Event detection, event characterisation and community detection on evolving networks

Submitted by Iraklis Moutidis to the University of Exeter as a thesis for the Degree of Doctor of Philosophy in Computer Science, May 2022.

(Signature) ....

Στον πατέρα μου Γρηγόριο Μουτίδη.

Θα ακουστεί αύριο.

Abstract

Since its inception the internet has revolutionized the way we communicate and interact with each other, how we acquire and share information as well as how we socialize and form communities. Social media platforms have significantly contributed to this revolution with billions of individuals using them on a daily basis all over the world. Anyone can ask a question, share personal information, opinions, media, news or content generated from other users and immediately reach any individual around the globe, forming an ever evolving network of interactions and relations. This makes clear that social and online media can be a valuable source of information and could help us reveal trends in the news and structures on them using techniques such as data mining and social network analysis. Also social and online media diffuse information instantly in contrast with traditional media that take much longer to make available any information, we can benefit from it by developing news detection methodologies that use streams of social or online media and monitor their evolution over time.

In this work the goal is to utilize social network analysis on evolving networks for discovering latent structures on online communities and conceive methodologies that extract useful information from online sources such as news articles, posts on social media and forums. This work has demonstrated what we can achieve by applying social network analysis on evolving networks.

In chapter 3 we were able to develop a novel event detection methodology that outperformed state of the art approaches and provided further insights such as a comprehensive summary of the event and the sentiment and emotions of users that were discussing about it. The method was applied in a number of heterogeneous data sets consisting of news articles and Twitter posts, for evaluating it and demonstrating how it works. We also used this methodology to detect the main events of the COVID-19 pandemic as it has been discussed on Twitter compiling a retrospective collection of viral events.

In chapter 4 by combining social network analysis and evolving networks we revealed communities of software developers on the Stack Overflow platform based on what technologies they ask and answer questions.

This work shows what we can achieve by applying social network analysis on evolving networks and provides two use cases that demonstrate this, pointing out that evolving networks should be one of the fields that social network analysts should focus on for further research in the future.

Acknowledgements

Firstly, I would like to express my sincere gratitude to my advisors Prof. Hywel Williams and Dr. Iain Weaver for the continuous support of my Ph.D study and related research, for their patience, motivation, and valuable knowledge. Their guidance helped me in all the time of research and writing of this thesis. I could not have imagined having better advisors for my Ph.D study.

I thank my fellow labmates from SEDA lab for the stimulating discussions, resource sharing, help and support in various challenges and for all the fun we have had in the last four years.

I thank the University of Exeter for funding this Ph.D and providing valuable resources and an excellent working environment. I also thank Adarga ltd. especially Rob Bassett Cross, Matt Williams, Alex Pavlides and Daniel Clarke for funding this Ph.D and for their guidance, support and brainstorming sessions during the duration of my degree.

Last but not the least, I would like to thank my family: my parents and sister for their valuable support throughout writing this thesis and my life in general.

Contents

List of Figures

List of Tables

Author's declaration

Two of the chapters in this thesis contain content that has been previously published and include work by my coauthors.

Chapter 3: Event detection and characterization on evolving networks

Chapter 3 contains works that has been published in the conference Complex Networks and their Applications VIII in 2019 [112] and was completed alongside Hywel T.P. Williams and in the journal Social Network Analysis and Mining, in 2020 [113] and was completed alongside Hywel T.P. Williams. For those papers I:

- Designed the pipeline of the system.
- Implemented the pipeline in python.
- Set up an evaluation methodology to test our system.
- Interpreted the results.
- Created visualisations of the main steps of the pipeline and of the generated results.
- Wrote the first draft of the manuscript, and edited it in response to comments from my coauthors and reviewers.

The part of that chapter that analyses tweets related with the COVID-19 pandemic is a work in progress and we were not able to publish it anywhere yet.

Chapter 4: Community evolution on Stack Overflow

Chapter 4 has been published in the journal PLOS ONE in 2021 [114] and was completed alongside Hywel T.P. Williams. An extended abstract of this work was also accepted on the IC2S2 conference in 2019. For this paper I:

- Designed the methodology for processing the data from Stack Overflow.
- Implemented the methodology in python.

- Interpreted and visualized the results.
- Wrote the first draft of the manuscript, and edited it in response to comments from my coauthors and reviewers.

The version printed in this thesis is as published in PLOS ONE and reproduced under the Creative Commons Attribution licence.

Chapter 1

Introduction

Over the last decades online mediums, communities and social media platforms have emerged and revolutionized the way we access information and socialize [7, 40, 120]. Every day a vast amount of information spreads around the world through the internet. Millions of people communicate and interact, news articles are published, opinions, questions and answers are posted on forums or social media platforms, creating a vast amount of data that can be processed to extract valuable knowledge[1]. In this work we aim to apply social network analysis techniques to heterogeneous sources of information streams to detect and extract events and insights from online communities and mediums.

Social network analysis characterizes the structural properties of the network and helps us detect significant entities or structures on the network, it also enables us to understand the significance and behavior of the network's entities. Such entities can be represented as nodes and relationships between different entities as edges on the network, however there are cases that those roles can be reversed. Social network analysis can be applied in a number of important fields both on industry and on research. Such fields are business intelligence, organizational behavior, human resources, event detection, inter organizational relations, the spread of contagious diseases, mental health, social support, the diffusion of information and animal social organization [5, 85, 171].

Numerous entities are associating with others forming various online social structures. Such associations can be explicit, for example, online interactions, friendship and mentions on online posts, or implicit, for example, co-appearing on the same article or sharing common characteristics. Those associations are forming networks that could reveal underlying information about the online community under investigation. A variety of techniques and metrics can be applied and reveal useful information about individual nodes or edges of the network as well as groups of them [32,68].

The main target of this work is to research what benefits we can acquire by applying social network analysis on evolving networks. We investigated what methodologies we can create in two

[1]https://ourworldindata.org/internet

main fields, event detection and insights acquisition on social media platforms. We were able to demonstrate that social network analysis on evolving networks is a promising undertaking that can produce valuable results.

In our work we apply social network analysis techniques to a series of dynamic networks created from various time windows. Those networks are named as dynamic networks and they have recently started capturing researchers interest [26, 38, 100, 144, 175]. Dynamic networks can be used to identify evolving communities and model the evolution of a system, where the behaviour of nodes (joining or leaving the network) can affect its community structure. In the rest of this work we also refer to dynamic networks as evolving networks which could be considered as a subset of dynamic networks that focuses on the network for a given duration of time [6, 52]. The rapid increase in usage of online social platforms results a vast amount of human-generated data having rich temporal information that can form dynamic networks. The way we represent dynamic networks can be split in two main categories, temporal networks and network snapshots. Temporal networks are graphs where on each vertex and edge a birth and death timestamps are assigned. This way every vertex and edge of the graph has a life duration. Network snapshots are a set of graphs composed from the vertices and edges that were present on the network during a specified time window. In our work we chose to use network snapshots since we could identify fixed periods of activity on each platform such as years, weeks, and days, and given the large volume of data we also decided to split the graphs into snapshots making the application of algorithms such as community detection faster.

This work has three main objectives. First to investigate whether we can utilize social network analysis to detect viral events on heterogeneous document streams and develop a methodology to automate this process. Every second an immense amount of information is being broadcast, creating the need for a mechanism that can cope with this amount of data and at the same time produce precise and reliable results. We discovered that tracking the popularity of named entities significantly benefits our methodology since most of the viral events are related with entities such as persons, locations or organizations. Second, how can we determine the context of the detected events? An increase in popularity of a named entity indicates that an event should have occurred, but we need more information on what the event is about. We found that alongside named entities, nouns and noun phrases are useful to describe the context of the event. By utilizing nouns, noun phrases and named entities we can generate a human comprehensible description of the event. Third, what insights can we obtain by applying social network analysis on online communities? We review the Stack Overflow question and answer website and study what network structures constitute implicit relations of users. After some experimentation we concluded that relations based on the common topic tags between two users create network structures where, after applying community detection algorithms, we can distinguish clusters of developers using specific programming languages or technologies. We were also able to monitor how those communities evolve over time and whether users moving between communities, stay in the same community, or completely abandon the platform.

In the following chapter, we will discuss about the background of our work and review the literature and outline the questions we seek to answer and their importance. In Chapter 3, we address the first of these questions by implementing and evaluating an event detection pipeline using social network analysis techniques. We then address the second question demonstrating our approaches for extracting the context of a detected event using text summarization, sentiment and emotion analysis. In Chapter 4, we use social network analysis on the evolving developers community of the Stack Overflow website. We present the most dominant communities on the platform and how they evolve. We finish by discussing the implications of these findings and possible directions of future research.

Chapter 2

Background and literature review

Applying social network analysis on evolving networks produces noteworthy knowledge when we analyse online and social media. Being able to employ those techniques for specific time windows generates more specific information, and enables us to utilize more approaches taking advantage of the time aspect of the data. In our work we use those approaches to implement an event detection methodology on news streams and extract a comprehensive summary for each detected event. We also apply social network analysis metrics and methodologies on evolving networks to detect and monitor communities of developers on the Stack Overflow platform.

2.1 Literature Review

There are many interesting examples of combining social network analysis and evolving networks in the literature, we will touch on some of them. Barabási et al. [16] worked on a co-authorship network of scientists from the fields of mathematics and neuroscience. They revealed the dynamic and the structural mechanisms that govern the evolution and topology of this complex system, they monitored the evolution of topological measures such as node degree and node separation, and they introduced a model that emulates the network's time evolution. Li, Li and Liu [93] apply social network analysis on evolving networks created from data collected by questionnaire surveys and interviews with 60 enterprises and assembling families and several government authorities from 2002 to 2008 in the rural area of the Nanzhuang Village, Yucheng County of Henan Province, China. Their focus is to detect which factors lead to creation and growth of manufacturing clusters in close, lagging and less developed rural areas. Wu et al. [175] utilize evolving networks to measure dependency among a set of international commodity futures prices from the the CRB index, introducing a new and straightforward way to investigate dependency variables. Their network analysis concluded that three commodities, crude oil, heating oil and copper, are found to play central roles in the commodity network, which means that price movements on those assets significantly affects the prices of other commodities. Lehnertz et al. [92] work on evolving networks that reflect the interaction dynamics between brain regions known in the literature as functional brain networks. They review methodologies for inferring networks from empirical time series and for a characterization of these evolving networks. Warren [28] uses relations between countries, such as alliances and treaties, to form a continuously evolving network which is observed in a series of discrete snapshots. His goal is to determine the form of the underlying utility function that affects alliance decisions. The results of his analysis conclude that the public nature of alliance treaties extends their effects far beyond those states that are part of an agreement.

2.1.1 Event detection and Tracking

The research area of topic/event detection and tracking (TDT) in news streams has a long history (e.g. [10, 172]). TDT typically combines natural language processing (e.g. named entity recognition, part of speech tagging, entity disambiguation), information retrieval (e.g. reversed indexing of document keywords, document similarity and clustering), social network analysis (e.g. using relations between document entities to cluster documents describing the same event into network communities) and machine learning (e.g. extraction of document features to identify context and create clusters). Event detection research in TDT has two main approaches: document pivot and feature pivot. The goal of the document pivot approach is to create document clusters that describe the same event and then extract the appropriate features from them to categorize incoming articles [11]. Petrovic, Osborne, and Lavrenko [130] introduce a document pivot approach where Locality Sensitive Hashing is used to cluster documents. The feature pivot approach focuses on detecting hidden features to cluster documents and identify news events [71, 173]; these approaches are often related to topic models based on Latent Dirichlet Allocation (LDA) [20].

TDT research can also be categorized as either live news event detection (dealing with document streams processed on-line or in another live mode) and retrospective event detection (off-line discovery of events in a historical corpus) [182]. Many event detection methods are based on some form of time series analysis applied to word frequencies. Kleinberg [84] used an infinite state automaton to model changes of word frequency in a document stream, with state transitions considered as events. Allan and Lavrenko [11] used a modified version of TF/IDF (term frequency-inverse document frequency, a form of weighted word frequency metric) to identify events as document clusters, also weighting by the time separation between the current document and the candidate event cluster. Since future document features are not known, the on-line approach of this algorithm needed to estimate the IDF metric which these authors approached by using an auxiliary data set.

Aiello, Petkos and Martin [4] in their BNgram method utilize a $df - idf_t$ score combined with named entity recognition tools to detect and characterize emerging trends on tweets. Another approach to event detection has a focus on word correlations, which are usually measured by distributional similarity [94, 170] or the number of word co-occurrences [133]. Fung, Yu, Yu and Lu [58] worked on detecting important bursty events in text streams. Their technique detects a set of features that correspond with a number of events in a given time window. The features are identified by statistically modelling the frequency of each individual word in an incoming document with a binomial distribution. Then these features are associated with events and time series analysis is used to detect significant changes which could correspond to an important event. This approach uses a large number of features and can be computationally expensive, in addition high frequency words may not be always useful for user interpretation of the content of a detected event. Another approach for event detection with Twitter data is to use recognized named entities in the text, cluster the documents that contain each entity, then apply machine learning algorithms to decide whether the selected documents constitute an event regarding the detected entity [4, 132].

A few studies have incorporated network analysis into their event detection methodology. Melvin, Yu, Ju, Young, and Wang [107] introduced a new approach where they create a noun phrase network (PhraseNet) by extracting high frequency phrases from tweets and linking them by their co-occurrences. After creating the network, events are identified by running a community detection algorithm using the assumption that a network community represents a potential event. To validate which candidate events are actual events their distributions over time (time series) are monitored and characteristics such as the number of peaks, intensity of the peaks and distribution variance indicate the importance of the event. The idea of utilizing a noun phrase network is also used by Sayyadi, Hurst, and Maykov [148]. Their algorithm creates a network of extracted terms, which they call a KeyGraph (after Ohsawa, Benson, Yachida [123] and Mori, Miura, and Shioya [108]). Nodes in the KeyGraph are document keywords and edges are formed when those keywords co-occur in a document. Finally they group together nodes that are strongly connected, forming clusters that describe a topic.

2.1.2 Document summarization and context extraction

The growth of available information and data in written form on the world wide web makes it very difficult if not impossible for the users to consume and process them. This creates the need for systems and methodologies for processing text information and documents to extract the context and a summary automatically. Text summarization constructs summarized text that contains the most important information from the original document or group of documents [9, 59]. This makes the information more easy to consume and to understand by the users.

Identifying and summarizing the context of a document or a collection of documents is a non trivial task with many applications in research and in industry. There are many approaches for extracting the context of documents and creating a comprehensive summary that focus on different disciplines such as natural language processing, information retrieval, semantic analysis, conceptual modelling and information extraction.

There is a number of ways we can classify text summarization methodologies. The first one is based on whether we want to extract the summary from one document or from a group of documents [50, 185]. Extracting the summary of a group of documents has some challenges. One issue is redundancy where the same sentence or n-grams are included multiple times on the summary. Some approaches deal with redundancy by selecting sentences at the beginning of the paragraph and then compare their text similarity of the next sentence with the already chosen sentences and it is selected only if this sentence consists of relevant new content [147]. Another idea for dealing with redundancy is the Maximal Marginal Relevance approach from Carbonell and Goldstein [29].

Context extraction and summarization methodologies can also be categorized as extractive and abstractive [59]. An extracted summary is produced by extorting and putting together the most informative sentences and words from a document. On the other hand an abstracted summary is generated by combining words, phrases and sentences that are not necessarily included on the original document. That way the summary contains the context from the original document presented with a different form. Abstractive summarization is much more complex than extractive summarization and machine learning and natural language processing methodologies are often used to deal with such tasks.

Another application of text summarization could be applied on web based documents such as web sites, blog posts and forums. The amount of information shared on the web is enormous and the user needs tools in order to search and consume this information. Search engines are a really useful tool to retrieve relevant information but the volume is still too big to be processed by the user. For that reason web based summarization techniques extract the context and create a comprehensive output that reflects the content of those websites. Radev et al. [134] proposed WebInEssence, an information retrieval system that summarizes clusters of related web pages which can help users to explore retrieval results systematically.

2.1.3 Detection and monitoring of evolving communities

Online and social media are prolific platforms for the formation of user networks with both explicit and implicit connections between them. We can analyse and investigate those networks in order to detect and recognize communities of individuals that are grouped together based on how they connect with each other. However since the internet evolves over time and users are getting introduced or leave online mediums and social platforms the need of methodologies for detecting evolving communities and monitoring them over time emerges.

