & Tagging Features

The recognition of name entities for a tweet is another challenging task that many researchers willing to investigate. Initially, by using the POSTagger in the NLP library, we can extract specific tags for certain words. As a result, we can manipulate specific tags such as verbs and nouns to provide more accurate results. Furthermore, the NER is another subtask that is used to extract and classify named entity mentions like person names, location, companies, percentages, dates, etc. With the above two methods from NLP, systems like [19] can extract all nouns, verbs and named entities from each tweet. A challenge of an entity-based approach is finding systems that identify entities from tweets precisely. Besides that, the authors from [15] examined the performance of NER on Twitter, and their results show that out-of-the-box solutions, such as the Stanford NER, perform adequately for most tasks.

2.4.4 Social Features

Another beneficial feature extraction is Twitter's features, which are included by all tweets. Common hashtags, mentions, RTs, and URLs can help systems generate clusters that related to candidate events more effectively. A lot of systems rely on these features, especially at the beginning of their event detection methods. For example, in Reuters Tracers [17], they start their clustering algorithm by grouping tweets that being retweeted between each other or having common hashtags. More-

over in [24], their purpose was to create a snapshot network with connected tweets related to Twitter's features.

Table 2.1 concludes the techniques for Event Detection from different papers that we mentioned above.

No	Author name	Ev.Detection Techniques	Description
1	Liu et.al. 2016 [16]	Real-time processing, Noise Filtering, FSD	Breaking and validated news
2	Saravanou et. al. 2018 [24]	Snapshot Network contained by related tweets	Identification of the main events and their highlights
3	Meladianos et. al. 2015 [21]	Graph-of-words network, Term weighting	Identification of main events and their highlights
4	Ifirim et. al. 2014 [12]	Aggressive data preprocessing, Hierarchical Clustering	Usage of twitter features combined with text processing
5	Abdelhaq et. al. 2013 [1]	Burstiness detection	Clustering based on location and keywords
6	Ozdikis et. al. 2012 [22]	Clustering procedure by using TF-IDF vectors	Cosine similarity as distance metric
7	Boettcher et. al. 2012 [6]	DBSCAN Clustering	Feature Extraction by TF-IDF vectorization, N-gram analysis

8	Ankerst et. al. [5]	OPTICS clustering	Feature Extraction by TF-IDF vectorization, N-gram analysis
9	Valkanas et. al. 2013 [26]	Anomaly-Based Detection	Affects the emotional state of a group of people that are close to the event
10	Mcminn et. al. 2015 [19]	Entity-based Event Detection	Usage of POS Tagging and NER tools
11	Zhao et. al. 2011 [30]	Topic-Specific Detection, Predefined Event analysis	Not efficient for real-time event detection

Table 2.1 An overview of different Event detection systems

Chapter Three

Methodology

Contents

3.1 Methodology Overview

Our methodology is based on four major pillars: Data Collection, Features Extraction, Event Clustering, and Event Summarization. First of all, we are bound to collect our data either by Twitter Streaming or within a corpus that includes labeled tweets for further evaluation of our model. Consequently, we have to clean our data in order to extract features from them. Then, we need to use these features and group our data by using the appropriate clustering method, which in this case is the DBSCAN algorithm. After tweets being grouped, the next step is to summarize each cluster by finding the most representative tweet, and also make some conclusions like the most common hashtags, mentions or URLs. An overview of our architecture depicted in Figure 3.1.

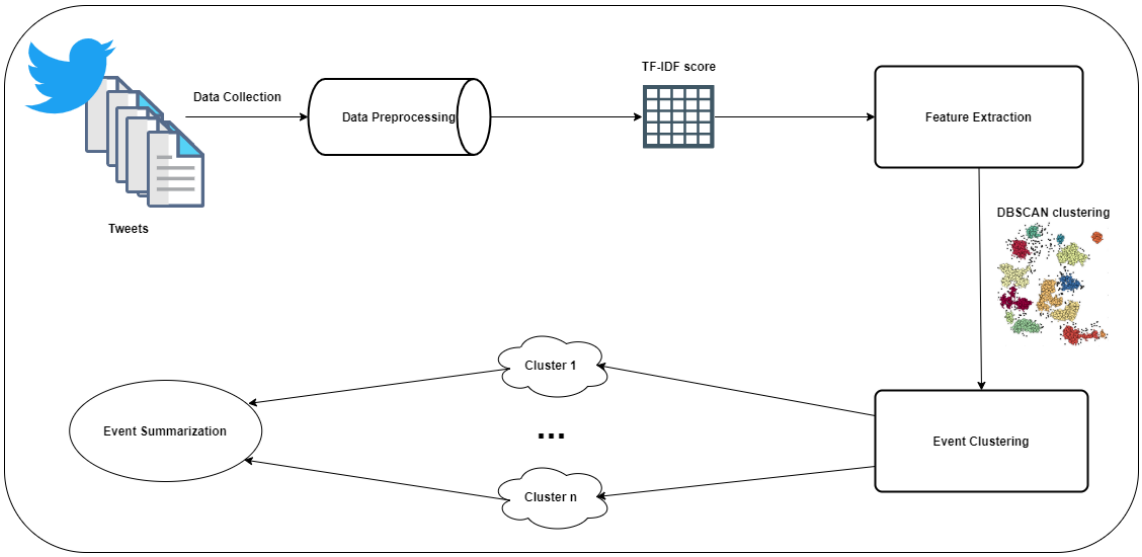


Figure 3.1 An overview of our methodology

In this research, we are not focusing exactly on detecting the events from the corpus. However, we want to create clusters that have purity, homogeneity, and completeness related to their topics that are talked about. Another challenging part that we focused on was to diminish as much as we can the possibility of noise clustering. Achieving these steps, we will give access to further investigations in the future like cluster classification, newsworthiness and velocity rating.

3.2 Data Collection

The first step in our methodology is the Data Collection, which means finding the best data-set for our preprocessing and experimentation. Finding the appropriate data-set is tough due to the nature of the problem. We need to process data in real-time with labeled tweets that are related to a particular event. We searched from different sources, and we conclude on the data-sets mention in Table 3.1.

At first, we used the data set of FIFA World Cup 2018 mainly to understand the tracking of a real predefined event. By analyzing tweets of a specific event, without noise tweets interrupting our procedure, we were able to witness the features that define it, the upwards and downwards of the trend, its peaks, its bottoms, the

burstiness of tweets when something remarkable had happened, etc. Although we got the factors of what an event represents, the data-set was not the appropriate one for our evaluations since it is lack of noise data, which plays a crucial role in assessing our approach with real-time data.

Index	Name	Rows	Download
1	FIFA World Cup 2018 Tweets	503,001	https://www.kaggle.com/rgupta09/world-cup-2018-tweets
2	Relevant tweets - Event 2012	152,951	http://mir.dcs.gla.ac.uk/resources/ [20]
3	Irrelevant tweets - Event 2012	121,747,031	http://mir.dcs.gla.ac.uk/resources/ [20]
4	Event Description - Event 2012	506	http://mir.dcs.gla.ac.uk/resources/ [20]
5	Event Categories - Event 2012	506	http://mir.dcs.gla.ac.uk/resources/ [20]

Table 3.1 Data sets we used in the research

For that reason, we turn our focus on the data-set that Mcminn et al. 2013 [20] created. They built a large corpus of tweets, starting on the 10th of October 2012 and ending on the 7th of November, for evaluating event detection on Twitter. They collected 121,747,031 tweets, with a large scale of noise tweets (e.g., spam tweets, ads, chi-chat tweets) and only 152,951 of them being labeled into 506 events. Moreover, the events are also categorized into eight general topics: Armed Conflicts Attacks, Arts, Culture Entertainment, Business Economy, Disasters Accidents, Law,

Politics Scandals, Miscellaneous, Science Technology, and Sports. (Figure 3.2) That