The interactions between network actors are most likely to evolve over time in social networks. Friend relationships are formed and getting terminated in social media platforms such as Facebook or LinkedIn, people are following or un-following other users on Twitter or Instagram or contribute on forums like Stack Overflow or Reddit by creating or answering posts. Other forms of social interactions can also be considered such as phone calls and messages as well as communication via e-mails. The rapid growth of social media platforms has also resulted in high volume data sets containing information about the online interactions of users. This has created new challenges on processing all this information such as the availability of such data sets due to legislation or company policies, the effort and hardware required to process it and the need for new methodologies and approaches that can handle such tasks. Evolving networks are getting more relevant for analysing social interactions on the web. Several overviews of social network analysis in evolving networks can be found in the literature [2, 14, 17, 129, 154, 160].

Tabassum, Pereira, Fernandes, and Gama [158] provide a brief overview applying social network analysis on a variety of networks including static, evolving and ego networks. Aggarwal and Subbian [2] published a survey about evolutionary network analysis and how it can be applied on different fields or tasks including social network analysis. Spiliopoulou [154] tries to focus the scientific community attention into the volatility of social networks. She addresses important questions and challenges, such as, how social networks evolve and whether we can detect mechanisms that contribute on their evolution and model those procedures. What consists a community in an evolving network and how communities emerge and decline over time. Thompson et al. [160] applied temporal network theory in the context of functional brain connectivity. They incorporate network evolution into the network theory and methodologies and present how we can transform well known static methods to apply for evolving networks. They also introduced methods for evolving networks and suggested adaptations for existing ones. Beck et al. [17] provide a ranked taxonomy of evolving network visualization approaches by systematically categorizing and tagging publications. They classify those methods into two categories using the time dimension as the primary feature, with evolving network visualized as animated diagrams or as static charts based on a timeline. Ohsaka et al. [122] propose a real-time fully dynamic index data structure designed for influence analysis on evolving networks. They remodel the data structure of the state of the art sketching method original created by Borgs et al. [23] and create analogous update algorithms for two kinds of queries, influence estimation and influence maximization, which have real life applications such as viral marketing and arising trend detection. Peel and Clauset [129] are establishing the problem of detecting change points in evolving networks which requires the identification of time periods, that global network patterns of actors, drastically change and measure the quality and influence on the network of such events. They also introduce an approach to manage this task which incorporates a generalized hierarchical random graph model with a Bayesian hypothesis test to quantitatively establish if, when, and precisely how a change point has occurred. Aynaud and Guillaume [14] are evaluating the performance of three community detection algorithms when the input network has slightly changed. The algorithms they use are the Fast Greedy from Clauset et al. [36], the Louvain algorithm [21] from Blondel et al. and the Walktrap algorithm [131], which is based on the idea that a small random walk will stay inside the community from where it's originating because there are many links inside and few bridges leading outside. They propose an improvement for the Louvain algorithm in order to track communities on evolving networks. Their contribution on the algorithm is a new parameter which is a trade off between community stability and quality.

2.2 Background

2.2.1 Natural Language Processing

Natural language processing (NLP) [19, 34, 95, 101] is the field of computer science that enables us to develop methodologies that analyse, "understand", and extract useful information from human language. It helps us organize previous knowledge and execute a plethora of different tasks, such as automated translation, named entity recognition, point of speech tagging, sentiment analysis, topic extraction, and text summarization. Natural language processing is considered a challenging field in computer science. Human language can be ambiguous, contain jargon and mistakes or abbreviations. To create methodologies that understand and extract knowledge from the human language we not only need to understand the meaning of individual words but also the underlying notions and how they are connected with each other in order to form a meaningful concept. Although processing and understanding natural language is a trivial task for a human, it is a challenging undertaking for the machines.

Natural Language Processing Tasks

In this section we will briefly introduce the main natural language processing tasks. Some of them have immediate application on the real world and other are an intermediate step to handle bigger and more complex problems.

Stemming

Stemming and Lemmatization are text normalization methodologies with the purpose to preprocess the input text, words, and documents. In natural language processing, we may want our method to acknowledge that the words "jump" and "jumped" are different tenses of the same verb. This can be the concept of reducing different kinds of a word to a core root. With stemming we reduce morphological variants of a root word into its base form. In a search query we often get variants of a root word [82, 174]. For example if the query contains the keyword "work" we might also need to return the words "working" and "worked". Here "work" is the stem word.

Part of Speech Tagging (PoS)

Part of speech (PoS) tagging is a process which refers to categorizing words in a document in correspondence with a particular part of speech, depending on the definition of the word and its context. The biggest challenge with PoS tagging is ambiguity. In English, many common words have multiple meanings and therefore multiple PoS tags. The goal of a PoS tagging is to resolve this ambiguity with precision based on the context of each word [63, 167]. For instance, the word "face" can be a noun or a verb.

Tokenization

Tokenization is a preprocess step for analysing natural language. It is the procedure of dividing a stream of textual data into words, terms, sentences, symbols, or some other meaningful elements called tokens. Methods that perform tokenization are often called tokenizers. A tokenizer breaks natural language input into chunks of information that can be considered as discrete elements [43, 166]. The token occurrences in a document can be used as a vector representing that document, making it suitable for other methods such as classifiers.

Syntactic Analysis

Syntactic analysis extracts the logical meaning of each sentence of the input document. It takes into consideration grammatical rules in order to define the logical meaning as well as the correctness of the sentences. It is the process of analyzing natural language with the rules of formal grammar [153, 155]. The syntactic analysis basically assigns a semantic structure to text. It is also known as syntax analysis or parsing. Figure 2.1 displays the extracted syntax tree of a simple sentence.

Named Entity Recognition

Named entity recognition (NER) [86, 117] is the NLP task that detects entities on the text that belong to a specific semantic category, such as persons, locations, organizations and currency. Named entity recognition can be divided in two major tasks. The first one is to detect each word or group of words that refer to an entity. This phase is usually treated as a chunking problem since in almost every case the named entity is a series of consecutive words with no nesting. The second task is to classify each detected entity into which category it belongs. Figure 2.2 displays an example of detected entities in an article.

Topic Modeling

Topic modeling [8] is the task of discovering the main topics occurring in a set of documents. Given an input document about a specific topic we would expect similar words to this topic to appear more frequently into the text. By creating clusters of similar words from a set of documents we are

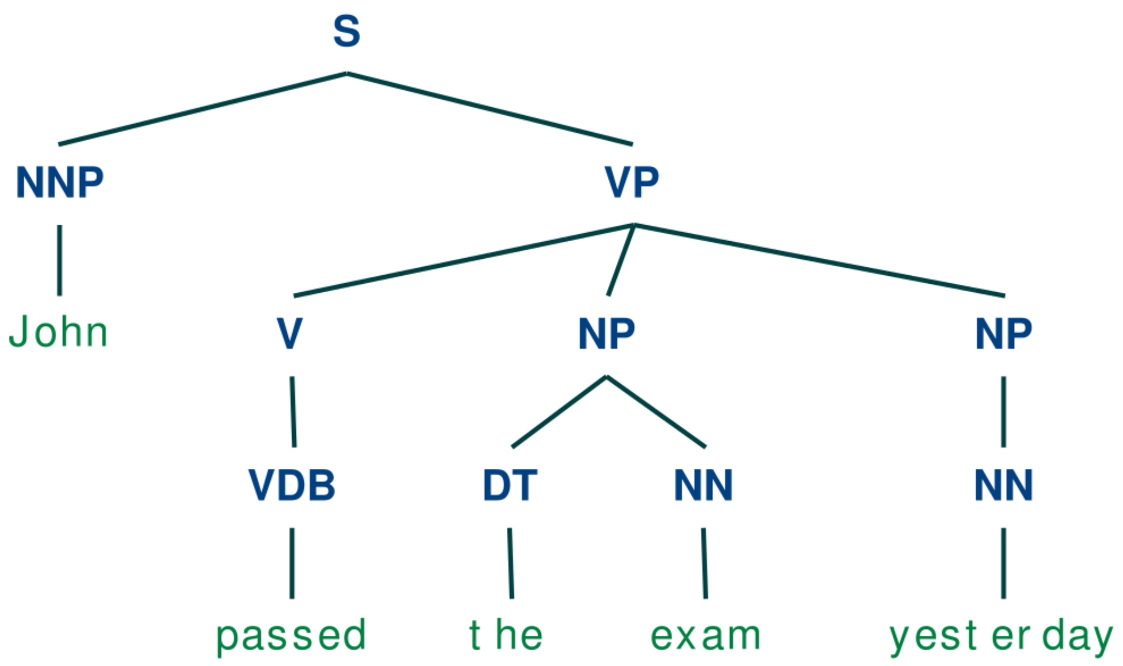


Figure 2.1: Syntax tree of a simple sentence. We can observe the point of speech tags for each word.

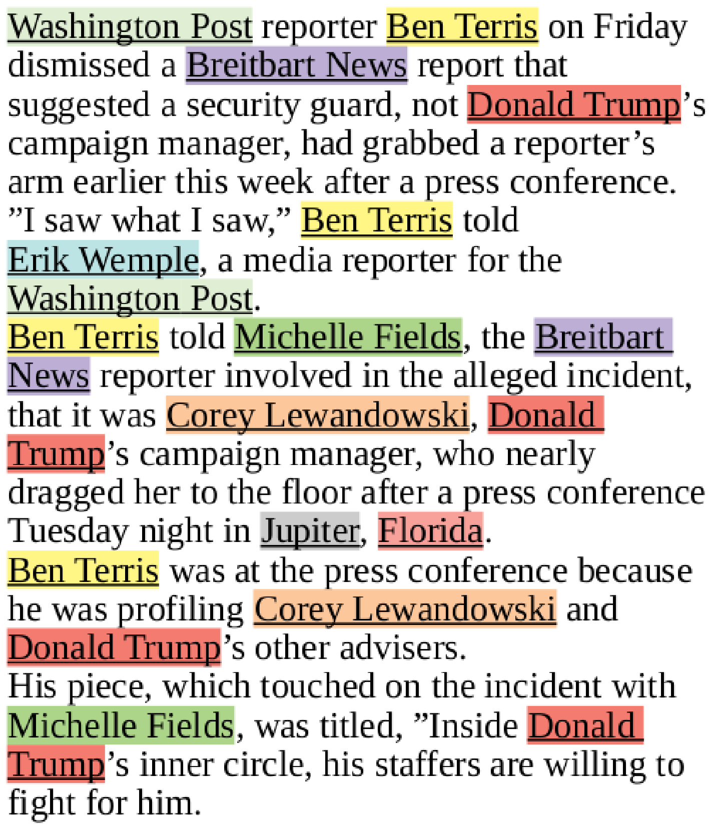

Washington Post reporter Ben Terris on Friday dismissed a Breitbart News report that suggested a security guard, not Donald Trump's campaign manager, had grabbed a reporter's arm earlier this week after a press conference. ”I saw what I saw,” Ben Terris told Erik Wemple, a media reporter for the Washington Post.
Ben Terris told Michelle Fields, the Breitbart News reporter involved in the alleged incident, that it was Corey Lewandowski, Donald Trump's campaign manager, who nearly dragged her to the floor after a press conference Tuesday night in Jupiter, Florida.
Ben Terris was at the press conference because he was profiling Corey Lewandowski and Donald Trump's other advisers.
His piece, which touched on the incident with Michelle Fields, was titled, ”Inside Donald Trump's inner circle, his staffers are willing to fight for him.

Figure 2.2: Detected named entities on an article. Multiple detected instances of the same entity are highlighted with the same color.

able to extract and specify the underlining topics. A topic model utilizes this in a mathematical framework, which allows analysing a set of documents and detecting, based on the occurrences of the words in each, what the topics might be and how similar a given document is to a given corpus of documents.

Text Summarization

Text summarization [46] is the process of shortening a document, to create a set of sentences or words (summary) that represents the most essential information within the original document, without losing any vital segments of the document. Text summarization methods can be divided into two main groups, extractive and abstractive methodologies. In extractive methods the main target is to detect the significant components of the text and add them to the summary while keeping them intact. In abstractive methods the target is to process the overall context of the document, detect the important parts of it, and generate new text that briefly narrates those parts.

Machine Translation

This field of language processing includes methodologies to translate text or speech from one language to another [124]. It is considered by many, one of the most challenging tasks given the fluidity of human language. The most simplified approaches are using word substitutions between the two languages but later methods have introduced statistical and deep neural network models achieving state of the art results [99, 187]. Other approaches are focusing in specific contexts excluding translations that do not fall on a given context [39].

Sentiment Analysis

Sentiment analysis [106] is the field that analyses human emotions, behaviour, expressions and opinions in text documents. It can be applied on a plethora of different contents such as surveys, product reviews, tweets, and on live forums. Various applications can benefit from sentiment analysis such as customer service and support, marketing and market analysis as well as news mining and event detection. There are three main approaches for opinion mining, the first is *Knowledge Based* which classifies text based on affect words such as happy, angry or pleased. The second is *Statistical Methods* that utilize machine learning techniques and deep learning. Lastly *Hybrid Methods* combine machine learning techniques with ontologies and semantic networks to detect implicit connections between text that does not explicitly expresses any sentiment with text that does so [3, 178].

Stanford Natural Language Processing Library

The Stanford natural language processing group is a collaborative project mainly created by professors, researchers, programmers and students from Stanford University. They are working on algorithms that enable computers to process and understand human languages. Their work ranges from basic research in computational linguistics to basic applications of NLP in human language, and it covers fields such as sentence understanding, automated question answering, machine translation, syntactic analysis, sentiment analysis, named entity recognition and text models. Their CoreNLP [102] library has been widely used in human and computational social sciences research. A unique characteristic of the Stanford NLP group is the efficient combination of advanced and deep linguistic modeling of data analysis with innovative machine and deep learning. The groups research has lead to development of technologies that widely cover the NLP field in many languages such as English, Arabic, French, and Chinese. Their research has lead to the creation of the CoreNLP library which is a complete and widely used tool set for working with NLP tasks.

In our work we used two tools from the CoreNLP library, the named entity recognition (NER) and the sentiment analysis tools. The Stanford NER tool is a Java implementation of a named entity recognizer. It comes with well engineered feature extractors for Named Entity Recognition, and many options for defining feature extractors. The group has made available classifiers that recognize three classes of entities, Persons, Locations and Organizations and there are versions for different languages. The classifier is a CRFClassifier [157] which is an implementation of (arbitrary order) linear chain Conditional Random Field (CRF) sequence models. And it can be also trained in training sets provided by the user.

The Stanford Core NLP Sentiment Analysis tool was developed by the Stanford NLP group as a module in the Stanford Core NLP toolkit. Its sentiment analysis model was trained using a recursive neural tensor network on a movie review data set. In contrast to earlier sentiment analyzers, in which a bag of words was used and word orders were ignored, the Stanford Sentiment Analysis tool parses input text into sentiment trees, where each leaf node refers to a word, every word in the input text is thus represented as a vector. Instead of producing numerical sentiment score, the results generated by the Stanford Sentiment Analysis tool are in the form of discrete categories, namely "very negative", "neutral", "positive" and "very positive".

2.2.2 Social Network Analysis

Social network analysis studies complex networks to recognize and utilize key features of components and detect meaningful structures, in order to manage their life cycle and predict their evolution. It examines the behavior of individuals at the micro level, how those individuals relate with each other (network structure) at the macro level, and the interactions between the two. A typical social network representation is a graph which has nodes for people, and edges connecting two nodes to represent one or more relationships between them. When we study small networks we can represent them visually and detect patterns of connections, clusters of nodes, and nodes that have significantly more connections than the rest of the network or being the only connection between two clusters of nodes. On bigger networks the task of visualizing it gets very challenging and many times impossible to perform, creating the need for graph analytic metrics and techniques for computing node and graph attributes. There are two categories of measurements that we can apply on a network. The first category provides information about the position and significance of individual entities (nodes and edges), the second category contributes with insights about the global structure of the social network. There is a vast literature about social network analysis, its methodologies and applications. Here we cite some of the most well known [22, 55, 85, 150, 171].

Centrality Measures

Centrality measures indicate which nodes or edges on a graph are significant. The main question of centrality is to define what makes an actor more central than another one. Various criteria have been considered to define the centrality of an actor and can take into account various characteristics about them, such as the number of its neighbours, the distance from other actors, and the number of shortest paths crossing the actor. The main centrality measures are degree centrality, betweenness centrality, closeness centrality, eigenvector centrality, katz centrality, and Pagerank [56, 118].

Degree Centrality

The degree centrality recognizes as most central or important nodes with the highest degrees (number of neighboring edges). In other words it counts how many neighbors a node has. Nodes with high degree in the network are considered more "central" and might be important to the network analysis. A weakness of this measure is that it does not take into account the centrality of the adjacent nodes, so we can have the case where two nodes have the same degree but one of them is connected with nodes with higher degrees making it more important. For directed networks, we have two versions of the measure: in-degree which is the number of in coming links to a node, or the number of predecessor nodes, out-degree is the number of out going links from a node, or the number of successor nodes. Most of the times, we are interested in in-degree, since in-links are given by other nodes in the network, while out links are determined by the node itself. For example in a scientific papers citation network the in-degree is more important than the out-degree, on the other hand in epidemic networks the out-degree might be more significant because it could characterize persons that accelerated the spread of an infectious disease. In the case of weighted networks the weighted degree of a node is the sum of the weights of the node edges.

Community Detection

Social networks represent the interactions and relationships of individuals. Such individuals tend to form clusters and communities based on their common preferences, choices and characteristics. Detection of these communities can be beneficial for numerous applications such as finding a common research area in collaboration networks, finding clusters of users that are purchasing the same products for recommendation systems and other marketing related tools, and finding protein interaction networks in biological networks. There is a broad literature proposing and discussing community detection algorithms and their applications [35, 61, 88, 90, 125].

Detecting communities of nodes in a network is a complex problem due to the number of definitions of what a community is. Generally, a community is a connected group of nodes densely connected with each other and sparsely connected with the nodes of the rest of the network. This criterion can be satisfied by a large number of different approaches which makes it difficult to establish a definition that can be universally accepted by the scientific community. Additionally, depending on the scientific field, the desired communities can have different properties. For example communities with or without common nodes, weighted or directed edges. For these reasons many different paths have been explored to reach an algorithm that satisfies the needs created from a specific scientific field.

We can identify four main network patterns to detect cohesive clusters of network entities that play an important role in community detection [150].

Cut: A cut is a partition of the nodes of a network into two disjoint subsets. Any cut has a set of edges that have one endpoint in each subset of the partition. These edges are said to cross the cut.

Connected Component: A connected component is an induced sub network in which any two nodes are connected to each other by paths, and which is not connected to any additional nodes in the rest of the network. Where a path is a finite or infinite sequence of edges which join a sequence of nodes which are all distinct.

Cycle: Network cycle is a non-empty trail in which the only repeated vertices are the first and last vertices.

Clique: A clique is a subset of network nodes such that every two distinct nodes in the clique are adjacent. That is, a clique of a network is an induced sub-network that is complete.

A large number of community detection algorithms has been suggested to find meaningful node communities with minimum processing time. Most of these algorithms are based on the optimisation of objective functions. Modularity [119] optimisation by so far is one of the most widely used techniques among them. However, modularity optimisation is an NP-hard problem.

Modularity

Modularity is a network metric which measures the degree to which densely connected components of a network can be divided into separate communities which interact more among themselves rather than other communities. In a highly interconnected system with low levels of modularity, a shock to one component may diffuse to other parts of the network and thus increase the risk of a system wide collapse. On the other hand, in a system with high levels of modularity, a failure to one component can be better contained and the disturbance will be less likely to diffuse to other components.

Community Detection Algorithms

A plethora of community detection algorithms have been proposed in the literature. These methods allow researchers to reveal communities in networks which can be used in wide range of applications such as recommendation systems, innovation diffusion and viral marketing. Here we will briefly describe the algorithms we used on our research.

Fast Modularity Optimization by Blondel et al. [21]. This algorithm, also known as the Louvain method, is a greedy approach to modularity maximization. The algorithm starts with assigning each node to a singleton and progresses by moving nodes to neighboring clusters in order to improve modularity, and terminates when there is no change in the modularity value. A new phase then starts where the formed communities are considered as nodes while the weights of inter community edges are aggregated and form an edge connecting the newly created nodes. The previous process is repeated on the new network.This method has complexity linear with the number of the network nodes and unfolds a hierarchical community structure with increasing coarseness and meaningful intermediate communities.

Maps of Random Walks by Rosvall and Bergstrom [142]. This algorithm, also known as Infomap, is a flow based and information theoretic clustering approach with complexity linear with the number of the network nodes. It uses a random walk as a proxy for information flow on a network and minimizes a map equation, which measures the description length of a random walker, over all the network clusters to reveal its community structure. Infomap aims at finding a cluster of nodes which generates the most compressed description length of the random walks on the network.

2.2.3 Evolving and dynamic networks

Social network analysis helped us understand the importance of the time dimension on social media. The topologies of networks representing social interactions tend to evolve over time creating the need of models that can help us keep track to those changes and be able to apply measurements and algorithms on them. Although on the rest of this work the terms evolving and dynamic networks are used interchangeably those two terms are not exactly the same. Evolving networks are a subset of dynamic networks. They focus on the changes happening to a network during a specific time period. On the other hand dynamic networks are focusing more on the changes that are happening to a network without necessarily taking into account the time aspect of an occurring change.

There are four main approaches to form a network taking into account time evolution:

Static Network. A static network contains every actor that took part on the phenomenon we are researching as well as every interaction between actors. Labels on nodes and edges can be introduced to help us identify the time period that a node/edge is added to the network.

Edge Weighted Network. Weighted networks are static networks where their nodes and edges are weighted based on the number of occurrences of the network actors and their interactions. Both this approach and static networks fail to capture any dynamics occurring on the network creating the need for the following two approaches.

Network Snapshots. Network snapshots are a collection of static networks, each one created on a specific time window of the total duration of the examined social phenomenon. That way changes on the topologies of the networks can be easily identified. However this approach increases the complexity of the task.

Temporal Networks. Temporal is a network that every node and edge contains time information regarding the specific instance that it got introduced to the network and the instance that it stopped being part of it. They allow the analysis of the network dynamics in detail, increasing on the other hand it's complexity and requires analytical methodologies adjusted for this kind of networks.

We can distinguish evolving networks into two categories based on how connections between nodes are formed. The first category is interaction networks where the edges represent interactions between two nodes, such as communication in person, phone calls, emails or video calls. The second category is relation networks where edges represent more stable relations, such as paper citations, sharing common attributes or belonging to similar social groups. Relation networks are well defined at any time spot and can be analysed by traditional network measurements and algorithms. On the other hand on interaction networks we need to define time windows where we can aggregate the occurrence of each edge to create a stable network. This must be done since most standard approaches do not produce comprehensible outputs when the lifetime of an edge is almost instant.

Network snapshots and temporal networks are two representations of evolving networks each having its advantages and disadvantages. An analyst could benefit using them based on the task he aims to achieve. Network snapshots should be preferred if the input data are separated in time frames (days/weeks/years), however if the data contain specific timestamps of an interaction between two actors (emails, phone calls) temporal networks could also be considered [140]. Computationally it is more efficient to calculate network measurements or apply algorithms, such as community detection, on network snapshots instead of temporal networks since we only need to apply each task once per snapshot in contrast with temporal networks where we need to rerun each task for each network modification that have occurred.

2.2.4 Community Life Cycle on Evolving Networks

The time dimension of evolving networks drastically affects the way we analyse social networks. One of the most common tasks in social network analysis is community detection where methodologies to achieve this task should be adjusted for evolving networks. Being able to detect meaningful sub topologies from underlying social phenomena is not a trivial task and its difficulty gets worse when we want to monitor the evolution of those detected clusters over time. In this chapter we will define the community detection problem on evolving networks and introduce solutions found in the literature to help us study the life cycle of the detected communities.

A definition of community life cycle has been given by Rossetti and Cazabet [140]: Let C be a community, its life cycle, which describes the complete history of its evolution, is composed of the directed acyclic graph (DAG) such that 1) the roots are the birth events of the community, or of the predecessors that it emerged from, 2) the leafs are death events, corresponding to death of C and of its successors, if C has been split, and 3) the central nodes are the remaining actions of C, its successors, and predecessors. The edges of the tree represent transitions between subsequent actions in the community's life.

As we mentioned on the previous chapter there is a lack of a standard and a plethora of different definitions of what a community is. A broad definition could be that a community is a set of nodes that are more connected with each other than the rest of the network. To expand this definition we also want to monitor how those communities evolve over time and define the important points of a community's life cycle. The related literature with community evolution points to two actions for tracking a community's dynamics, and those are the emergence and vanishing of a node or edge. This kind of changes on a network topology could affect the results of community detection algorithms as well as other measurements.

The actions of emergence and vanishing can cause a number of transformations to the communities we are monitoring. The first mention on the literature was mabe by Palla et al. [126] where they specified six transformations (birth, growth, merge, split, contraction and death). We can also find the transformation persist when no significant changes were made on the community. Cazabet, Rossetti and Amblard [31] introduce an eighth transformation called resurgence which occurs when a community emerges, vanishes and then reemerges. Rossetti and Cazabet [140] also discus about the community life cycle and the task of community detection on evolving networks.

The eight transformations of an evolving community are displayed on figure 2.3 and described below:

Birth. The initial emergence of a new community created by any number of nodes (Figure 2.3a).

Death. The community stops to exist. All nodes are either vanish from the network or join another community (Figure 2.3b).

Growth. New nodes are introduced on the community either coming from other communities or they just joined the network, causing the community to increase in size (Figure 2.3c).

Contraction. The community size is decreased due to nodes joining other communities or vanish from the network (Figure 2.3d).

Merge. Two or more communities merge into a single one (Figure 2.3e).

Split. A community splits into two or more components (Figure 2.3f).

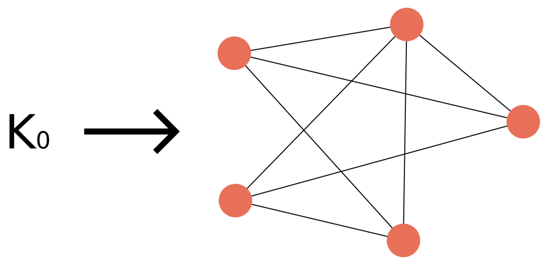
Continue. A community persists on two consecutive time slots (Figure 2.3g).

Resurgence. A vanished community rebirths keeping almost the same characteristics as if it has never stopped existing (Figure 2.3h).

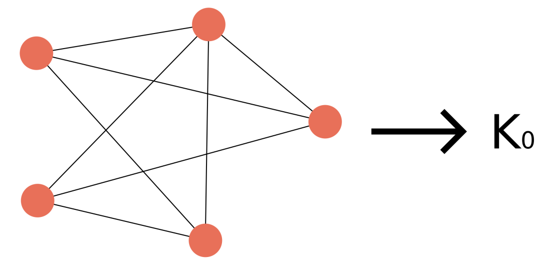
There are some transformations that need more specific treatment when we handle communities in evolving networks. Those are merge, split and resurgence which are often annotated based on their similarity with components of previous or next time slots. For example in the merge transformation the new community that is formed can be characterized as one of the previous ones or as completely new. In the first case we have an *absorption* and we need to decide which of the previous ones will persist to exist and which one will vanish. Some scenarios could be to remove the community with fewer nodes existing in the merged one or remove the most recent of the two. In the second case the merged communities could create a different new component which should be annotated as a separate one, resulting the two former ones to vanish. In this case we have a *replacement* of the old communities. Similar approaches can be followed in the case of a split. We then need to decide if the previous community will remain as one of the new ones or if it should vanish and two or more will be born.

2.2.5 Community Detection Algorithms for Evolving Networks

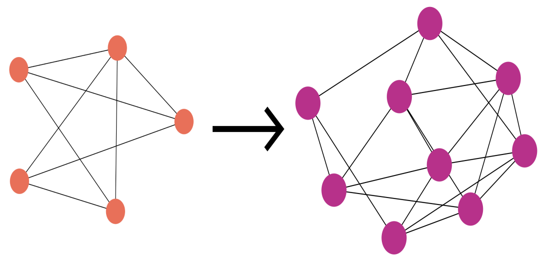
In the literature there are many community detection algorithms for evolving networks. Previous works survey those algorithms and try to classify them in different categories [13,70,103,140]. The classification of [140] best matches with the angle that this work studies evolving networks, that is the time dimension.



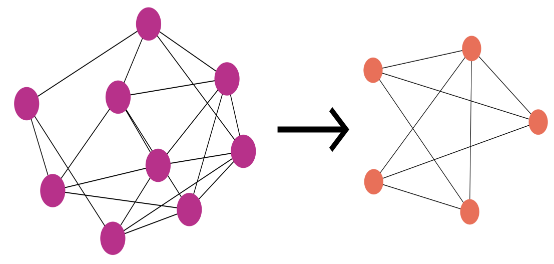
(a) Birth of community. A new community emerges that did not exist before the current time slot.



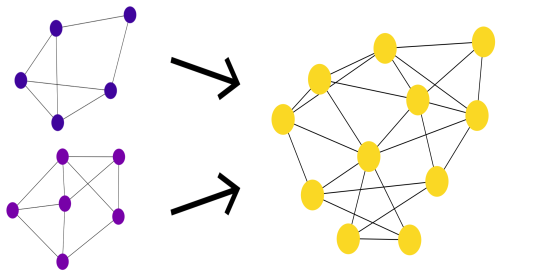
(b) Death of community. The community disappears on the current time slot.



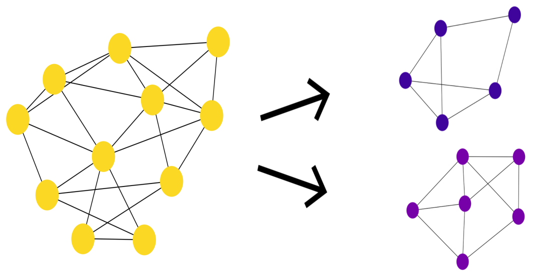
(c) Growth of community. New nodes are joining and the community grows bigger.



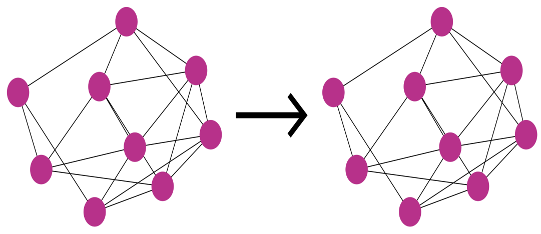
(d) Contraction of community. A number of nodes stop being part of the community and it decreases in size.



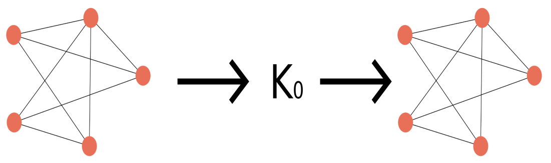
(e) Merge of communities. Two or more communities are combining their nodes and edges forming a new one.



(f) Split of community. A community is getting split forming two or more new communities.



(g) Community continues to exist.



(h) Community resurgence. A community vanishes and after one or more time slots re emerges.

Figure 2.3: Community life cycle transformations.

In this categorization there are three high level groups describing evolving networks and their features. Each group contains subcategories of detection algorithms that align with the high level description of an evolving community.

The high level groups of evolving communities are:

Instant Optimal Community Detection. This definition considers that the attributes of a detected community in time slot t depend only on the network characteristics at the time slot t. For monitoring the evolution of a community we need to match different communities that were detected on different time slots. All communities that were detected on time slot t are optimal for the network topology of the same time slot and completely independent from the networks of different time slots.

Temporal Trade off Community Detection. In the second group a detected community on time slot t depends on the network topology of this time slot as well as on networks from previous time slots. There is a trade off between the an optimal community formation at t and a community that also takes into account how it evolved in the past. Lastly we should also point out that those communities do not depend on future modifications of the network, making it suitable for on the fly community detection.

Cross Time Community Detection. In this group the community detection task takes into consideration the whole network evolution rather than a single time slot. The detected communities depend on both previous or future time slots of the network.

Instant Optimal Detection Algorithms.

This family of algorithms is identical with the approaches followed for detecting communities on non evolving networks. This is the approach we took for community discovery on evolving networks for both the event detection in news streams methodology and for analysing the user communities of Stack Overflow. A series of networks snapshots is produced and on each one of them the algorithm is applied identifying the optimal partition. Following that partitions from different time slots are compared and being matched based on common characteristics they might share (common nodes for example) forming an evolving timeline of one community.