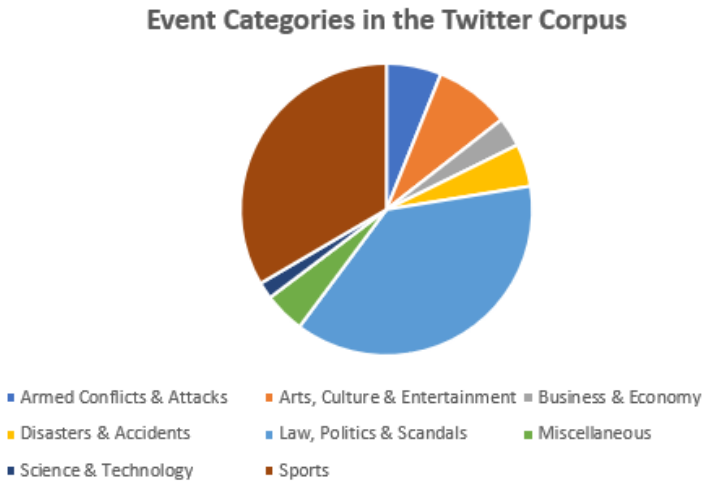
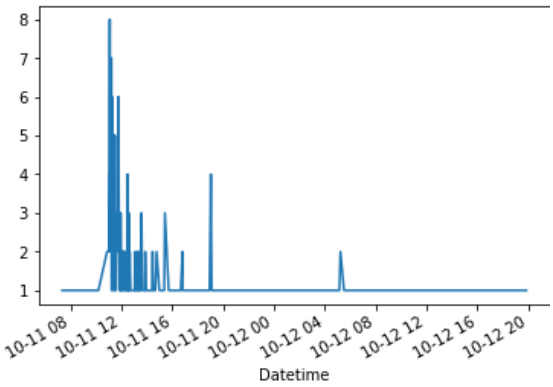
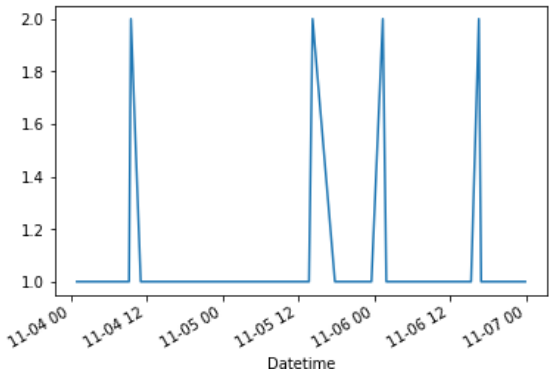


Figure 3.2 The percentage of tweets for each category in our dataset

twitter corpora was the most representative data set that we were able to utilize. [20] made available their data-set in public with only their tweet_id. In order to make use of these tweets, we used the Hydrator Framework and extracted them as JSON. Hydrator is an Electron-based desktop application for hydrating Twitter ID data-sets, which helps you turn tweet_ids back into JSON. Even though only 52% of tweet_ids, we were able to download because the other tweets were being deleted. To make fair evaluations in our research, we get rid of specific events from our data-set because they were harmed from the issues that we faced, such as deleting events that consist of 15 tweets at maximum or events that do not witness any anomaly or burstiness. In this research, we collect 501,657 tweets from the corpus, containing only 97,754 relevant tweets and 403,903 irrelevant tweets.



(a) Track of a non-crushed event which was included to dataset



(b) Track of a crushed event which was eliminated from dataset

Figure 3.3 Trend of a crushed and non-crushed events from dataset

3.3 Feature Extraction

Feature Extraction is the most challenging part of this research due to the variety of features that we able to derive from the literature that we found. Hence, we desired to manage almost all the characteristics that a tweet consists such as text, hashtags, mentions, RTs, and URLs. Unfortunately, due to the unavailability of libraries to recognize name-entities efficiently from tweets, we directed on extract features like n-grams, tweet features, and TF-IDF vectorization.

3.3.1 Data Preprocessing

Before extracting and processing features, it is essential to make data preprocessing steps for each timestamp when we collect our data in order to make sense of them. These steps will help our system utilize the incoming data more efficiently by using cleaning_data for the NLP methods. First things first, we get rid of non-English tweets for being able to use NLP functions from the NLTK library (Figure 3.4). After language filtering, for each tweet, we preprocess the text as follows. We normalize the text by separating it from URLs, user mentions, hashtags, retweet mentions and also removing digits and other punctuations(e.g. !"$%&(́)*+,-). Afterward, we tokenize the remaining clean text by white space and also remove the stoping_words.

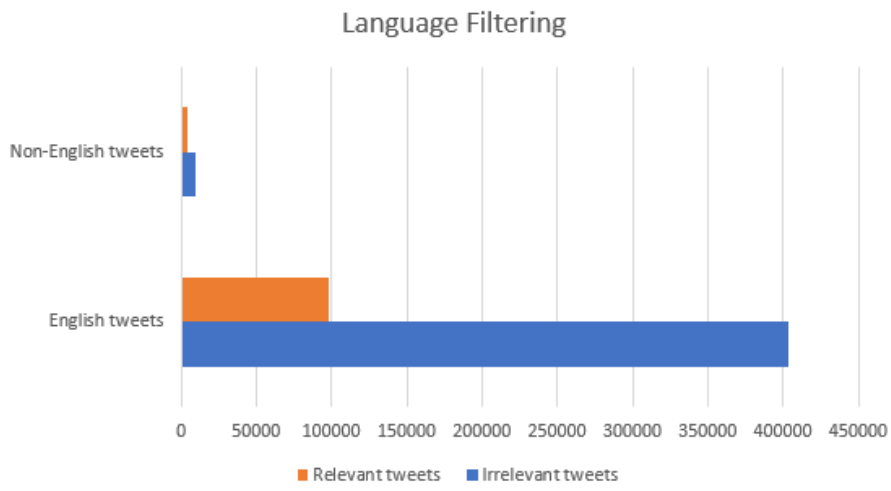


Figure 3.4

The next stage for our data preprocessing is stemming each token from the clean text, which will help us convert every word to its root. In the end, in order to prepare the data set for the TF-IDF vectorization, we append back to the clean text the tweet_features that we separated before. Figure 3.5 shows an example of how a tweet is prepossessed and prepared for the Information Retrieval function.

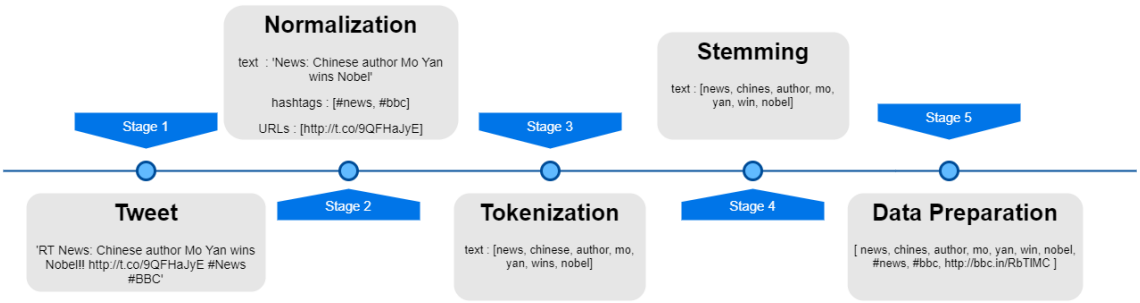


Figure 3.5 The stages for data preprocessing

3.3.2 Information Retrieval

The next step in our implementation after the data cleaning is the vectorization of data. In other words, the system converts each tweet into a vector with one dimension per word and fill the dimension with the word count.

3.3.2.1 Bag-of-words construction

In our case, we implement the idea of utilizing tweet segments and, more specific the combination of uni-grams, bi-grams, and tri-grams instead of only uni-gram tokens. The n-gram model that we used is helpful for the detection of frequent key-phrases among a large number of event-related messages, and it is much better than trying to find them with one-word relations. Thus, our system creates a bag-of-words vocabulary for each time_window, consists of 1-3gram phrases. We then normalize each vector in our vocabulary by document word frequency (TF-IDF). The reason for this action is due to machine learning algorithms that cannot work with the raw text directly, so we should convert our BOW vocabulary into vectors of numbers.

3.3.2.2 TF-IDF

TF-IDF, known as the Frequency-Inverse Document Frequency model, is one of the most common ways of computing the weight of a term in the BOW vocabulary. In other words, it tries to estimate the importance of each term regarded to the whole collection and its document. TF is concerned as the number of times a term appears in a document divided by the total number of terms in the document. Every document has a term frequency. On the other hand, the IDF component estimates the weight of the term across all documents in the corpus. It is computed by the log of the number of documents divided by the number of documents that contain the specific term (Equation 3.1). Lastly, the TF-IDF is measured by the multiplication between TF and IDF. The formula is shown in Equation 3.2.

$$idf(t, D) = log \frac{\mid D \mid}{1 + \mid \{d \in D : t \in d\} \mid} \tag{3.1}$$

$$tfidf(t, d, D) = tf(t, d) \times idf(t, D) \tag{3.2}$$

For our research, the collection is the whole data-set of tweets presented on a specific time_window i, which in our case has 10-minute length. The document is every tweet m contained in the data-set, and the term is every word, which in our case is key_phrase, included in the BOW vocabulary.