The technique introduced from Aynaud et al. [13] is a good example of an *Instant Optimal* algorithm. It consists of two steps, *Identification* where static communities are detected for each time slot of the evolving network, and *Matching* where the communities of each time slot are being aligned with communities of previous time slots.

The main advantages of instant optimal algorithms are their simplicity and that they can be easily parallelized. This group of algorithms extends the existing detection methods of static networks. For each time slot a traditional algorithm can be applied. Matching the detected communities is also a subject that has been extensively studied since it derives from the set matching theory. The nature of instant optimal techniques allows to easily parallelize the community detection step since we can apply a detection algorithm for each network co-responding to a time slot independently. Given the computational complexity of community detection this can significantly improve the performance of detection algorithm on evolving networks. A drawback of the this category of algorithms is the inconsistency of community detection algorithms. The detected communities are not exactly the same on different instances of an algorithm on the same network, those differences can be big in some cases. This can affect the matching step of instant optimal algorithms since the differences between communities of different time slots can vary. There are two approaches handling this problem. In the first one the community cores are extracted and used in the matching step [143], this reduces the inconsistencies caused by the detection algorithms. The second approach gathers the partitions for every time slot and matches the communities simultaneously [65].

There are three subcategories of instant optimal detection algorithms. They differentiate between them by the approaches they use to match the detected communities while the detection itself is essentially identical. Those subcategories are:

Iterative Similarity Based. On these approaches a measurement of how similar two communities of different time slots are is calculated. Such measurements can be Jaccard similarity [79] between nodes, graph edit distance [146] or Simrank similarity [81]. This is also the approach we used in our methodology. Time consecutive communities with the highest similarity are included in the same evolving community. Methodologies using this approach are [24, 25, 42, 74, 77, 126, 143]

Iterative Core Nodes Based. Approaches in this subcategory identify a number of significant nodes for each community as core nodes. Core nodes could have high centrality scores or meet other criteria. Communities on adjacent time slots including the same core nodes can be matched and be included on the same evolving community. The methods of Wand et al. [169] and Chen et al. [33] are two examples of such approaches.

Multistep Matching. This subcategory of methods does not match communities only between consecutive time slots but also between time slots that are far apart in the network evolution. The matching is done using the same similarity measures we mentioned on iterative similarity based methods. The methodologies of Falkowski et al. [48], Spiliopoulou [49] and Morini et al. [109] are using the multistep matching approach to detect, monitor and visualize evolving communities.

Temporal Trade Off Community Detection

This group of algorithms processes evolving communities since the first time slot of the network evolution. The matching process compares not only communities of consecutive time slots but also communities of n previous time slots. The two main steps of temporal trade off algorithms are *Initialization* where communities are detected on the first time slot of the evolving network, and *Update* where for each new time slot the detected communities are being matched with communities of every previous time slot.

The main advantage of temporal trade off approaches is the ability to cope with the inconsistency of community detection algorithms. On the other hand those methods can not be easily paralellized since each step needs the communities detected at previous time slots as input.

We can separate temporal trade off algorithms in three subcategories:

Update by Global Optimization. Those methods are using the partition created at the previous time slot as seed to initialize a global optimization process at the current time slot. The method of Aynaud and Guillaume [14] is a good example of this subcategory, where the authors are using the Louvain algorithm. The works of Görke et al. [67], Shang et al. [151] and Alvari et al. [12] belong to this subcategory.

Update by a Set of Rules. This group of approaches keep track of network changes, such as node or edge emerging or vanishing, that occurred on consecutive time slots, and define a set of rules dictating how those changes affect the detected communities. Methods in this subcategory are the works of Cazabet and Amblard [30], Duan et al. [45], Lee et al. [91], Zakrzewska and Bader [186], and Rossetti et al. [141].

Informed Community Detection by Multiobjective Optimization. This group of methods tries to balance the quality of detected communities on a single snapshot with the coherence of the matched communities from previous snapshots. The goal is that a community is detected at a given time slot t to represent the natural evolution of the community detected on the previous time slot (t - *1*). Approaches of this category are the work of Tang et al. [159], Yang et al. [181], Crane and Dempsey [37], Görke et al. [66], and Folino and Pizzuti [53].

Cross Time Community Discovery

In this family of algorithms the detection takes place in a network that aggregates the nodes from every time slot of the evolving network. Each node on this network corresponds to a node on a specific time slot. There are two kinds of edges on this network, edges that connect nodes on a single snapshot and edges that connect nodes of different adjacent snapshots. The advantage of those algorithms is that they do not face problems of instability and community drift. Some disadvantages are that events such as splits and merges can not be detected, also they are not able to handle real-time detection since those approaches require full knowledge of the evolving network states through time and the introduction of a new network snapshot will cause the algorithm to redo the whole process.

We can divide cross time detection algorithms in four categories:

Fixed Memberships, Fixed Properties. In this category the nodes remain to only one community and there is no emergence or vanishing of communities. This has the result that the communities are stable for the time period they are studied. The methods of Duan et al. [44] and Aynaud and Guillaume [15] belong to this category.

Fixed Memberships, Evolving Properties. The algorithms of this group are aware that communities are not homogeneous along time. For example, nodes in one community can interact more actively during some recurrent time slots, or their activity might increase or decrease along time. Each community has a profile that records the evolution of its activity. Methods in this category are the works of Gauvin et al. [60] and Matias and Miele [104].

Evolving Memberships, Fixed Properties. Approaches in this category allow node memberships to change as the networks evolves. For example nodes can switch between communities. Approaches of this category are the works Ghasemian et al. [62], Xu and Hero [177], Matias et al. [105], and Herlau et al. [72].

Evolving Memberships, Evolving Properties. Methods in this category have no limitations on how evolving communities behave, nodes can switch between them, communities can emerge or vanish, and their density can vary between different time slots. The methods of Mucha et al. [115], Himmel et al. [73], Viard et al. [165] and belong to this category.

2.3 Motivation

Although this work utilizes parts of many fields its main focus is to demonstrate applications of social network analysis in evolving networks for event characterisation and community detection. Combining SNA with evolving networks has been recently introduced and there is not enough research yet on what we can actually accomplish by combining those two fields. In this work we introduce a methodology for detecting news events in online media streams, it also utilizes other fields such as natural language processing and time series analysis. We applied this methodology in a series of different data sets and we were able to extract major events as well as a brief summary and the sentiment associated with the posts reporting the event. We also present what insights we can acquire from this combination on social media platforms like Stack Overflow. We were able to detect the major programmers communities, monitor their evolution over time and extract the main concepts and technologies that characterize them for each year. We strongly believe that this work demonstrates the advantages and capabilities of social network analysis on evolving networks and can potentially lead to more research related with our approaches.

This work attempts to answer three main questions related with its main objective:

- RQ1: Can we detect breaking news events from heterogeneous input streams by utilizing social networks analysis and evolving networks?

Detecting news events in online media is a challenging task due to the volume, velocity, veracity and variety of the input data. Here we demonstrate that the combination of SNA and evolving networks can produce a state of the art methodology for event detection. The main contribution of our work here is that monitoring the weighted degree of the network nodes (named entities) over time can produce an improved method for detecting peaks on the popularity of entities that does not only counts their appearing frequency but also the co-occurrence with other important entities on the same document. In our knowledge this is the first time that the combination of SNA and evolving networks is used to tackle this kind of tasks and displays the value we can benefit from it. On chapter 3 we introduce a new method for detecting and summarizing peaking events in heterogeneous news texts and evaluate its performance using news articles and tweets.

- RQ2: How can we extract the summary of the context of a detected event?

Another contribution is that using noun phrases, named entities, community detection and sentiment analysis the method of chapter 3 produces a comprehensive summary of each detected event. Noun phrases are effective for describing the detected event and alongside with named entities provide a description of it. Sentiment analysis on the other hand provides another dimension that characterizes the event. Based on what sentiment prevails on the detected event different decisions can be made.

- RQ3: Can we extract useful insights from online communities using social network analysis on evolving networks?

On chapter 4 we analysed the online programmers question and answer website Stack Overflow. The main target is to demonstrate how we can extract information about the relationships between programmers on the platform and what communities those individuals are forming. For relating users we used an implicit approach where two users are considered as connected if they share common programming language or technology tags. Those relations are forming structures where, after applying community detection algorithms, clusters of developers using particular programming languages or technologies are forming. We also monitored the evolution of the detected communities and whether programmers migrate to other communities, stay on the same or completely abandon the Stack Overflow platform. This work shows that applying SNA on evolving networks can produce insights of a social media platform that traditional approaches using static networks can not.

Chapter 3

Event detection and characterization on evolving networks

3.1 Introduction

The growth and spread of the world wide web has radically changed the way news content is published and disseminated to the public. Traditional news platforms have moved online, while user-generated content such as blogs and social media have expanded the variety and availability of media content. This ongoing transformation has many advantages, but also creates new challenges for human analysts whose role requires comprehension of new information, e.g. for trading decisions in financial markets, insurance risk estimation, or open source intelligence applications. The volume and velocity of the media ecosystem has become so big that human analysts cannot manually process and make sense of all the information that may be relevant. This creates the need for automated systems that can assist human analysts by processing large volumes of content from heterogeneous news streams in real time, detect and track breaking news events, and provide informative summarisations of complex media data sets. Such information can help users to make timely and well-informed decisions.

Here we present a methodology that utilizes social network analysis techniques for trending topic detection and characterization in news and social media streams. We make the assumption that news events link multiple entities (e.g. people, organizations and locations) at a particular point in time. We hypothesize that news events may therefore be effectively detected by studying the temporal evolution of a knowledge graph that links entities based on their co-occurrence in news documents. In this work, we explore this approach by developing a software pipeline that creates and analyses complex networks in which the nodes are entities and the edges represent entity co-occurrence in news articles and Twitter posts. Changes in node degree are used as an indicator of possible news events, which are then characterized using an extended knowledge graph that incorporates noun phrases alongside entities. Finally we apply sentiment and emotion analysis on a set of tweets related with each detected event to identify whether the sentiment of the users that are discussing about it is negative, positive or neutral. This approach helps us further understand the context of the detected event by giving a simple affective measurement based on the sentiment of the crowd discussing the event.

To evaluate our method we use two data sets. The first data set consists of articles from the *ALL The News* data set from the Kaggle website[1]. This data set contains 140,000 articles from major news sites of the United States of America, such as the New York Times, Breitbart, CNN, New York Post and Reuters. We perform a focused evaluation against human annotation and state of the art techniques for news event detection during the week of the 2016 Brexit referendum in the UK. The second data set is Twitter data relating to the FA Cup Final football match between Chelsea and Liverpool in 2012, which consists of tweets collected during the match by Aiello et al. [4]. To demonstrate the sentiment extraction from detected events we use three Twitter data sets from three different political elections:

- *US2012*: This data set contains approximately 3.6 million tweets where 2.1 million of them are retweets. They were collected during the US presidential election in 2012 by [4]. President Barack Obama for the Democrats was victorious over his Republican opponent Mitt Romney.
- *US2016*: This data set contains approximately 5.4 million tweets where 3.1 million of them are retweets. They were collected during the US presidential election in 2016 by [96]. President Donald Trump won the election for the Republicans against the Democrat candidate Hilary Clinton.
- *UK2019*: This data set contains approximately 4 million tweets where about 2 million of them are retweets. They were collected within our own research group during the UK General Election in 2019. The Conservative Party led by Boris Johnson won the election and achieved a large Parliamentary majority over the Labour Party led by Jeremy Corbyn.

The US2012 and US2016 data sets were collected using election-specific keywords, so contain mostly tweets related to the elections. The UK2019 data set was extracted from a larger data set of tweets geo-located in the UK and Europe. Therefore the UK2019 data contains tweets related to a wide range of topics. For each data set, we restrict the data to a 10-day time period spanning seven days before the day that the elections took place and two days after. The tweet IDs for all

[1]https://www.kaggle.com/snapcrack/all-the-news

three data sets are available online[2].

Finally we perform a retrospective analysis using our methodology to detect major events from the discourse that occurred on Twitter about the COVID-19 outbreak during the first six months of the pandemic. The main objective of this analysis was to detect the major events on that time period and generate summaries as well as extract the overall sentiment and emotions for each one of them. Another motive for this work is to test the NED methodology on a large scale data set of real world data. For event detection, we used the methodology we introduced and applied it on time windows of one day for the study duration of six months. For the tweets corresponding to each detected event, we then applied sentiment analysis and emotion detection techniques to each tweet and summarized how the public reacted to each event.

Understanding, monitoring and analysing the discourse about the COVID-19 pandemic on this platform can help us extract valuable insights about this crisis and understand how people perceive and react to it. More specifically we can detect the most significant discussed topics on the medium, extract the topics being discussed and monitor how they evolve over time.

Monitoring the detected topics can also help us understand the sentiment and emotions of the population that discusses them. It might be an effective way to measure people's thoughts on different topics and we can understand how people respond to various kinds of situations such as travel and movement restrictions, the introduction of vaccines, social distancing, new variants of the virus and more. The sentiment and emotions about a specific event could be utilized to predict how the public could react when similar events occur, since during this pandemic we have had a number of waves of COVID-19 cases. Furthermore, sentiment and emotion analysis can reveal impacts in society on a psychological basis. Various types of mental disorders such as anxieties, depression and panic attacks have manifested during this pandemic. We can estimate how severely a community is affected during a specific event by detecting the dominant sentiment and emotion when this event occurs. That way more specific counter measures against a given situation could be applied, mitigating the negative effects on the community that event created. For example, a community where the emotion of anger is dominating could be treated by easing the measures applied to it or a community where the emotion of fear is dominating could be treated by better informing the population about the event. Sentiment and emotion analysis for active users is an efficient way to monitor public opinion. In this pandemic, these types of approaches could significantly contribute to help governments and policymakers.

Although Twitter does not match the whole society it is a widely used platform where discourse about various topics such as politics is taking place. In this work we focus on how sentiments and emotions are evolving in this platform, without claiming that it represents the situation on the real world. The influence of Twitter in the public opinion is out of the scope of this work but the reader could refer to authors such as Stieglitz and Dang-Xuan [156], Ranco et al. [136], Tumasjan

[2]https://drive.google.com/open?id=1a8apanZWjuoxvNXI-2O8fdznKHwIWNYM

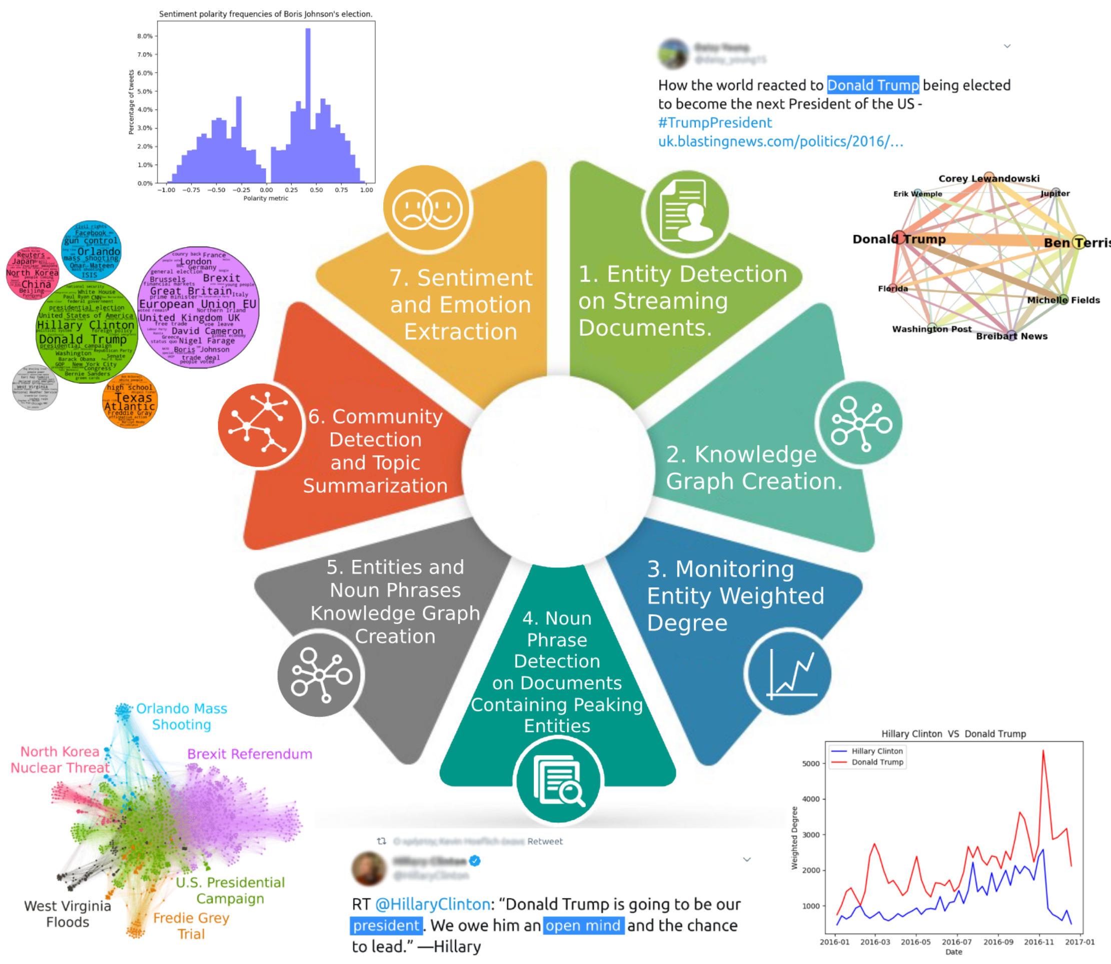
et al. [162], Abraham et al. [1], Kharde and Sonawane [83] and Imran et al. [78].

In the next section, we describe and explain our *Method.* Further sections present the *Evaluation* of our method and it's *Results*, before a final *Discussion* and *Conclusion* section consider the main findings, implications and future goals of this work.

3.2 Method: Network Event Detection (NED) for news streams

Network Event Detection (NED) makes the assumption that the occurrence of a trending topic or news event (these two terms are used interchangeably hereafter) is signified by changes in prominence of three kinds of named entities: Persons, Locations and Organizations. NED identifies such entities in a stream of documents (articles, tweets) and forms a sequence of networks in which the nodes are unique entities and the edges represent entity co-occurrence within a document. News events are detected by finding peaks in time series associated with individual entities; here we use weighted node degree as the key metric, though other statistics might be tested in future work. The news document stream is filtered down to only retain documents referring to these "peaking" entities, based on a working assumption that these are the most news-worthy entities. An entity-phrase network is then created to help interpret the detected events, which includes noun phrases as nodes alongside the previously detected entities. Community detection is then used to find communities within the entity-phrase network, with each found community considered to represent a candidate event. Finally sentiment and emotion analysis is applied to the documents associated with each event and the overall sentiment about it is extracted. Fig. 3.1 shows the main steps of the process.

The graph-based approach has three main advantages. Firstly, the graph structure allows the importance of an entity to be measured not only using the frequency of its appearance in documents but also taking into account the frequency of appearance of other co-occurring entities in the same document. Here an entity is important not only if it appears frequently, but also if it co-occurs with other frequently appearing entities. Secondly by utilizing a KeyGraph (the entity co-occurrence graph extended with noun phrases) we can generate comprehensible event summaries. Here applying the Louvain community detection algorithm [21] divides the KeyGraph into strongly connected entity/noun-phrase communities, each corresponding to a candidate event. Finally by using sentiment and emotion analysis we can reveal another dimension of the detected event which further assists us to understand and assess the context and impact of the event. Bellow we will describe each step of the NED methodology.


Sentiment polarity frequencies of Boris Johnson's election.
Percentage of tweets
Polarity metric
7. Sentiment and Emotion Extraction
1. Entity Detection on Streaming Documents.
2. Knowledge Graph Creation.
3. Monitoring Entity Weighted Degree
4. Noun Phrase Detection on Documents Containing Peaking Entities
5. Entities and Noun Phrases Knowledge Graph Creation
6. Community Detection and Topic Summarization
Corey Lewandowski
Erik Wemple
Jupiter
Donald Trump
Ben Terris
Florida
Michelle Fields
Washington Post
Breibart News
Orlando Mass Shooting
North Korea Nuclear Threat
Brexit Referendum
U.S. Presidential Campaign
West Virginia Floods
Fredie Grey Trial
Hillary Clinton VS Donald Trump
Hillary Clinton
Donald Trump
Weighted Degree
Date

How the world reacted to Donald Trump being elected to become the next President of the US - #TrumpPresident uk.blastingnews.com/politics/2016/...

Retweet

RT @HillaryClinton: "Donald Trump is going to be our president. We owe him an open mind and the chance to lead." —Hillary

Figure 3.1: Diagram of the main steps of the Named Entity Driven (NED) event detection method.

3.2.1 Entity Detection on Streaming Documents. Step 1 of the methodology.

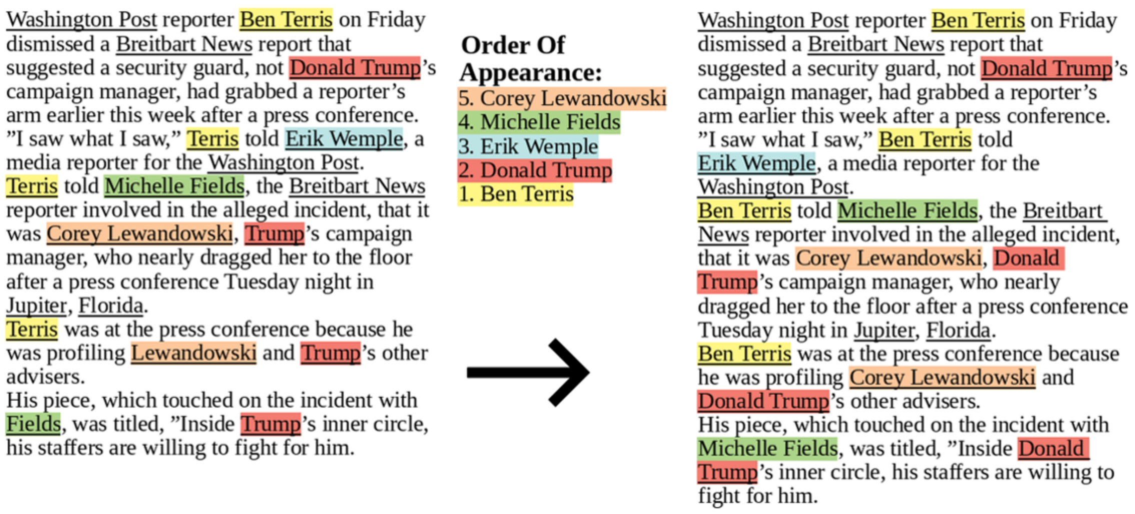
The goal of NED is to detect important events, given a stream of newspaper articles or a stream of tweets. There are many ways to access news streams or collections of such documents. For streaming tweets the user can refer to the Twitter API[3] where he can acquire API keys in order to be able to stream tweets related with the project he is working on. For streaming news articles the user can use APIs such as the News API[4] or the Newsdata API[5].

For each document we apply a named entity recognition (NER) technique in order to detect three kinds of entities: Persons, Locations and Organizations. We experimented with a number of different NER tools (specifically NLTK ne_chunk[6], Stanford NER [51] and Spacy[7]). We chose to work with the Stanford NER classifier for news articles because it is trained with the CoNLL[8] data set which consist of Reuters newswire articles, which is ideal in our case. For detecting named entities in tweets, we chose a different classifier. Tweets are qualitatively different from news articles: they are at most 140 characters long (in the 2012 data set used here, though the limit has been raised to 280 characters), they contain a lot of misspelled words (typos, jargon, lower case letters), and often lack sufficient context for determining the type of an entity. For entity recognition in tweets we chose the classifier of Ritter et al [138], which according to the authors outperforms the Stanford NER system for this task.

For news articles, we also supplement entity recognition with a disambiguation process. For Person entities, we replace single words (typically first or last names) with the full name of the entity. Each document is scanned for Person entities. When a single word Person entity is found, it is replaced by the most recent matching multi-word Person entity phrase that was found. If no match is found the single-word name is retained. Figure 3.2 presents an example of this procedure. For Location and Organization entities, we disambiguate by expanding abbreviations and reference to manually collated dictionaries of exceptions.

For disambiguating named entities on tweets we developed an approach consisting of two phases (applied in Step 1 of Figure 3.1). This is illustrated in Figure 3.3. In the first phase, tweets that contain both a single-name Person entity and a URL are identified using the classifier of [138], then the linked content from the URL web page is retrieved and the text content extracted using the *beautifulsoup* library [137]. All named entities are detected within the article text using the [51] classifier, which performs better and faster in larger documents such as articles. The main idea of this phase is that the linked documents will contain the entities of the tweet using both their first and last name. For that reason our approach is to try to match single name entities from the tweets

[3]https://developer.twitter.com/en/docs/twitter-api
[4]https://newsapi.org/
[5]https://newsdata.io/
[6]http://nltk.org
[7]https://spacy.io
[8]http://www.conll.org


Washington Post reporter Ben Terris on Friday dismissed a Breitbart News report that suggested a security guard, not Donald Trump's campaign manager, had grabbed a reporter's arm earlier this week after a press conference.
"I saw what I saw," Terris told Erik Wemple, a media reporter for the Washington Post.
Terris told Michelle Fields, the Breitbart News reporter involved in the alleged incident, that it was Corey Lewandowski, Trump's campaign manager, who nearly dragged her to the floor after a press conference Tuesday night in Jupiter, Florida.
Terris was at the press conference because he was profiling Lewandowski and Trump's other advisers.
His piece, which touched on the incident with Fields, was titled, "Inside Trump's inner circle, his staffers are willing to fight for him.

Order Of Appearance:
5. Corey Lewandowski
4. Michelle Fields
3. Erik Wemple
2. Donald Trump
1. Ben Terris

Washington Post reporter Ben Terris on Friday dismissed a Breitbart News report that suggested a security guard, not Donald Trump's campaign manager, had grabbed a reporter's arm earlier this week after a press conference.
"I saw what I saw," Ben Terris told Erik Wemple, a media reporter for the Washington Post.
Ben Terris told Michelle Fields, the Breitbart News reporter involved in the alleged incident, that it was Corey Lewandowski, Donald Trump's campaign manager, who nearly dragged her to the floor after a press conference Tuesday night in Jupiter, Florida.
Ben Terris was at the press conference because he was profiling Corey Lewandowski and Donald Trump's other advisers.
His piece, which touched on the incident with Michelle Fields, was titled, "Inside Donald Trump's inner circle, his staffers are willing to fight for him.

Figure 3.2: ort of track changes to check w.

with full name entities on the linked document (Figure 3.3). For matching the strings of the entities we used case sensitive matching. If the method detects a full named entity that contains the single name entity in the tweet it replaces it on the tweet text. This approach produces satisfactory results. Around 20% of tweets in the data sets we processed include an embedded URL (linked to news articles in most cases) from which the full name of a Person entity could be successfully identified.

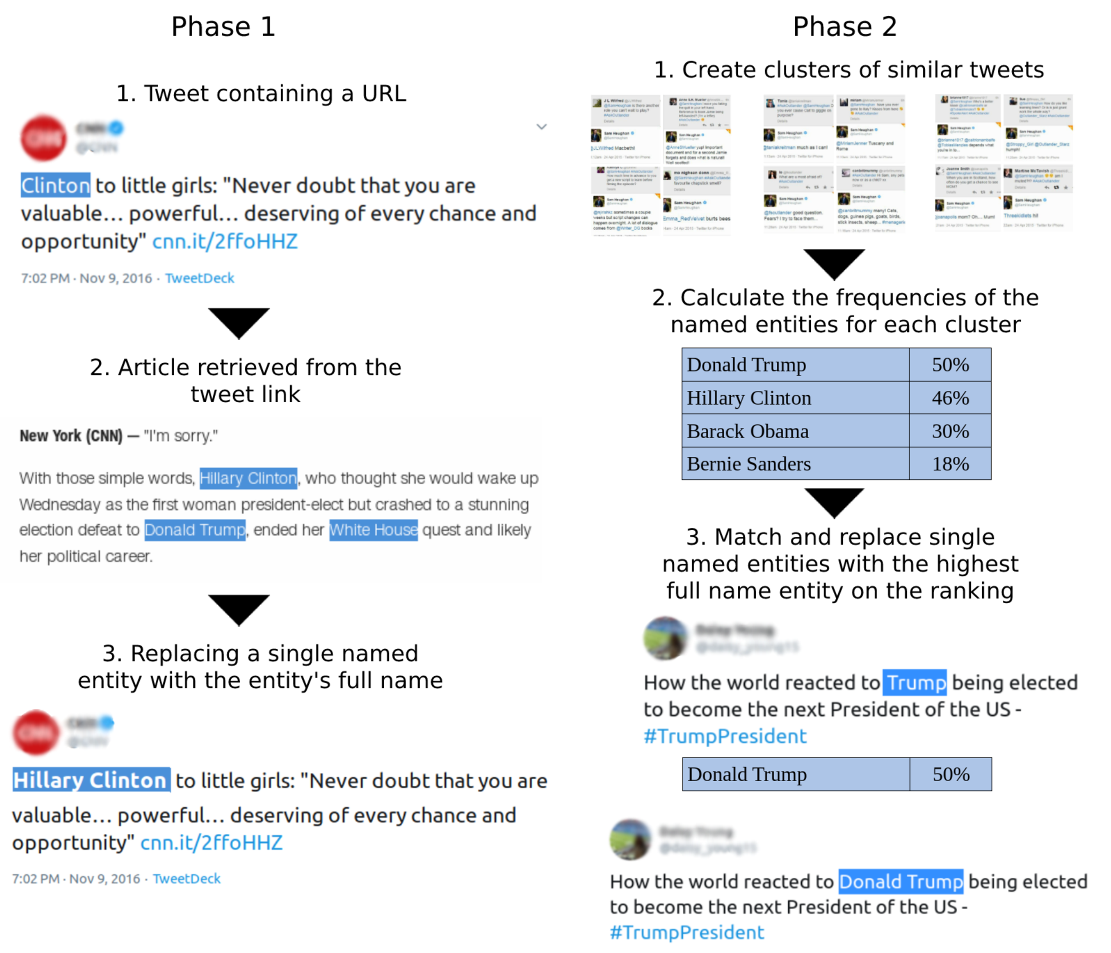
In the second phase, we cluster together tweets containing Person entities based on the string similarity of their full text (using a normalized Levenshtein distance metric [184] and the K-medoids [128] clustering algorithm) and also tweets sharing one or more hashtags, creating a 'super document' representing each cluster. For each cluster we create rankings with the frequencies of each full named entity appearing on the super document. Next we get every single named entity and scan the named entity ranking of the super document. If a full named entity contains the single named one we replace it. This approach further improves our disambiguation process. It is computationally efficient since it uses the named entities that were previously detected in the initial document processing stage of the NED pipeline.

3.2.2 Knowledge Graph Creation. Step 2 of the methodology.

For creating the knowledge graph we use the detected entities as nodes and entity co-occurrence in articles to form weighted edges. In determining edge weight, we consider the importance of each entity within each article. To do this, we assign a significance value to each detected entity in a given document as follows:

$$S_x(v) = \frac{tf(v,x)}{\sum_{v' \in V} tf(v',x)} \tag{3.1}$$

where v is the current entity (node), x is a piece of text (article, tweet), $tf(v,x)$ is the term frequency (the raw count of term v in text x), and V is the set of all entities in the current


Phase 1
1. Tweet containing a URL
2. Article retrieved from the tweet link
3. Replacing a single named entity with the entity's full name
Phase 2
1. Create clusters of similar tweets
2. Calculate the frequencies of the named entities for each cluster
3. Match and replace single named entities with the highest full name entity on the ranking

Clinton to little girls: "Never doubt that you are valuable… powerful… deserving of every chance and opportunity" cnn.it/2ffoHHZ

7:02 PM · Nov 9, 2016 · TweetDeck

New York (CNN) — "I'm sorry."

With those simple words, Hillary Clinton, who thought she would wake up Wednesday as the first woman president-elect but crashed to a stunning election defeat to Donald Trump, ended her White House quest and likely her political career.

Hillary Clinton to little girls: "Never doubt that you are valuable… powerful… deserving of every chance and opportunity" cnn.it/2ffoHHZ

7:02 PM · Nov 9, 2016 · TweetDeck

Donald Trump	50%
Hillary Clinton	46%
Barack Obama	30%
Bernie Sanders	18%

How the world reacted to Trump being elected to become the next President of the US - #TrumpPresident

Donald Trump	50%

How the world reacted to Donald Trump being elected to become the next President of the US - #TrumpPresident

Figure 3.3: Person entity disambiguation on tweets. Phase 1 Left. Tweet with a single word Person entity and the linked article containing the full name of the entity. We apply named entity recognition, match the single name entity on the tweet with the full name entity on the article and finally replace it on the tweet. Phase 2 Right. All tweets after phase 1 are being clustered based on their string similarity. Frequencies of full name entities are calculated for each cluster and rankings of them are created. Single named entities are being replaced by the highest full named entity in the frequency ranking, containing the single named entity.

document. Fig. 3.4 shows example output from application of (3.1).