3.4 Event Clustering

Event Clustering is the part of the methodology where we are bound to use the features we extracted to create clusters. These clusters will consider as candidate events since tweets that contain on each cluster consist of common elements. The first thing we do, after feature extraction, is to get rid tweets from the Vector Space Model that do not contain any term from BOW vocabulary. This step helps us removing out-of-vocabulary tweets that are not meaningfully grouped into a cluster.

After eliminating non-relevant tweets and modified the vectorized features, we needed to find the most appropriate clustering method for our system. Because in our research, we focus on identifying events without any pre-trained data, we followed unsupervised clustering methods. Getting tweets from the real-time procedure, without knowing the exact number of tweets and events, we wanted to emphasize finding a clustering approach that is advantageous dealing with outnumbered noise-data and manage to extract clusters consisted of relevant tweets with a common topic. Under an in-depth investigation, we decided on our model to utilize the DBSCAN algorithm, known as the Density-based spatial clustering of applications with noise [31].

The DBSCAN algorithm relies on a density-based concept of clusters, which is based on creating clusters with arbitrary shapes without also setting the number of groups.

According to [8], it is also efficient on large spatial data-sets. The main idea of the DBSCAN algorithm is to view clusters as regions of high density that are separated by areas of low density. Therefore, two parameters required to measure the frequency of a region:

- The number of points within a circle of Radius Eps (ϵ) from point P.
- The minimum number of points (MinPts), the circle with radius ϵ, at least contains.

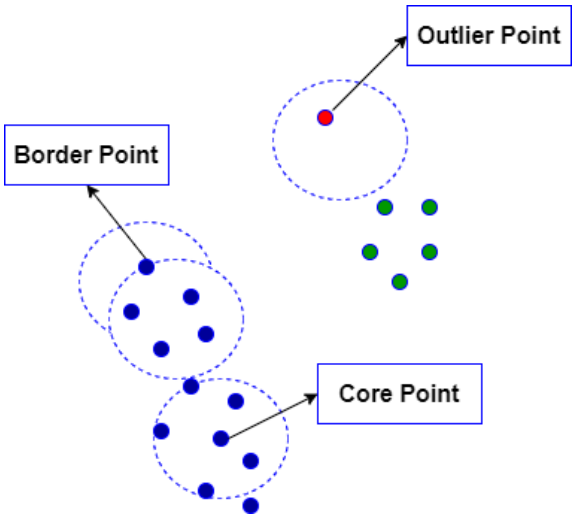


Figure 3.6 Core and Border Points in a Database D. Red data point is Noise.

Moreover, DBSCAN algorithm divides its objects into three exclusive groups:

1. **Core Points**: Points that are at the interior of a cluster.
2. **Border Points**: Points that have fewer than *MinPts* within Eps, but is in the neighborhood of a core point.
3. **Outlier Points**: Any point that is not a core point nor a border point.

Hence, the DBSCAN algorithm will help us not only group relevant topics in an arbitrary shape but also detect and get rid of outliers in the data-set.

$$\cos(\mathbf{A},\mathbf{B}) = \frac{\mathbf{AB}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n}\mathbf{A}_i\mathbf{B}_i}{\sqrt{\sum_{i=1}^{n}(\mathbf{A}_i)^2}\sqrt{\sum_{i=1}^{n}(\mathbf{B}_i)^2}} \tag{3.3}$$

Another parameter in the DBSCAN algorithm that we considered was the type of metric to use when calculating the distance between points in a feature array. We decided to measure the distances with the cosine similarity measure. Because our features were extracted as binaries and are set as vectors, we followed a nonEuclidean distance to detect similar topics more efficiently. Mathematically, Cosine s used to measure the cosine of the angle between two vectors projected in a multi-dimensional space (shown in Equation 3.3). As shown below in Figure 3.7, the smaller that angle, the more similar are the two vectors, which are arrays containing the term counts (in BOW vocabulary) of two documents (tweets).

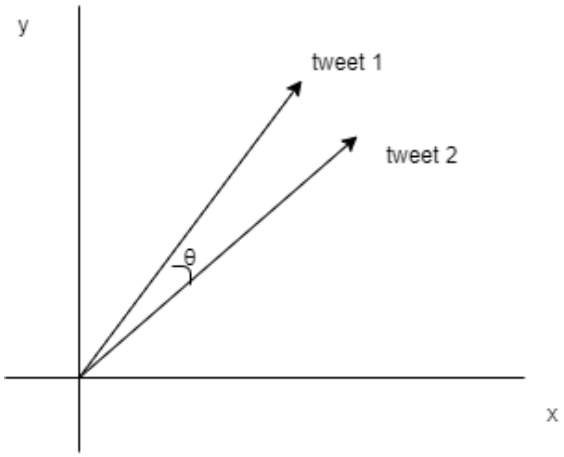


Figure 3.7 Vector representation of tweets.The more θ is small,the more similar are the tweets.

Besides that, we managed to cluster our features with other clustering methods like Hierarchical clustering [12], K-means clustering, HDBSCAN clustering (OPTICS), and Affinity propagating clustering, in order to compare our result with these and draw some conclusion. In any case, from the problem's theory, we decided to follow the DBSCAN approach due to the reasons we said above.

3.5 Event Summarization

The final step in our implementation is to conclude the details about clusters that have created. We need to determine the most representative tweet for the cluster, which will present accurate and useful information. At first, we concerned the idea of recognizing the name entities for each cluster and show the tweet with the highest score of entities. Hence, we experimented with the Stanford NLP and the SpaCy libraries. However, we found that in the majority of cases, these methods failed to recognize entities due to the specific language of words like arbitrary capitalization and short names, or the sentence structure of tweets. Therefore, we decided to simplify our process and summarize each cluster related to the most common words in the text. At this time, we desire to extract a useful and straightforward tweet for every group, which will also be readable for the viewer in the outcome. Moreover, except for the title of the cluster, we can extract some other pieces of information for the cluster, like the most common hashtags, mentions, and URLs alike. These types of information will help us in further investigations about Event Detection, like tracking a specific event that occurred in real-time.

To find most content words for a cluster, we need to do some critical steps. Repeatedly, we need to clean every tweet as we said in chapter 3.3.1; tokenizing, removing stop-words, and stemming. However, at this time, we are bound to use only the stemming words from the text of each tweet and discard tweet_features. After cleaning our data, we need to find the tweet with the highest density of information. Therefore, we score words using a metric known as CountVectorizer from the sklearn library. CountVectorizer is a function that converts a collection of text documents to a matrix of token counts. Moreover, we defined our features from CountVectorizer extracted as binaries. Thus, the Vector Space Model separated as non-zero counts, which are set as 1, and zero counts which are set as 0. This is helpful for discrete probabilistic models that model binary events rather than integer counts.

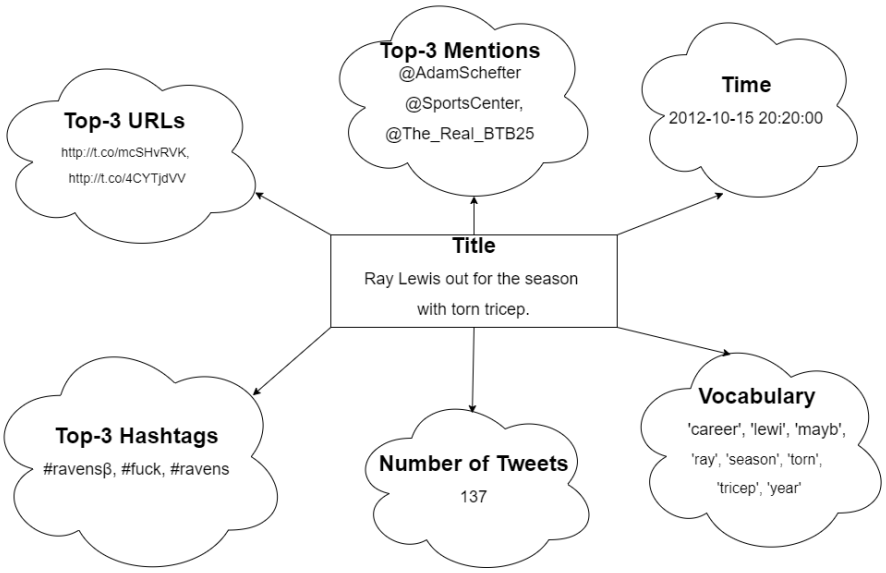


Figure 3.8 Example of how our model detects an Event and concludes the most representative tweet along with other useful information such as the Time the Event detected, the Top-3 Hashtags, the Top-3 User Mentions, the Top-3 URLs, the number of tweets where clustered and the BOW vocabulary.

After extracting our features as binaries, we are ready to find the particular tweet that maximizes the density of content words. Figure 3.8 shows an example of how our model summarized an event detected and what pieces of information extracts.

Chapter Four

Evaluation

Contents

4.1 Evaluation Metrics

There are various ways to measure the performance of a cluster, analyze similarity and the difference from it. We decided to utilize six metrics that provide us a complete view for each clustering approach that we followed; Purity, AMI, Homogeneity, Completeness, Fowlkes-Mallows scores and the percentage of Events Detected.

Purity is an external evaluation criterion for cluster quality and is a measure of the extent to which clusters contain a single class. It requires labeled data to assess a clustering concerning ground truth. Therefore, to calculate purity, each cluster is assigned to the label where it is the most frequent. Consequently, the sum for every cluster is taken and divided by the total number of data points which in our case

are the tweets.

$$Purity = \frac{1}{N}\sum_{i=1}^{k} max_j|c_i \cap t_j| \tag{4.1}$$

Equation 4.1 shows the calculation of the Purity, where N = The total number of tweets(data points), k = number of clusters, c_i is a generated cluster, and t_j is the label which has the maximum number for cluster c_i.

In **AMI (Adjusted Mutual Information)**, we need to given the knowledge of the ground truth assignments, which in our case is the event that each relevant tweet belongs to, and our clustering algorithm assignments of the same samples. Therefore, the AMI score assesses the agreement of the two assignments, ignoring permutations [28]. The AMI score is close to 1 when the labels included in the two clusters are similar (when the score is 1, then the two clusters are identical). In contrast, if the score is close to zero, then the two clusters are expected to be more independent with each other.

$$\text{AMI} = \frac{\text{MI} - E[\text{MI}]}{\text{mean}(H(U), H(V)) - E[\text{MI}]} \tag{4.2}$$

Equation 4.2 shows the calculation of the AMI score, where MI(Equation B.5) is the mutual information score between two label assignments U and V, H(U)(Equation B.2) and H(V)(Equation B.4) is the entropy of each assignment and E[MI] is the calculation of expected value for the mutual information.

Homogeneity score evaluates whether each cluster contains only members of a single class on each time window that we generate clusters (Equation 4.3). On the other hand, **Completeness** score evaluates whether all members of a single class are assigned to the same cluster on each time window (Equation 4.4).

$$\text{Homogeneity} = 1 - \frac{H(C|K)}{H(C)} \tag{4.3}$$

$$\text{Completeness} = 1 - \frac{H(K|C)}{H(K)} \tag{4.4}$$

Fowlkes-Mallows (FMI) score measures the similarity of two clusters of a set of points. The higher the value of the Fowlkes–Mallows score, the more similar the clusters and the benchmark classifications are. Before calculating this measure, we need to clarify some essential definitions.

1. **True Positive (TP):** The number of pairs of points that belong to the same clusters in both the true labels and the predicted labels [10]. In our example, this is the total number of correctly predicting tweets in a particular event.

2. **False Positive (FP):** The number of pairs of points that belong to the same clusters in the true labels and not in the predicted labels [10]. In our example, the total number of tweets that clustered in a different cluster while they are relevant to each other.

3. **False Negative (FN):** The number of pairs of points that belong to the same clusters in the predicted labels and not in the true labels [10]. In our example, the total number of tweets that clustered and are irrelevant to each other.

$$\text{FMI} = \frac{\text{TP}}{\sqrt{(\text{TP} + \text{FP})(\text{TP} + \text{FN})}} \tag{4.5}$$

Explaining these metrics leads to the explanation of the formula that calculates the Fowlkes-Mallows score in Equation 4.5. The score ranges from 0 to 1. A high value points out a functional similarity between two clusters. More specifically, values that are close to zero demonstrate two largely independent label assignments, while values close to one indicate significant agreement. Furthermore, when values are precisely 0 means that the two label assignments are purely independent, and when

the score of FMI is 1, it means that the assignments are equal.

Finally, we measured the percentage of events that our system detected overall, and we presented it as **Events Clustered**. It is essential to mention that this metric does not indicate any performance of any clustering method because it does not take into account the creation of clusters that contain irrelevant tweets. Therefore, the results of this ratio help us to be aware only about the number of real events that were detected within our system.

We used these metrics for evaluating our model with different clustering approaches. In other words, we utilized a combination of parameters and features in order to assessed which is the best combination. Therefore, we executed our model with each combination of parameters, and we calculate these metrics to draw a variety of conclusions.

4.2 Comparisons

4.2.1 Experiment Setup

Comparing different types of parameters in our clustering approach is one of our main contributions, which will help us adjust the appropriate inputs in our system. Therefore, our first experiment was to analyze each combination of parameters for the clustering algorithm that we decided to use, known as the DBSCAN algorithm. Setting the appropriate n-gram structure for our creation of Bag-Of-Words vocabulary for each time window and also the value of epsilon, which is the most critical DBSCAN parameter in order to generate clusters, where the two parts that we needed to consider first. Consequently, we compare our results from the clustering metric that we made with other clustering methods, such as HDBSCAN, Hierarchical, Kmeans, Affinity, and made the most important observations. Furthermore, we have to mention that every experiment takes place at my personal computer, which

is composed of 2-cores and 4-threads CPU clocked in 2.3 GHz, 10 Gb RAM, and Windows 10 Operating system. From the software implementation, we used Jupyter Notebook, which helps us to write Python programming language. Moreover, we used the sklearn library, which contains all the clustering methods that we utilized and also the metrics for measuring and evaluating each combination.

4.2.2 Results

This section presents the results of our study and provides four summary tables, one for every n-gram analysis that we made. These tables contain the AMI, Homogeneity, Completeness, FMI, Purity score, and the ratio of detected events for each combination of parameters we tried. Moreover, observing the results on each table, we managed to draw some important conclusions and considerations about our clustering approach. Furthermore, we create a visual comparison of each combination model to give a better understanding of the appropriate feature's impact on the outcome.

Epsilon	**AMI**	**Homog.**	**Compl.**	**FMI**	**Ev.Clustered**	**Purity**
0.4	0.783	0.887	0.862	0.936	0.780	0.980
0.5	0.752	0.858	0.852	0.926	0.825	0.975
0.6	0.710	0.800	0.845	0.915	0.884	0.964
0.7	0.652	0.689	0.895	0.907	0.924	0.922
0.8	0.658	0.736	0.876	0.896	0.963	0.858
0.9	0.459	0.633	0.708	0.825	0.915	0.847

Table 4.1 Comparison of metrics for 1-gram DBSCAN Clustering

Epsilon	**AMI**	**Homog.**	**Compl.**	**FMI**	**Ev.Clustered**	**Purity**
0.4	0.815	0.919	0.882	0.946	0.671	0.982

0.5	0.803	0.904	0.880	0.943	0.747	0.981
0.6	0.788	0.889	0.870	0.940	0.808	0.981
0.7	0.749	0.839	0.862	0.928	0.867	0.975
0.8	0.660	0.709	0.887	0.903	0.918	0.942
0.9	0.540	0.638	0.837	0.840	0.971	0.843

Table 4.2 Comparison of metrics for 1-2gram DBSCAN Clustering

Epsilon	**AMI**	**Homog.**	**Compl.**	**FMI**	**Ev.Clustered**	**Purity**
0.4	0.818	0.921	0.885	0.947	0.637	0.982
0.5	0.809	0.920375865	0.876	0.942	0.699	0.983
0.6	0.796	0.907	0.869	0.942	0.766	0.983
0.7	0.767	0.880	0.858	0.933	0.820	0.981
0.8	0.721	0.807	0.863	0.919	0.882	0.969
0.9	0.541	0.580	0.900	0.850	0.963	0.879

Table 4.3 Comparison of metrics for 1-3grams DBSCAN Clustering

Epsilon	**AMI**	**Homog.**	**Compl.**	**FMI**	**Ev.Clustered**	**Purity**
0.4	0.827	0.934	0.884	0.947	0.609	0.983
0.5	0.807	0.925	0.868	0.939	0.679	0.984
0.6	0.793	0.914	0.860	0.939	0.721	0.984
0.7	0.767	0.899	0.840	0.929	0.778	0.983
0.8	0.737	0.862	0.833	0.922	0.823	0.980
0.9	0.591	0.645	0.893	0.866	0.918	0.931