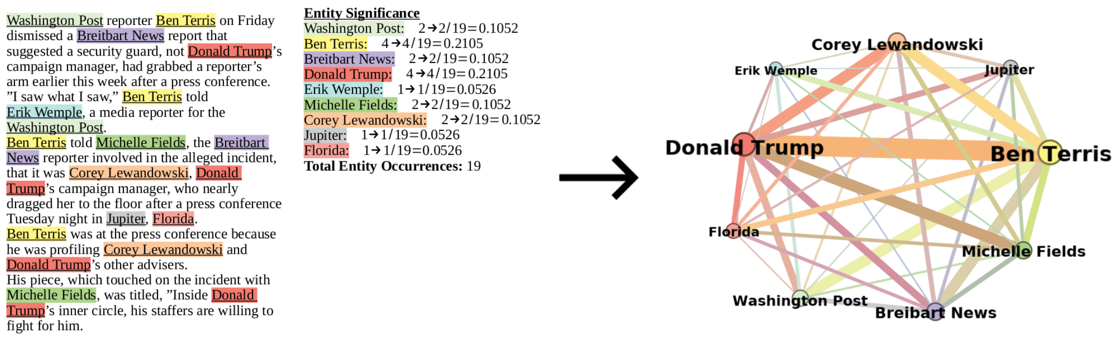
The contribution from a document d to the edge weight joining two entity nodes i, j is then given by:

$$w_d(i,j) = \begin{cases} S_d(i) + S_d(j) & \text{if } i,j \in V \\ 0 & \text{otherwise} \end{cases} \tag{3.2}$$

where V is the set of entities in the current document. Figure 3.4 demonstrates the knowledge graph created. The aggregation of such graphs from all articles of a given time period forms the overall knowledge graph, according to:

$$W(i,j) = \sum_{d \in D} w_d(i,j) \tag{3.3}$$

where D is the set of all documents.



Washington Post reporter Ben Terris on Friday dismissed a Breitbart News report that suggested a security guard, not Donald Trump's campaign manager, had grabbed a reporter's arm earlier this week after a press conference. "I saw what I saw," Ben Terris told Erik Wemple, a media reporter for the Washington Post.
Ben Terris told Michelle Fields, the Breitbart News reporter involved in the alleged incident, that it was Corey Lewandowski, Donald Trump's campaign manager, who nearly dragged her to the floor after a press conference Tuesday night in Jupiter, Florida.
Ben Terris was at the press conference because he was profiling Corey Lewandowski and Donald Trump's other advisers.
His piece, which touched on the incident with Michelle Fields, was titled, "Inside Donald Trump's inner circle, his staffers are willing to fight for him.

Entity Significance

Washington Post:	2→2/19=0.1052
Ben Terris:	4→4/19=0.2105
Breitbart News:	2→2/19=0.1052
Donald Trump:	4→4/19=0.2105
Erik Wemple:	1→1/19=0.0526
Michelle Fields:	2→2/19=0.1052
Corey Lewandowski:	2→2/19=0.1052
Jupiter:	1→1/19=0.0526
Florida:	1→1/19=0.0526
Total Entity Occurrences:	19

Figure 3.4: Entity significance calculation and the knowledge graph generated from them. The size of the nodes corresponds to their weighted degree and the size of the edges to their weight. The color of each node corresponds to the color of each entity on the left.

3.2.3 Monitoring Named Entity Weighted Degree. Step 3 of the methodology.

In this work we explore detection of news events using as an indicator the significant increase of the weighted degree of an entity node. Depending on what application we are working on we are grouping incoming documents on time slots with varying length. The length of those time slots could be from seconds to days, months or even years. For example on a stock exchange application where we want to detect an event as soon as possible we could consider to use a time slot of minutes or hours. We then form the knowledge graph we described on the previous section for each time slot by using the documents published or created on that time slot and calculate the weighted degree of each node/named entity. After that we create time series of each one of the detected named entities on the graph monitoring their weighted degree over the time slots we have defined. For example the *All The News* data set is discretised into 13 one day blocks from 14/6/2016 to 26/6/2016. The *FA Cup final* data set is discretised into one minute blocks spanning the duration of the game. A sequence of knowledge graphs is created from all time slots in each data set. Then weighted degree time series are created for all entity nodes from the graph sequence. By monitoring the evolution of weighted degree for all nodes over time, our intention is to detect events based on changes in network structure, revealed in entity time series. To detect large changes we calculate the first differences of the time series to remove trends, then we calculate the mean and standard deviation from a sliding window of X days. A "peaking entity" is then defined as an entity node whose weighted degree exceeds a threshold of Y standard deviations away from the rolling mean.

Depending on what application the user wishes to develop, we can vary X and Y to consider larger/smaller events or longer/shorter timescales. Setting a higher standard deviation threshold enables the method to detect fewer and more "important" events in terms of how popular the related entity became on the news stream and how much its popularity increased or decreased during consecutive time slots. With a lower standard deviation threshold the methodology detects more events as it becomes more sensitive on less "important" events. With a bigger sliding window our methodology is able to detect events with a broader duration over time, for example election periods. A smaller sliding window is able to detect events with a narrower duration such as sport or public events.

Figures 3.6, 3.5, 3.7, 3.8 and 3.9 illustrate some examples of the weighted degree time series analysis we described, using a window of five time slots ($X = 5$) and a threshold distance from the mean of 2 and 4 standard deviations ($Y = 2, Y = 4$). Those examples are from the Brexit referendum that took place in United Kingdom on 2016 and the presidential elections that took place in the United States of America on 2016.

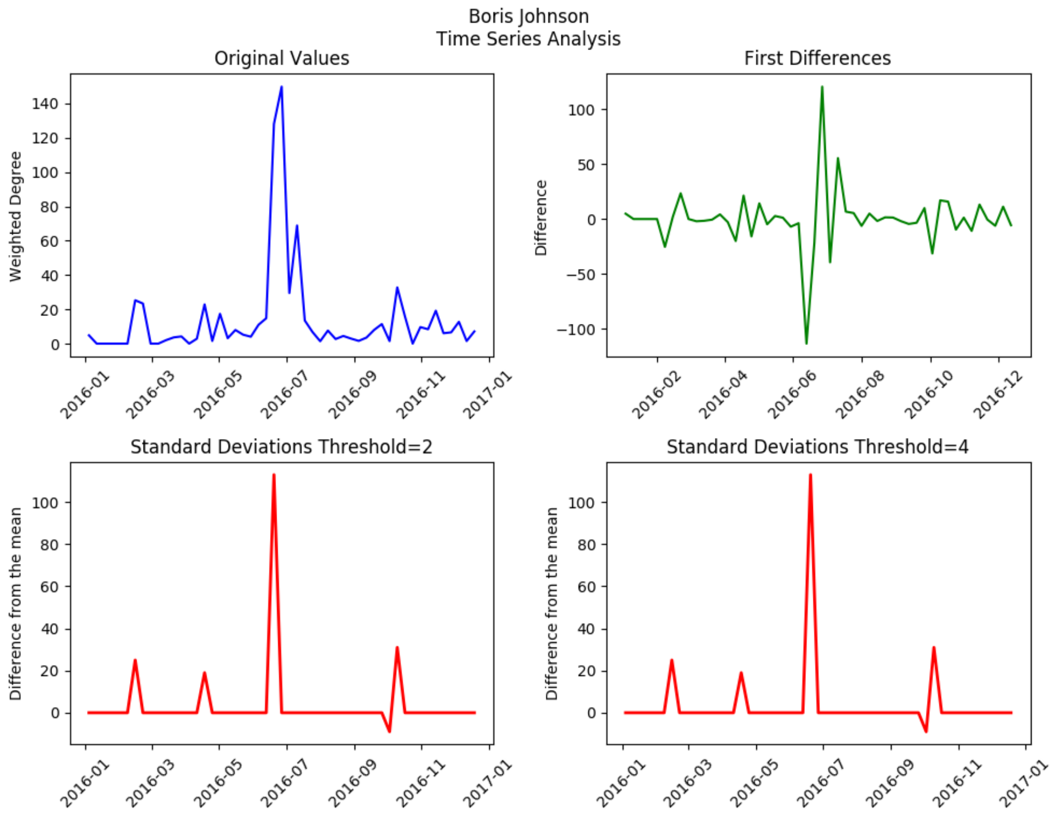


Figure 3.5: Weighted degree time series for the Boris Johnson entity. The detected peak occurs when the “Brexit” referendum took place.

3.2.4 Noun Phrase Detection on Document Containing Peaking Entities. Step 4 of the methodology.

After detecting the peaking entities for each time slot we gather all the documents that contain them. That way for each time slot we have a collection of documents that contain only peaking entities. This filters out documents that are not relevant to the trending topics we have detected. Doing so gives a substantial improvement in the performance of our method in a quantitative and qualitative manner since filtering out documents that do not contain peaking entities, significantly prunes the amount of data our system has to process and improves the quality of the results by removing noisy input. Each document collection then contains documents that potentially describe

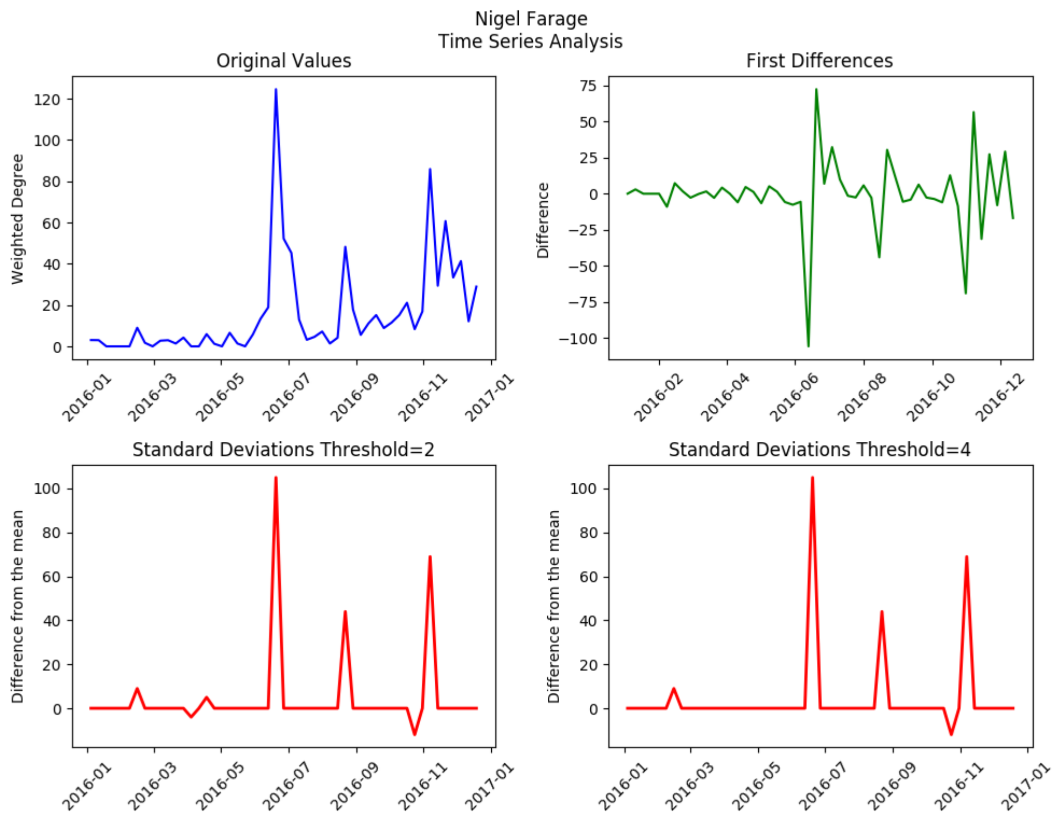


Figure 3.6: Weighted degree time series for the Nigel Farage entity. The first detected peak occurs when the "Brexit" referendum took place, the second highest peak after November occurs when the US Presidential elections took place.

one or more news events. For detecting which documents contain peaking entities a single string match is enough although for performance reasons we used indexes where named entities were the keys and ids of the documents containing them were the values. Finally we apply the ToPMine algorithm [47] to detect nouns and noun phrases on the documents that contain peaking entities. This algorithm combines a novel phrase mining framework to divide a document into single and multi word phrases, and a new topic model that operates on the induced document partition. This approach detects high quality topical phrases with low computational cost in a variety of data sets including academic papers, abstracts, reviews, and news articles, which makes it very useful for our application. Figure 3.10 presents an example of this algorithm.

3.2.5 Entities and Noun Phrases Knowledge Graph Creation Step 5 of the methodology.

On this step a second generation of graphs is created for each time slot using only the documents that contain peaking entities. We use named entities as well as nouns and noun phrases that were detected on the previous sections. This kind of graphs are referred to as KeyGraphs [148]. Our working assumption here is that by using named entities combined with nouns and noun phrases we will be able to produce comprehensive descriptions of the detected events.

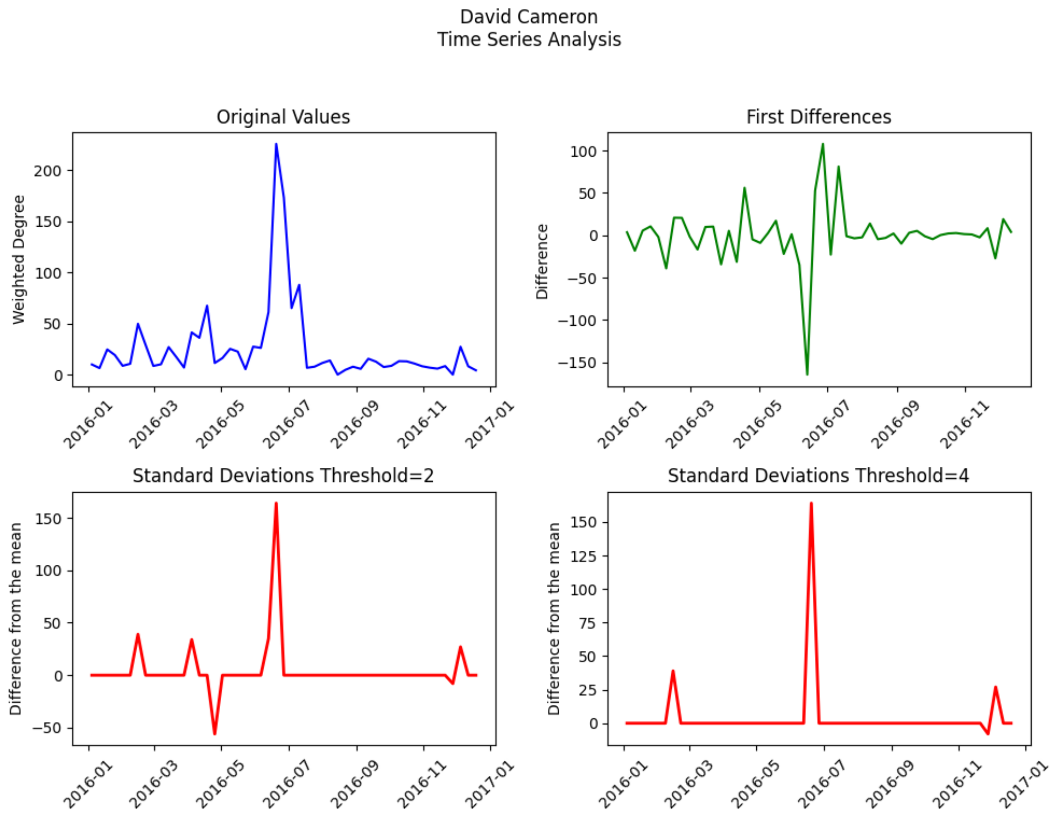


Figure 3.7: Weighted degree time series for the David Cameron entity. The detected peak occurs when the "Brexit" referendum took place.