Table 4.4 Comparison of metrics for 2-3grams DBSCAN Clustering

Overall, what stands out from these tables is that there are general and specific conclusions derived from them. Let's start with general observations. Initially, AMI reaches a higher score when the epsilon value is lower. That means each cluster of tweets contains the highest similarity compared to their labeled data. Therefore, Homogeneity and Purity score reach higher scores too. FMI also has a higher rating when the epsilon value is lower, which means that our system generates pure clusters. On the other hand, creating purer and more valid groups does not mean that our model detects a higher number of events. When the epsilon value is lower, the ratio of events that are detected and clustered is low. By setting more upper epsilon, the detection of events getting a higher percentage, while our system also generates irrelevant clusters. With these observations, we decided to find the appropriate epsilon value and n-gram parameters in order to have a high ratio of detected events and also creating clusters that have a high amount of purity and even high FMI scores. Processing uni-grams, bi-grams, and tri-grams phrases and setting the epsilon value at 0.8, was the optimal decision for our requirements and future purposes. This combination of parameters reaches FMI and Purity score at 0.919 and 0.969, respectively, and finds the 88% of the Events. Again, the selection of parameters depends on our research purposes. At this time, we concerned about creating clusters with high purity scores and also detect a good percentage of Events.

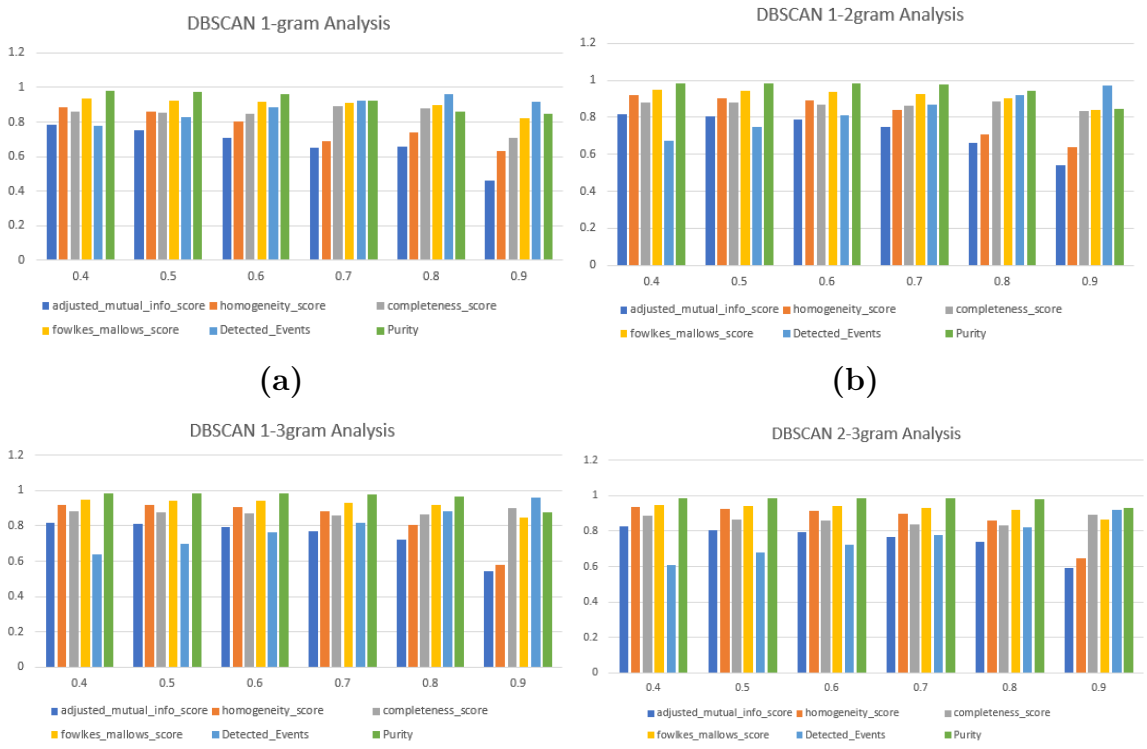


Figure 4.1 Visual Comparison for each n-gram analysis

Besides that, we compare our optimal method with other clustering approaches, like Affinity, HDBSCAN, Kmeans, and Hierarchical clustering. We observed that the DBSCAN method is better for real-time clustering than the other methods. Having the ability to create clusters with arbitrary shape and also get rid of noise data, is one of the reasons that reach higher scores related to the other clustering methods. Table 4.5 presents the score for each metric that we measure by utilizing the 1-3gram analysis for BOG vocabulary creation for feature extraction.

Method	AMI	Homog.	Compl.	FMI	Ev.Clustered	Purity
Affinity	0.359	0.875	0.472	0.313	0.985	0.961
Hdbscan	0.153	0.423	0.252	0.671	0.654	0.920
Kmeans	0.324	0.933	0.420	0.349	0.997	0.946
Hierachical	0.519	0.782	0.611	0.523	0.994	0.953
Dbscan	0.721	0.807	0.863	0.919	0.882	0.969

Table 4.5 Comparison of metrics for 1-3grams Clustering

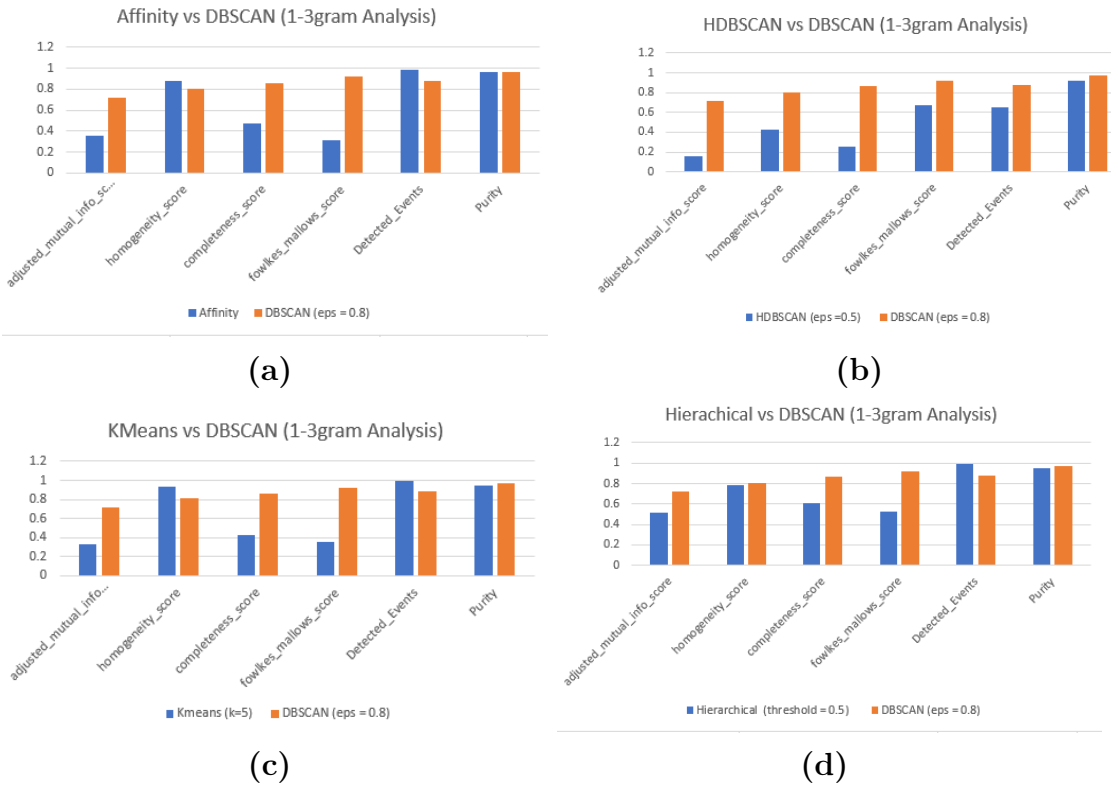


Figure 4.2 DBSCAN vs Other Clustering methods (1-3gram Analysis)

4.3 Data Analysis with the Optimal Method

As we mention in the previous section, we needed to decide what combination is the appropriate one in order to make further investigations, particularly in the clusters that our model generated. In our study, we decided that the combination of 1-3gram feature extraction, eps = 0.8, and the clustering algorithm of DBSCAN, is the perfect one by detecting 83% of events and also reach 92% FMI score, which means that we generate pure clusters with low noise. Therefore, in subsection 4.3.1, we decided to analyze a specific event by tracking its clusters and also in paragraph 4.3.2 analyze the irrelevant groups which were created and draw some important conclusions.

4.3.1 Event Tracking

For this experiment, we took a specific event from the Twitter corpora, with its description. As you can see in Figure 4.4, except for the title of the event, it indicates the time of the first and last posted tweets of the specific event. Most importantly, it shows the peak times were Twitter users post tweets in the related topic. In other words, in peak times, our model needs to identify that there is a burst of tweets and generate clusters associated with this topic. Moreover, in Figure 4.3, it shows the frequency of tweets for the specific event that we track, and also the points of when the event hit a peak on tweets.

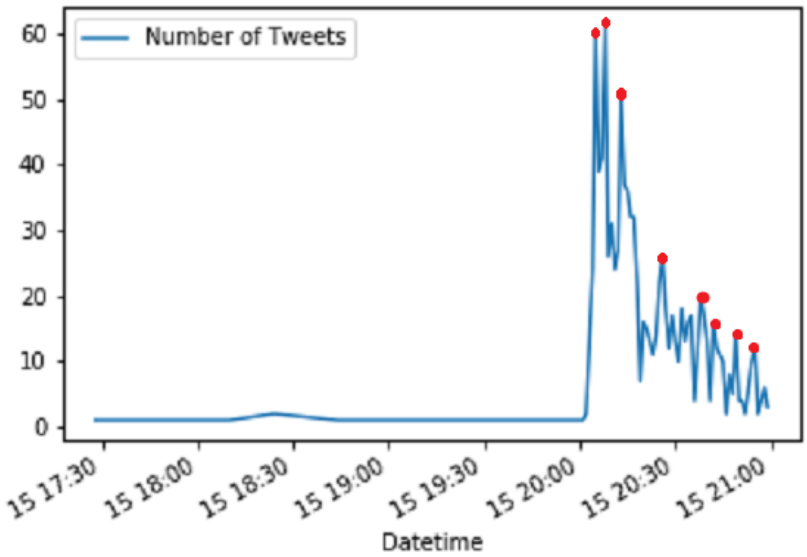


Figure 4.3 The frequency of tweets for a particular event in the corpus

Event Title	Ray Lewis tore a tricep muscle out for possibly a year and maybe career.
Start Time	10/15/2012 17:28
End Time	10/15/2012 20:59
Peak Times	20:05, 20:08, 20:13, 20:26, 20:32, 20:38, 20:42, 20:49

Figure 4.4 The Description of the tracking event in Figure 4.1

Table 4.6shows the summarized clusters for the current event, which contains a representative title, keywords, the number of tweets that were clustered, and the time where the cluster was generated. As you can see in the table, the first summarized cluster was detected at 20:08, 3 minutes after the first peek of the event.

That means our model can detect an event that occurred at a proper time. In the following clusters, we observed that each title related to the same topic, with similar keywords. That means, our topic detection was validated by the continuous mention on the same topic.

id	Time	**Topic Headline**	Topic Keywords	Length
778	2012-10-15 20:08:00	Ray Lewis out for the year... #heartbroken	'done', 'lewi', 'ray', 'year'	21
779	2012-10-15 20:12:00	Ray lewis out for the year	'career', 'lewi', 'mayb', 'ray', 'season', 'torn', 'tricep', 'year'	187
780	2012-10-15 20:16:00	@AdamSchefter: Ray Lewis torn tricep - out for year and maybe career. @tgillionaire	'career', 'lewi', 'mayb', 'ray', 'season', 'torn', 'tricep', 'year'	98
781	2012-10-15 20:20:00	Ray Lewis out for the season with torn tricep	'career', 'lewi', 'mayb', 'ray', 'season', 'torn', 'tricep', 'year'	144
782	2012-10-15 20:24:00	Ray Lewis out for season with torn triceps #timetoretire	'career', 'lewi', 'ray', 'season', 'torn', 'tricep'	62
783	2012-10-15 20:28:00	So Ray Lewis is out for the season with a torn tricep	'career', 'lewi', 'ray', 'season', 'torn', 'tricep', 'webb', 'year'	24
...	...	...	...	...

794	2012-10-15 21:01:00	Ravens lose Ray Lewis for season with torn triceps http://t.co/iSi43gZe	'career', 'lewi','lose', 'make','raven', 'ray', 'season', 'torn', 'tricep', 'year'	7
796	2012-10-15 21:05:00	@SportsCenter: BREAKING: RT @AdamSchefter Ray Lewis torn tricep - out for year and maybe career.	'break','career', 'lewi','news', 'ray', 'sad','torn', 'tricep','year'	6

Table 4.6 Event Tracking - Topic Headline

Another interesting part of the tracking of a specific event is the extraction of standard social features, like hashtags, mentions, and URLs. As you can see from the Table 4.7, all summarized clusters shared similar characteristics, which means that every event can be tracked and validated by them.

id	Top_Hashtags	Top_Mentions	Top_URLs
778	['#heartbroken', '#fuck']	-	-
779	['#ravens', '#nfl', '#wamp']	['@AdamSchefter', '@SportsCenter', '@Ravens']	-
780	['#ravens', '#whatsgoingon', '#footballfantasy']	['@AdamSchefter', '@SportsCenter', '@tgillionaire']	-
781	['#ravens', '#fuck', '#gotdamn']	['@AdamSchefter', '@SportsCenter', '@The$_{R}eal_{B}TB25'$]	['http://t.co/mcSHvRVK', 'http://t.co/4CYTjdVV', 'http://t.co/ttun0Efy']

782	['#smh', '#timetoretire', '#sadtweet']	['@AdamSchefter', '@SportsCenter', '@christianertel']	-
783	['#depressing', '#fuckthis', '#ravennation']	['@AdamSchefter', '@SportsCenter', '@DonJulio822']	-
...	...	...	...
794	['#outforseason']	['@AdamSchefter', '@theScore']	['http://t.co/iSi43gZe', 'http://t.co/sLfIdqzZ']
795	-	-	-
796	-	['@AdamSchefter', '@SportsCenter']	-

Table 4.7 Event Tracking - Social Features

4.3.2 Irrelevant Clusters Analysis

Taking irrelevant clusters under consideration is also another vital part that we needed to analyze, mainly for future studies that include Noise filtering or Cluster Classification. Figure 4.5 demonstrates the number of clusters generated for each length of tweets. Overall, the majority of irrelevant clusters have as length six and below, which means that unrelated clusters contain a small number of related tweets. Afterward, we analyzed the frequency of words for these irrelevant clusters and created a graph that includes the 50-most frequent words (shown in Figure 4.6). In this figure, we observed that many clusters contained hate comments about popular names like Obama or Romney, who related to politics. Furthermore, a group of advertisements and spam messages also clustered into our model. Finally, a remarkable group of tweets that contain daily conversations also directed into clusters in our model.