3.2.6 Community Detection and Topic Summarization. Step 6 of the methodology.

The next step of our methodology is to extract and summarize the detected events for the filtered collections of documents for each time slot. For each created keygraph we apply the Louvain community detection algorithm [21]. This algorithm detects high modularity partitions of large networks in short time and unfolds a complete hierarchical community structure for the network, thereby giving access to different resolutions of community detection. Each detected community of nouns, noun phrases and named entities is considered as a candidate for an event. We create a bag-of-words summary of the event, sorting each bag of words using the weighted degree of each node to generate an easily interpreted synopsis of the event. The summary is a ranking of these entities based on their weighted degree on the final knowledge graph for each time slot of the analysis pipeline, representing how often an entity is mentioned in the document stream and how often it co-occurs in the same text as other frequently mentioned entities. In Figure 3.11 on the left we present an example KeyGraph generated from documents mentioning peaking entities on the 24$^{\text{th}}$ of June 2016 with labels added based on manual inspection of the communities and on the right we present the entities and noun phrases with the highest weighted degree on each community from the same KeyGraph, which together describe the detected event. Tables 3.1 and 3.2 present the detected topics of our methodology for each predefines time slot.

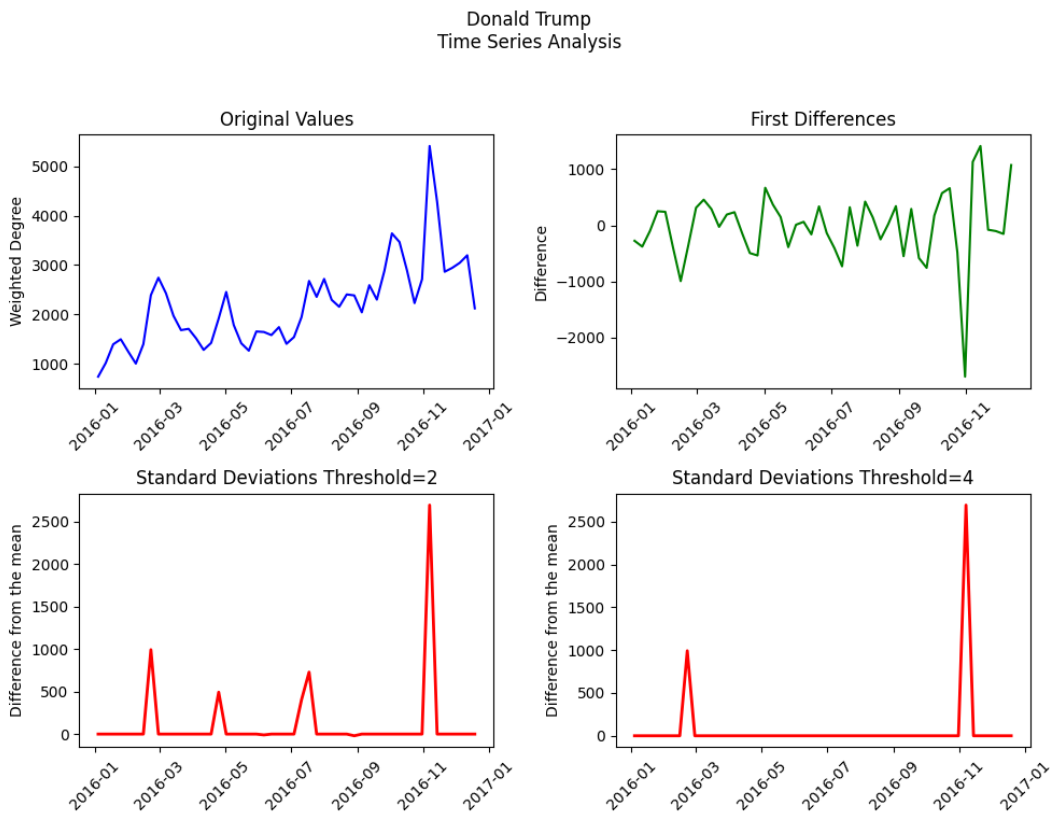


Figure 3.8: Weighted degree time series for the Donald Trump entity. The detected peaks occur when the presidential elections in USA took place and on events such as when the election debate took place.

3.2.7 Sentiment and Emotion Extraction. Step 7 of the methodology.

On the final step of our methodology, we augment the factual summarisation that was generated on the previous step with a simple affective judgement of the event using sentiment analysis and emotion detection. Effectively, this step seeks to determine whether the event is seen as 'good' or 'bad' by the authors contributing to the news stream and what emotions prevail on the discourse about this event. This part of the methodology (Step 7 in Figure 3.1) operates by determining the sentiment and emotions in the text describing the event. To do this, all the documents that contain at least one of the entities or noun-phrases given in the event summary are gathered in a collection. Sentiment and emotion analysis is then applied to all documents in the collection to mine the overall sentiment and emotion distribution of the user crowd that posted documents about the event.

For sentiment analysis, the VADER model [76] was used, since it addresses many of the challenges of sentiment analysis in documents coming from micro-blogging platforms such as Twitter, Reddit or Facebook. Key challenges arise from the small size of the text, widespread use of abbreviations, informal language and slang, as well as typographic mistakes and sarcasm. Manual comparison of the VADER model with several other algorithms on a sample of tweets showed VADER to give the best performance.

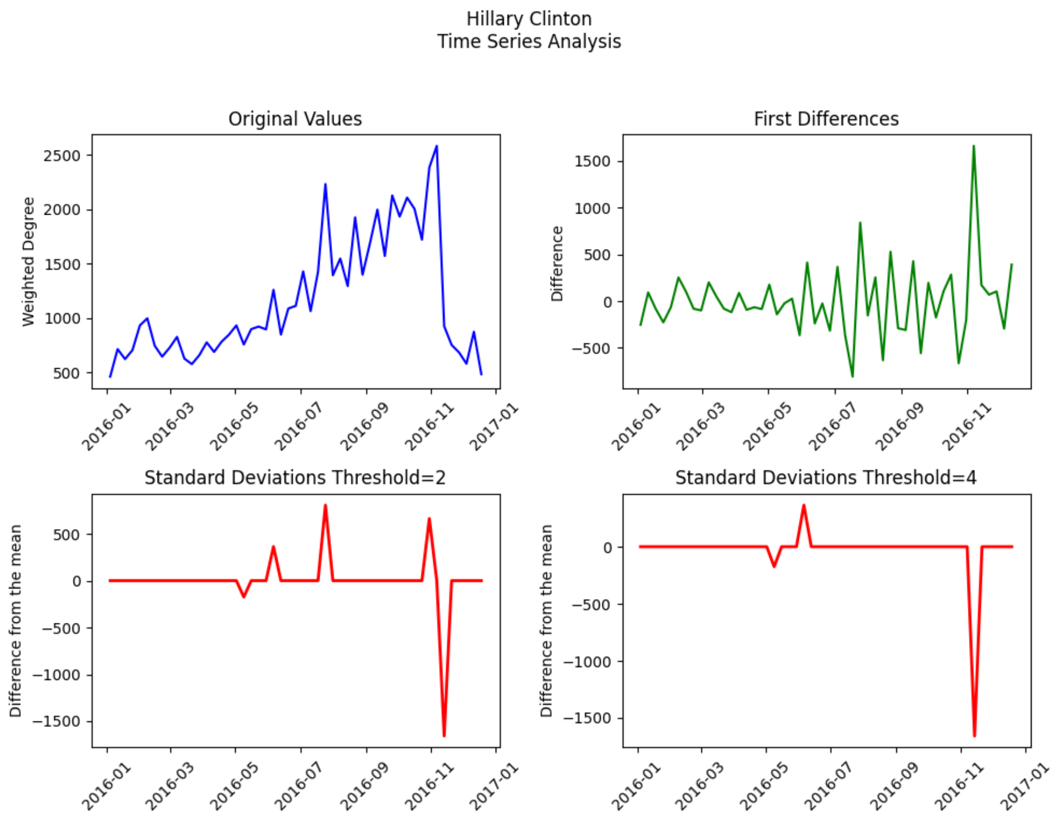
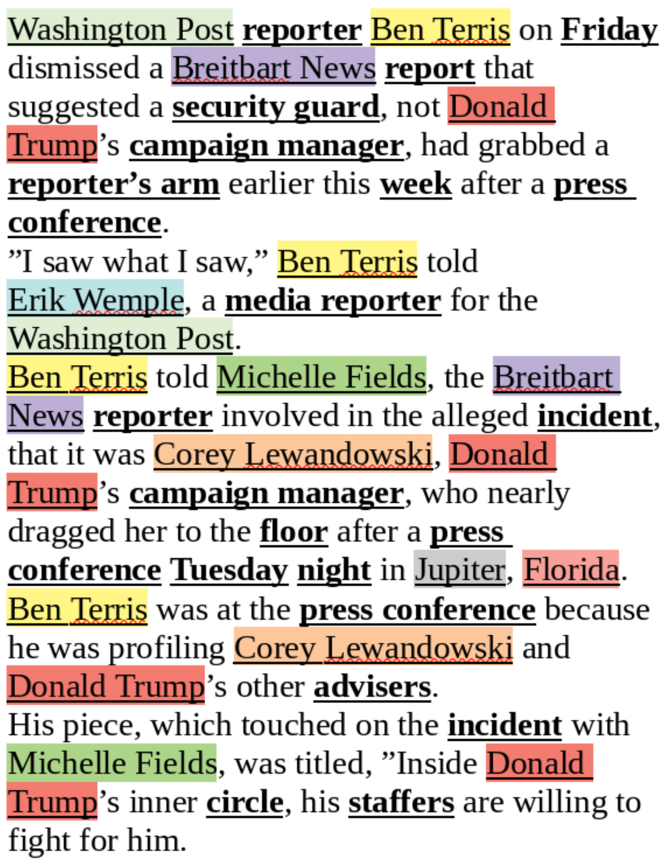


Figure 3.9: Weighted degree time series for the Hillary Clinton entity. The detected peaks occur when the presidential elections in USA took place and on events such as when the election debate took place.

For each tweet, VADER was applied to tweet text to calculate a sentiment polarity value, using the compound metric of the model to reflect how positive, neutral or negative the expressed sentiment is in the whole tweet. Polarity values are bounded between -1 for the most extreme negative sentiment and +1 for the most extreme positive sentiment. This gives a very simple unidimensional measurement of sentiment for a given sentence (or tweet). If the compound sentiment polarity score is denoted s, here we consider a tweet as positive for $s \geq 0.05$, negative for $s \leq -0.05$, or otherwise neutral ($-0.05 \leq s \leq 0.05$).

For emotion detection the text2emotion classifier [9] was used because of its performance on data sets with large volume of documents. The method consists of three main steps. In the first step, each document is cleaned by removing stop words and other unwanted textual parts and then applying natural language processing techniques such as part of speech tagging and syntactic analysis. In the second step, the method detects words in the text that express emotions or feelings, then it categorizes each word based on the feeling it represents. Finally, it stores the counts of each emotion-related word of the document. In the third step, a dictionary is generated for each input document with keys for the five emotions (Fear, Happiness, Sadness, Surprise and Anger) and values for the percentage that each emotion describes the whole document which is calculated based on the counts of each emotion related word detected in the input text. The emotion with

[9] https://github.com/aman2656/text2emotion-library



Washington Post **reporter** Ben Terris on **Friday** dismissed a Breitbart News **report** that suggested a **security guard**, not Donald Trump's **campaign manager**, had grabbed a **reporter's arm** earlier this **week** after a **press conference**.
"I saw what I saw," Ben Terris told Erik Wemple, a **media reporter** for the Washington Post.
Ben Terris told Michelle Fields, the Breitbart News **reporter** involved in the alleged **incident**, that it was Corey Lewandowski, Donald Trump's **campaign manager**, who nearly dragged her to the **floor** after a **press conference** **Tuesday** **night** in Jupiter, Florida.
Ben Terris was at the **press conference** because he was profiling Corey Lewandowski and Donald Trump's other **advisers**.
His piece, which touched on the **incident** with Michelle Fields, was titled, "Inside Donald Trump's inner **circle**, his **staffers** are willing to fight for him.

Figure 3.10: Nouns and noun phrases (in bold and underlined) detected alongside named entities using the ToPMine algorithm.

the higher score on an emotion category could characterize the whole document. Alternatively for each document we can demonstrate the emotions of it by presenting the percentages of all five emotions that were detected on the document.

3.3 Event Detection Evaluation

To evaluate our method we compare performance against several state of the art techniques described by Aiello et al [4]. The evaluation considers events detected on two data sets, one of newspaper articles and one from Twitter, as described above. On both data sets we applied tokenization pre-processing for all the methods, using the Twokenizer tool [121] to remove stop words and punctuation, and to compress redundant character repetitions. We used the following event detection methods:

- Latent Dirichlet Allocation - LDA
- Document-pivot topic detection - Doc-p
- Graph-based feature-pivot detection - GFeat-p
- Frequent Pattern Mining - FPM

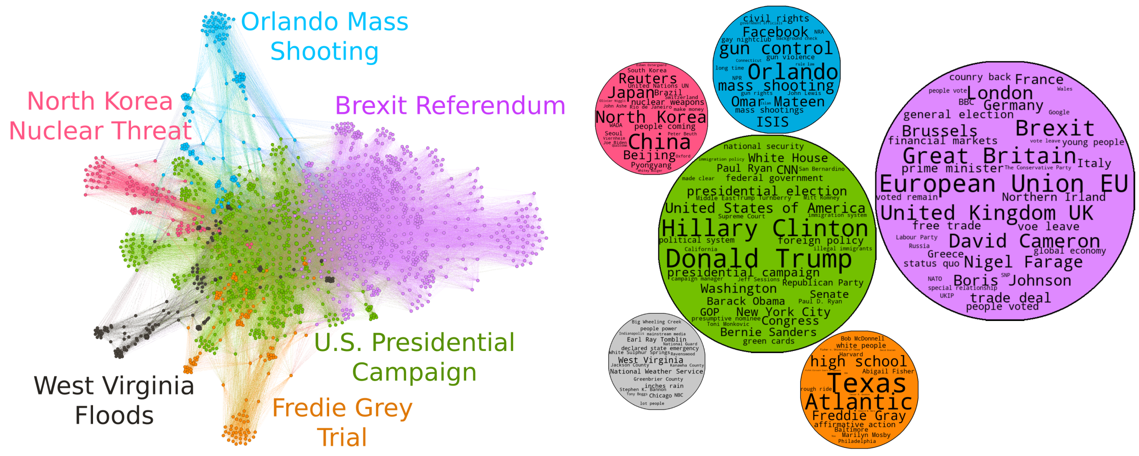


Figure 3.11: KeyGraph created using the peaking documents of the 24th of June 2016 (Left). The font size corresponds to the weighted degree of each entity/noun phrase node (Right).

Date	Topics Detected
19/6/2016	Orlando Mass Shooting, NBA Finals, Hamilton Musical, Brexit Referendum
20/6/2016	Lewandovski Fired, Brexit Referendum, Orlando Mass Shooting, NBA Finals, Gun Control, Anton Yelchin Death, Donald Trump Las Vegas rally incident
21/6/2016	Donald Trump Nomination, Euro 2016 Football, Orlando Mass Shooting, U.S. Presidential Campaign, Brexit Referendum, Tesla Solar City
22/6/2016	U.S. Presidential Campaign, Brexit Referendum, Orlando Mass Shooting, Donald Trump Nomination
23/6/2016	Brexit Referendum, Barack Obama Executive Amnesty, U.S. Presidential Campaign, Donald Trump Nomination, Gun Control, Abigail Fisher University admission Trial, Freddie Gray Trial, Volkswagen Emissions Scandal
24/6/2016	Brexit Referendum, U.S. Presidential Campaign, Orlando Mass Shooting, Freddie Grey Trial, North Korea Nuclear Threat, West Virginia Floods
25/6/2016	Brexit Referendum, U.S. Presidential Campaign, Orlando Mass Shooting, Euro 2016 Football, Kanye West Music Video
26/6/2016	Brexit Referendum, U.S. Presidential Campaign, Pope Francis on Gay People, Gun control

Table 3.1: Topics Detected on the All The News data set

- Soft Frequent Pattern Mining SFPM
- BNgram
- K-Means clustering

3.3.1 Event detection methods

Latent Dirichlet Allocation (LDA)

LDA [20] is a generative probabilistic technique to model and summarize large document collections in an unsupervised way. In the context of topic detection, one of the common approaches is to assume that each document is relevant to a topic with a certain probability. The contents of a document is modeled with respect to terms that appear on it. In LDA, a three-level hierarchical Bayesian model is used, where each topic is characterized by a distribution over these terms.