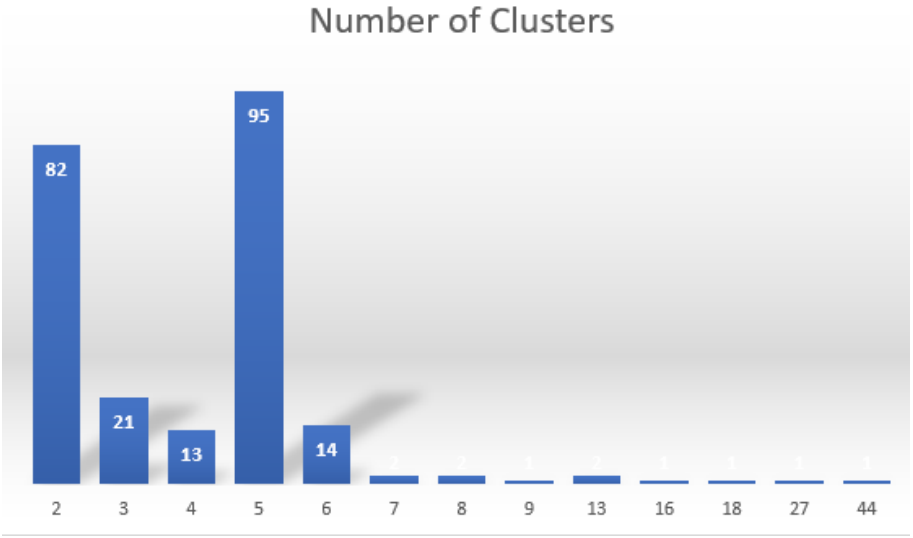


Figure 4.5 The number of clusters generated for each length of tweets

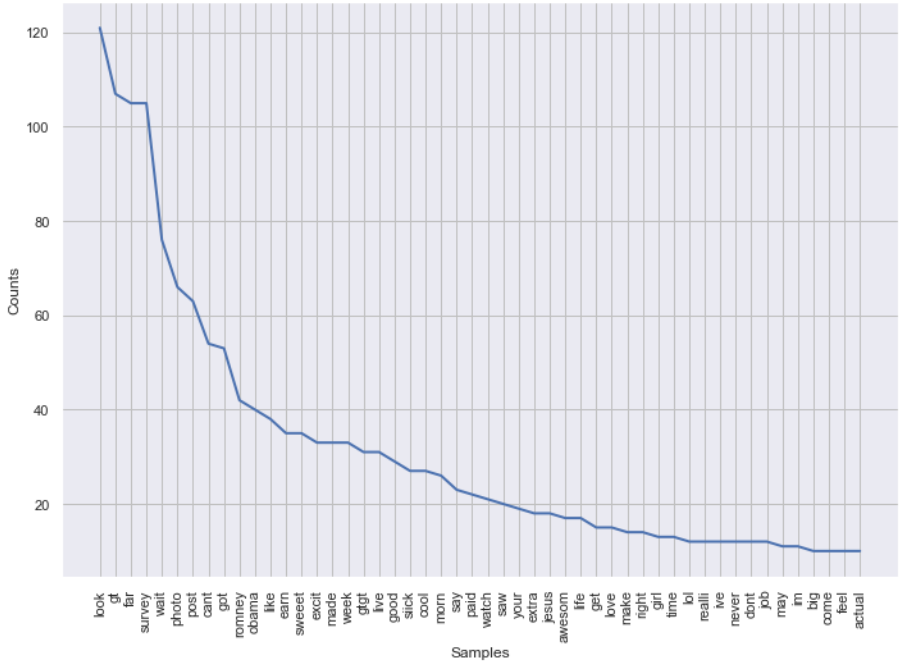


Figure 4.6 Word-frequency from irrelevant clusters

4.4 Lessons learned

Through our investigation about the Event Detection techniques combined with the evaluation metrics that we used, we summarize into fundamental knowledge. The lessons learned that we received from this work through different viewpoints, will help us dig into more research about this work with much more success.

First of all, we managed to utilize the most appropriate clustering method regarding our formulation of the problem. We wanted to get rid of irrelevant data to the fullest extent for each time window, and also to generate relevant clusters in an arbitrary shape. Therefore, the determination of the DBSCAN algorithm for our system was the most successful choice compared to the other clustering algorithms. Furthermore, the selection of cosine similarity as a metric in the DBSCAN algorithm to estimate the distance between tweets in a feature array was crucial for the succeed clustering. Because each tweet converted into feature vectors, the perfect comparison for these modified features was cosine similarity measurement.

Regarding the evaluation metrics that we have taken, we achieved to learn how the system works into a different combination of parameters. By using a combination that will generate clusters that provide high purity of relevant tweets and a low amount of noisy tweets, will also decline considerably the percentage of events, which means that our system lacks complex features clustering. In contrast, when the system generates a little lower purity clusters, we observe that it also produces a high ratio of events and irrelevant clusters. Through evaluation metrics, we concluded that by extracting 1-3gram features and setting the Epsilon value to 0.8, our system gets a high score from any perspective, with 96,9% Purity and 88.2% Detection of Events.

Tracking a specific event from the data-set that detected from our system was

an experiment to draw some helpful conclusions, especially for the TDT part of the research. Firstly, we realized how important is the volume of tweets and their spikes into a time-window for the topic to be detected by the system. We also perceived that specific keywords, hashtags, and mentions that represent a cluster within a particular time-window had occurred continuously during the detection of events into different time-window and clusters. In my viewpoint, this observation will encourage us to measure the newsworthy and veracity score regarding a detected topic in further investigations.

Last but not least, we analyzed the noise clusters that our system generated, and we collected remarkable observations. In contrast with the relevant detected clusters, noise clusters are suffered from the stability to generate relevant clusters in time. In the majority of cases, the features of irrelevant clusters do not repeatedly occur in continuous time-windows. Another interesting observation is that most of the irrelevant clusters, provide hate-of-speech content regarding a popular event or person. With that in knowledge, we recognize that by eliminating clusters that contain hate-of-speech content, we successfully improve the outcome of our system detection. Finally, we perceive from our experiments that irrelevant clusters are generated with a low number of tweets. Thus, we can consider these kinds of clusters as outliers and eliminate them from our system.

Chapter Five

Conclusion

Contents

5.1 Conclusion

Taking all into consideration, the goal of this research is to generate clusters with a high quality of purity that also related to a relevant topic. We strongly believe that we achieve the object, and we offer some valuable information that will help researchers digging into more investigation on this very tough problem. The plethora of information that we retrieved from every tweet and also the amount of our data, gives us a state-to-the-art outcome, which can also represent the processing of real-time data. The experiments we covered, suggest that by using a combination of text and twitter features combined with n-gram analysis, will help us extract the most appropriate feature for high-quality clustering. Besides that, we figured out that the DBSCAN algorithm has a remarkable impact on our clustering approach, more specifically by recognizing and eliminating the noisy data from the creation of our clusters, and the best possible outcome. Consequently, we also managed to summarize each cluster generated by finding the most common words, hashtags, URLs, the number of tweets contained on each cluster, and mainly the most representative title. This stage of summarizing each cluster Conclusion that we created in our

model, helped us to draw some conclusions and compare the differences between relevant and irrelevant clusters. Finally, we claim that the decision of our clustering method DBSCAN, which we concluded to utilize in our system, is one of the most critical keys to overcoming this challenging issue of Event Detection in Micro-blogs, and especially overcome the issue of noise data.

5.2 Future Work

The investigation of Event Detection in Micro-blogs can be extended into a wide range of further works in the future. Our study focuses mainly on clustering methods that will group tweets related to a particular topic. However, there are also plenty of tasks that are essential for system's better performance. First of all, great future work in our study is the implementation of noise filtering before the feature extraction. In other words, the implementation of a system that will classify the tweets into spam messages, chit-chat dialogue, advertisements, and regular messages is a crucial part of our research. The evolution of our model by appending this function will result in more precise the outcome.

Moreover, interesting work in the future would be the analysis of location from each topic that was detected by our system. An investigation on where precisely each topic was detected is another tough but also an attractive part of our case. For instance, our system would be able to define which topics are local events and which are global. This goal would achieve by digging further into the social features provided by tweets. Unfortunately, only 2% of tweets globally contain the geographical coordinates of where they posted, although there are other interesting features from tweets that are also considered for this part. Extracting data from shared URLs and getting the location of each user who posted the tweet are examples of features that would be interested in this work.

Besides that, great future work will be the calculation of the Newsworthiness

Conclusion score for each candidate event that is clustered by our system. Newsworthiness, in our case, is depended on the content of each cluster. This part of the research can be implemented after the Event Summarization, where our model consumes the most common words of tweets, the most common hashtags, and also the most representative title for the cluster. Using POS libraries and Named Entity Recognition tools maybe acted as a trigger to tackle this future part of work.

Bibliography

[1] Hamed Abdelhaq, Christian Sengstock, and Michael Gertz. "Eventweet: Online localized event detection from twitter". In: *Proceedings of the VLDB Endowment* 6.12 (2013), pp. 1326–1329.

[2] Charu C Aggarwal and Karthik Subbian. "Event detection in social streams". In: *Proceedings of the 2012 SIAM international conference on data mining*. SIAM. 2012, pp. 624–635.

[3] James Allan. "Introduction to topic detection and tracking". In: *Topic detection and tracking*. Springer, 2002, pp. 1–16.

[4] James Allan et al. "Topic detection and tracking pilot study final report". In: (1998).

[5] Mihael Ankerst et al. "OPTICS: ordering points to identify the clustering structure". In: *ACM Sigmod record* 28.2 (1999), pp. 49–60.

[6] Alexander Boettcher and Dongman Lee. "Eventradar: A real-time local event detection scheme using twitter stream". In: *2012 IEEE International Conference on Green Computing and Communications*. IEEE. 2012, pp. 358–367.

[7] Wenwen Dou et al. "Event detection in social media data". In: *IEEE VisWeek Workshop on Interactive Visual Text Analytics-Task Driven Analytics of Social Media Content*. 2012, pp. 971–980.

[8] Martin Ester et al. "A density-based algorithm for discovering clusters in large spatial databases with noise." In: *Kdd*. Vol. 96. 34. 1996, pp. 226–231.

[9] Bi Fang-Ming, Wang Wei-Kui, and Chen Long. "DBSCAN: Density-based spatial clustering of applications with nose [J]". In: *Journal of NanJing University (Natural Sciences)* 48.04 (2012).

[10] Edward B Fowlkes and Colin L Mallows. "A method for comparing two hierarchical clusterings". In: *Journal of the American statistical association* 78.383 (1983), pp. 553–569.

[11] Mahmud Hasan, Mehmet A. Orgun, and Rolf Schwitter. "A survey on real-time event detection from the Twitter data stream". In: *Journal of Information Science* 44.4 (2018), pp. 443–463. ISSN: 17416485. DOI: 10.1177/0165551517698564.

[12] Georgiana Ifrim, Bichen Shi, and Igor Brigadir. "Event detection in twitter using aggressive filtering and hierarchical tweet clustering". In: *Second Workshop on Social News on the Web (SNOW), Seoul, Korea, 8 April 2014*. ACM. 2014.

[13] Shenghang Jiang et al. "Robust nonparametric quantification of clustering density of molecules in single-molecule localization microscopy". In: *PloS one* 12.6 (2017).

[14] Chenliang Li, Aixin Sun, and Anwitaman Datta. "Twevent: segment-based event detection from tweets". In: *Proceedings of the 21st ACM international conference on Information and knowledge management*. 2012, pp. 155–164.

[15] Chenliang Li et al. "Twiner: named entity recognition in targeted twitter stream". In: *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval*. 2012, pp. 721–730.

[16] Xiaomo Liu et al. "Reuters tracer: A large scale system of detecting & verifying real-time news events from twitter". In: *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*. 2016, pp. 207–216.

[17] Xiaomo Liu et al. "Reuters tracer: A large scale system of detecting & verifying real-time news events from twitter". In: *International Conference on Information and Knowledge Management, Proceedings* 24-28-October-2016.October (2016), pp. 207–216. DOI: 10.1145/2983323.2983363.

[18] Leland McInnes, John Healy, and Steve Astels. "hdbscan: Hierarchical density based clustering". In: *Journal of Open Source Software* 2.11 (2017), p. 205.

[19] Andrew J McMinn and Joemon M Jose. "Real-time entity-based event detection for twitter". In: *International conference of the cross-language evaluation forum for european languages*. Springer. 2015, pp. 65–77.

[20] Andrew J McMinn, Yashar Moshfeghi, and Joemon M Jose. "Building a large-scale corpus for evaluating event detection on twitter". In: *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*. 2013, pp. 409–418.

[21] Polykarpos Meladianos et al. "Degeneracy-based real-time sub-event detection in twitter stream". In: *Ninth international AAAI conference on web and social media*. 2015.

[22] Ozer Ozdikis, Pinar Senkul, and Halit Oguztuzun. "Semantic expansion of tweet contents for enhanced event detection in twitter". In: *2012 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*. IEEE. 2012, pp. 20–24.

[23] Nikolaos Panagiotou, Ioannis Katakis, and Dimitrios Gunopulos. "Detecting events in online social networks: Definitions, trends and challenges". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Vol. 9580. 2016, pp. 42–84. ISBN: 9783319417059. DOI: 10.1007/978-3-319-41706-6_2.

[24] Antonia Saravanou et al. "Detection and Delineation of Events and Sub-Events in Social Networks." In: *ICDE*. 2018, pp. 1348–1351.

[25] Malcolm Slaney and Michael Casey. "Locality-sensitive hashing for finding nearest neighbors [lecture notes]". In: *IEEE Signal processing magazine* 25.2 (2008), pp. 128–131.

[26] George Valkanas and Dimitrios Gunopulos. "Event Detection from Social Media Data." In: *IEEE Data Eng. Bull.* 36.3 (2013), pp. 51–58.

[27] George Valkanas and Dimitrios Gunopulos. "How the live web feels about events". In: *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*. 2013, pp. 639–648.

[28] Nguyen Xuan Vinh, Julien Epps, and James Bailey. “Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance”. In: *The Journal of Machine Learning Research* 11 (2010), pp. 2837–2854.

[29] Lexing Xie, Hari Sundaram, and Murray Campbell. “Event mining in multimedia streams”. In: *Proceedings of the IEEE* 96.4 (2008), pp. 623–647.

[30] Siqi Zhao et al. “Human as real-time sensors of social and physical events: A case study of twitter and sports games”. In: *arXiv preprint arXiv:1106.4300* (2011).

[31] Yuzhen Zhao, Xiyu Liu, and Xiufeng Li. “An improved DBSCAN algorithm based on cell-like P systems with promoters and inhibitors”. In: *PloS one* 13.12 (2018).

APPENDICES

Appendix A

Abbreviations

AMI Adjusted Mutual Information

API Application programming interface

BBC British Broadcasting Corporation

BOG Bag of Words

CNN Cable News Network

CPU Central Processing Unit

DBSCAN Density-based spatial clustering of applications with noise [9]

FIFA Fédération Internationale de Football Association

FMI Fowlkes-Mallows Index

FN False Negative

FP False Positive

FSD First Story Detection

HDBSCAN Hierarchical Density-based spatial clustering of applications with noise [18]

IDF Inverse Document Frequency

JSON JavaScript Object Notation

LSH Locality-sensitive hashing

MI Mutual Information

NED New Event Detection

NER Named Entity Recognition

NFL National Football League

NLP Neuro-linguistic programming

NLTK Natural Language Toolkit

OPTICS Ordering Points To Identify the Clustering Structure [13]

POS Part-of-speech tagging

RAM Random-access memory

RT ReTweet

TDT Topic Detection and Tracking

TF Term Frequency

TP True Positive

URL Uniform Resource Locator

Appendix B

Equations

$$P(i) = |U_i|/N \tag{B.1}$$

$$H(U) = -\sum_{i=1}^{|U|} P(i)\log(P(i)) \tag{B.2}$$

$$P'(j) = |V_j|/N \tag{B.3}$$

$$H(V) = -\sum_{j=1}^{|V|} P'(j)\log(P'(j)) \tag{B.4}$$

Equations B.2 and B.4 are the calculation of the entropy of the two set of label assignments U and V. Equations B.1 and B.3 represent the calculation of the probability of U and V are picked at random respectively.

$$\mathrm{MI}(U,V) = \sum_{i=1}^{|U|}\sum_{j=1}^{|V|} P(i,j)\log\left(\frac{P(i,j)}{P(i)P'(j)}\right) \tag{B.5}$$

Equation B.5 is the calculation of the mutual information score between two label assignments U and V.

$$H(C|K) = -\sum_{c=1}^{|C|}\sum_{k=1}^{|K|} \frac{n_{c,k}}{n}\cdot\log\left(\frac{n_{c,k}}{n_k}\right) \tag{B.6}$$

Equation B.6 is the calculation of conditional entropy of the classes given the cluster assignments.

$$H(C) = -\sum_{c=1}^{|C|} \frac{n_c}{n}\cdot\log\left(\frac{n_c}{n}\right) \tag{B.7}$$

Equation B.7 is the calculation of the entropy of the classes.

Appendix C

Dictionary Features

adjusted_mutual_info_score

bi-gram: n-gram of size 2

chit-chat Small talk

cleaning_data: The data after the preprocessing.

completeness_score

content-content: A relation between two contents

dataset: A set of data

Detected_Events

entity-based: Based on entities

Eps: epsilon

fowlkes_mallows_score

graph-of-words: A graph that contains words

hashtag: Symbol "#" that occur in a tweet content

homogeneity_score

key_phrase keywords that create a phrase.

mention: Symbol "@" that occur in a tweet content

micro-blogs: Social Media platforms that allow users exchange content with each other.

MinPts: Minimum number of points.

multi-dimensional

n-gram: A sequence of n words from a given content of text.

non-zero Not equal to zero

out-of-the-box

outliers Noise data

preprocessing: An important step that includes data modifications in order to be processed at the next step.

pre-trained: Data that are trained before.

stoping_words: Commonly used words.

time_window: A predefined duration of time.

tweet_features: All entities that includes in a tweet(mentions, hashtags, RTs, URLs)

tri-gram: n-gram of size 3

tweet_id: The unique id of tweet.

uni-gram: n-gram of size 1

user-content: A relation between a user and content