Time Slot	Topics Detected
16:16	Kick off the game.
16:26	Chelsea scores a goal with Ramires
16:41	Kalou gets substituted
16:53	Mikel gets a yellow card
17:01	Mikel challenging yellow card
17:03	End of first half
17:18	Start of the second half
17:24	Chelsea scores a second goal
17:25	Lampard assists Drogba for the goal
17:36	Liverpool scores a goal
17:46	Save from Cech
17:56	Andy Caroll makes a header
18:09	Full time end of the game Chelsea wins the FA Cup

Table 3.2: Topics Detected on the FA Cup data set

This technique finds the relevance of documents to different topics and clusters text corpora into groups of documents, where each group represents the set of documents that are relevant to a topic. Document modeling assigns each document to a topic with some probability and this preserves essential statistical relationships that are useful for trending topic detection, classification, collaborative filtering, and summarization of text corpora.

Document-pivot topic detection (Doc-p)

The document-pivot method clusters documents based on textual similarity. The clustering of the documents is optimized using the Locality Sensitive Hashing (LSH) algorithm from [130]. The main steps of this method are:

- Online clustering of documents: Documents are vectorized using the tf-idf representation [145] and distances between them are calculated using cosine similarity. Using the LSH technique the distance between an incoming document and the best matching document from each cluster is calculated; depending on a distance threshold the incoming document then either joins an existing cluster or creates a new cluster.
- Clusters smaller than a given size threshold are filtered out.
- A $significance$ score is calculated for each cluster, such that clusters with less frequent words

get a higher score.

- The top n clusters based on their $significance$ score are returned as the algorithm output.

Graph-based feature-pivot detection (GFeat-p)

This feature-pivot approach forms a graph using the documents of the corpus, in which the nodes of the graph correspond to terms in the documents and the communities of the graph correspond to events. The method uses the approach in [121] to rank all terms in the corpus based on the observed likelihood of appearance compared to the corresponding likelihood in a reference corpus consisting of randomly collected tweets. The method then selects the terms with the higher ranking and creates a term graph. For extracting the events from the graph the Structural Clustering Algorithm for Networks (SCAN) [180] is used to produce communities of terms, with each community considered to represent a distinct topic.

Frequent Pattern Mining (FPM)

In contrast with feature-pivot methods similar to GFeat-p, FPM takes into account the simultaneous co-occurrences between more than two terms. This is accomplished using a transaction mining technique for topic detection called frequent item set mining [64], to determine which *items* are likely to co-occur. In the context of event detection in document streams, an item is a word mentioned in a document. The transaction is the document and the transaction set is the set of documents published in a specific time slot (here a one-day or one-minute block). The frequency with which any set of terms occurs in a given time slot is defined as item set support, and any item set that meets a minimum support threshold is called a pattern. The method applies highly scalable Frequent Pattern (FP) detection on each time slot and then ranks the Frequent Patterns in order to find the most relevant keyword set for each time slot. The detected item sets correspond to and describe the trending topics for that time slot.

Soft Frequent Pattern Mining (SFPM)

SFPM examines co-occurrence patterns between sets of terms with cardinality larger than two. It is similar to FPM, but less strict, as it does not require all terms in these sets to co-occur frequently. SFPM ensures topic cohesiveness by grouping together large subsets of the terms that co-occur frequently, but at the same time allowing terms that occur less frequently with the rest of the group. For that reason this method is called *Soft* FPM.

The SFPM technique works by maintaining a set of terms S, to which new terms are added in a greedy manner, according to how often they co-occur with the terms in S. The set of terms S is initialized with a term and is represented by a vector D_S of length n where n is the number of documents in the specific time slot. The i^{th} element of D_S denotes how many of the terms of S

occur in the i^{th} document. Each incoming term is represented with a similar vector D_t of length n where the i^{th} element of the vector is a binary indicator that represents whether the term t occurs in the i^{th} document. Finally the set S is expanded with *similar* terms until a similarity threshold is met between the set and the next term. The terms of each set are describing the detected topic. The number of topics is determined by the number of initial terms selected by the user.

BNgram

The BNgram method utilizes co-occurrence between n-grams instead of unigrams. The method introduces a new feature selection approach, which monitors the frequency of appearance of each term in every time slot to detect emerging events on a news stream. The method uses the $df - idf_t$ metric which introduces time to the classic $tf - idf$ score to penalize topics that contain terms appearing in previous time slots. In addition, a *boost* factor is considered to raise the score of proper nouns (persons, locations and organizations). After the calculation of the scores BNgram clusters together n-grams based on distances between n-grams or clusters of n-grams. For calculating the distances between n-grams a similarity metric is introduced, defined by the fraction of posts that contain two n-grams. Each n-gram is assigned to a separate cluster and then the "group average" hierarchical clustering algorithm [116] is applied to merge the closest pairs of n-gram clusters. The clustering is repeated until the similarity between the nearest un-merged clusters falls below a fixed threshold, producing the final set of topic clusters for the corresponding time slot. Finally, the clusters are ranked according to the highest $df - idf_t$ score, and each cluster represents a detected topic.

K-Means clustering

Finally we implemented a method using the K-Means clustering algorithm [98], to create clusters of documents, represented by $tf - idf$ vectors. To determine the number of clusters we ran the algorithm with an initial arbitrary size of 15 clusters and merged similar clusters based on a similarity threshold. Finally each document cluster corresponds to a detected topic and the terms with the highest $tf - idf$ score were selected to describe it.

3.3.2 Evaluation against the FA Cup Final tweet data set

This data set consists of tweets that were crawled during the FA Cup Final in 2012, between Chelsea and Liverpool, and was partitioned in one minute time slots. In a previous evaluation of topic detection methods [4], the authors of that study reviewed the published media report accounts of the events and chose a set of stories that were significant, time-specific, and well represented on news media to build a topic ground truth. The evaluation test was for each algorithm to recover the topic ground truth. Each ground truth topic consists of a set of *keywords* and a *headline* describing it. For the FA Cup Final data set, a total of 13 one-minute slots were annotated with one topic each.

We ran our NED method for the FA Cup data set and calculated the same evaluation metrics that were used in the earlier study [4]. The aim of this evaluation is for the various methods to generate topics that contain as many keywords as possible from each ground truth topic. Each method returns the top 2 detected topics, each one of them described by a set of n-grams.

Three metrics were used to evaluate the topic detection methods on this data set:

- *Topic recall*: Percentage of ground truth topics successfully detected by a method. A ground truth topic was considered to be successfully detected when an algorithm produced a topic with a set of keywords that contained all mandatory keywords in the ground truth topic.
- *Keyword precision*: Percentage of correctly detected keywords (as described above) from the total number of keywords for the detected topics matched to some ground truth topic in the time slot under consideration. The total precision of a method is computed by micro-averaging the individual precision scores over all time slots.
- *Keyword recall*: Percentage of correctly detected keywords from the total number of keywords for the ground truth topics that have been matched to some detected topic in the time slot under consideration. The total recall is similarly computed by micro-averaging.
- *Keyword F_Measure*: Keyword F_measure is a measure of the methods accuracy. It is calculated from the keyword precision and keyword recall of the method and it is their harmonic mean.

3.3.3 Evaluation against the All The News data set

To evaluate the different methods on the articles of the ALL The News data set we first created a ground truth set of events by manually annotating 1861 articles from an eight-day period from 19/6/2016 until 26/6/2016. For each article, we annotated the main topics and then created groups of articles based on the annotated topics. We also used articles from five previous days for the methods that required a time window of previous days to detect emerging trends (NED, BNgram). The time slot for this data set was one day. Multiple articles that discuss the same subject were considered to form a topic; for each day, the topics that were discussed by more than 4 articles were taken as the ground truth topics to be detected in the evaluation.

For the *All the News* evaluation, we used implementations of the LDA, Doc-p and BNgram methods from [4] and we also implemented the K-Means approach. For all the methods, we manually inspected which of the returned groups of n-grams (detected topics) matched the annotated ground truth topics. An example of the generated groups of n-grams generated by each method can be found in Table 3.5.

Method	T-REC	K-PREC	K-REC	K-F_Measure
LDA	69,23%	16,37%	68,29%	26.40%
Doc-p	76.92%	33.73%	58.33%	42.27%
GFeat-p	0%	0%	0%	N/A
FPM	30.77%	**75%**	42.86%	**54.54**%
SFPM	61.54%	23.36%	65.79%	34.47%
BNgram	76.92%	29.89%	57.78%	39.39%
NED	**84.61**%	24.74%	**79.31**%	37.71%

Table 3.3: FA Cup Final data set evaluation results

Method	Precision	Recall	F_Measure
LDA	40.8%	40.8%	40.8%
Doc-p	40.7%	21.9%	28.47%
BNgram	53.95%	38.04%	44.61%
K-Means	51.24%	35.32%	41.81%
NED	**66.98**%	**50.48**%	**57.57**%

Table 3.4: All The News data set evaluation results

To evaluate the topic detection methods we used two metrics:

- *Precision*: Percentage of ground truth topics successfully detected by each method within the total number of candidate topics returned.
- *Recall*: Percentage of ground truth topics successfully detected by each method within the total number of ground truth topics.
- *F_Measure*: F_measure is a measure of the methods accuracy. It is calculated from the precision and recall of the method and it is their harmonic mean.

3.3.4 Evaluation Results

Our proposed NED algorithm outperformed all of the comparator methods on most of the metrics that were calculated in both evaluations. Results for the *All The News* evaluation are presented in Table 3.4 and results for the FA Cup Final evaluation are presented in Table 3.3.

For the FA Cup Final data set which consists of Twitter posts, the NED method outperformed the rest of the methods in topic recall and keyword recall. NED did not perform as well as three of the methods (FPM, Doc-p and BNgram) in keyword precision. The reason for this is that our method produces a relatively large number of keywords to describe each topic and for that reason the keyword precision drops significantly. Restricting the number of keywords describing each topic can efficiently address this issue (data not shown) but it has an adverse effect on topic recall and keyword recall.

For the the *All The News* data set which consists of longer news articles, NED achieved substantially better precision and recall than all the other methods tested.

Method	Topic Description
LDA	European, brexit, Britain, new, Trump, vote, EU, political, leave, it's going, Clinton, UK, campaign, May, united, president, two, British, first, economic
Doc-p	independence, remain, interdependent, heralded, British, countervailing, brexit, institutionalists, Farage, proximate, pen's, EU, globalist, xenophobes, pollsters, unbridled, Hillary, globalism, foreshadowing, comeuppance
BNgram	people, Brexit, vote, European Union, two, United Kingdom, new, even, referendum, last, get, told, years, want, time, leave, political, British, May, back
K-Means	EU, Britain, European, brexit, trump, leave, vote, British, union, UK, Europe, United, referendum, voters, remain, economic, Cameron, Kingdom, trade, Farage
NED	European Union EU, Great Britain, United Kingdom UK, London, Brexit, David Cameron, Nigel Farage, Brussels, Boris Johnson, Germany, trade deal, prime minister, Northern Ireland, general election, vote leave, country back, special relationship, people vote, UKIP, vote leave

Table 3.5: Brexit Referendum topic descriptions

3.3.5 Event Summarisation

Although it is hard to quantify the quality of a given summary of an event or trending topic, we present some examples from the *All The News* evaluation that we believe qualitatively demonstrate the good performance of the NED method on this aspect of the event detection task. Table 3.5 gives the top n-grams generated by each method for the detected event corresponding to the Brexit referendum that occurred on 24th June 2016, one day after the referendum took place (note that news articles are typically published the day after the event takes place).

It is not possible to rank the performance of the different methods based on subjective interpretation, but we maintain that the NED method generates a comprehensible and intuitive characterisation of the Brexit event. Performance appears to be strong relative to other methods.

3.3.6 Disambiguation Evaluation

The disambiguation method was tested in a subset of the US2016 data set, containing 1700 tweets. From the 1700 tweets, 658 contained single named entities. The first phase of the method identified 171 tweets where their url link contained a full name entity matching with the single word entity of the tweet it originated. It was able to successfully disambiguate 125 single named entities from the total 171 single named entities that were replaced ($\sim$ 73%). The remaining 46 entities were replaced with a wrong full name entity. The second phase of the method successfully disambiguated 362 single named entities from the total of 401 single named entities that were replaced ($\sim$90%). The remaining 86 tweets containing single named entities could not be correctly disambiguated by our method. In total we were able to disambiguate $\sim$74% of single named entities in the evaluation data set. Table 3.6 presents a list of names and the corresponding replacements generated from the disambiguation methodology.

Original Entity	**Disambiguated Entity**
Trump	Donald Trump
Trump	Cuando Donald Trump
Trump	Trump Gabbard
Trump	America Trump
Trump	Pero Trump
Trump	Ivanka Trump
Trump	Donald Trumps
Trump	Trump Jennifer Rubin
Trump	Anti Trump
Trump	Team Trump
Trump	Con Trump
Trump	Slaoui Trump
Trump	Derochsaid Trump
Trump	Ste McGovern Retweeted Nick Harris Trump
Trump	Barron trump
Trump	Jenifer Rubin
Trump	Trump La fiscala de Nueva York
Trump	Donald Trump Hillary Clinton
Trump	Donald Trump Desde
Trump	Donald Trump Joe Biden George Floyd
Trump	Matthew Yglesian Trumps
Trump	Donald Trump Donald John Trump Trump
TRUMP	DONALD TRUMP
Trump	Donald Trump Nigel Farage Share
Trump	Matthew Yglesias Trumps
al	Comentarios Donald Trump
Clinton	Hillary Clinton
Clinton	Bill Clinton
al	Hillary Clinton Donald Trump
Simpson	Lisa Simpson
Biden	Joe Biden
Donald	Giro di Walter Donald
Donald	Cio Donald

Icahn	Carl Icahn
Kevin	Kevin Maguire
Gaga	GirlsNoteBook Gaga
Pence	Mike Pence
Jupp	Jupp Bordeaux
Jupp	Alain Jupp
Obama	Barack Obama
Obamas	President Barack Obamas
Obama	President Obama
Obama	Michelle Obama
Simpson	Los Simpson
Merkel	Angela Merkel
George	George TakeiBlasts Mike Pence
Sally	Sally Kohn
Williams	Lynn Wright Retweeted The Five Williams
Comey	James Comey
Gloria	Gloria Alvarez
Israel	David Israel
Bernie	DebraMessing Bernie
Victoria	Victoria de Trump
Kirk	HotlineJosh Kirk
Simpson	Els Simpson
America	America Donald Trump
GOD	GOD'S HAND
Assange	Julian Assange
President	President Trump
Bush	Billy Bush
Josh	HotlineJosh Kirk
Snoop	Snoop Dogg
Texas	Texas Senators Cruz
David	David Letterman

Table 3.6: Disambiguated entities of the US2016 data set.

3.4 Results

3.4.1 Sentiment Analysis on Elections Data sets

We applied our methodology to the US2012, US2016 and UK2019 data sets. Here our goal is to extract the sentiment of the crowd discussing about the outcome of each election. The time window was set to 1-day duration and the system successfully identified major events for all three data sets on the day following each election, when many Twitter users and news commentators were discussing the election results.

The sentiment analysis component of our pipeline was applied to groups of tweets associated with each election event (step seven of our methodology). Table 3.7 displays the percentage of tweets tagged as positive/neutral/negative for each election event, using the polarity ranges defined above. Sentiment is deemed negative for polarity $s \leq -0.05$ and positive for $s \geq 0.05$. This analysis shows that the majority of tweets associated with each election event were neutral (∼60% of all tweets). Manual inspection of these tweets suggests that the reason so many were neutral is that many Twitter users in these data sets were affiliated to, or retweeting tweets from, major news agencies like CNN, CBS, BBC, Reuters, etc. Such content tends to be moderate in its use of language and rarely expresses overt sentiment. Tweets like *"RT @SkyNewsBreak: Exit Poll: Barack Obama and Mitt Romney tied at 49% in swing state Virginia."* were retweeted many times, but contain no positive or negative sentiment in their language. This creates a large volume of neutral tweets. Table 3.7 also shows that in all cases, there are more positive than negative tweets. This may reflect user behaviour, for example, users may be more likely to tweet when they feel happy about the election outcome.

	Election Year		
	2012	2016	2019
Positive	34%	25.3%	22.9%
Neutral	57.8%	60.6%	60.4%
Negative	8.2%	14.1%	16.7%

Table 3.7: Sentiment scores for each detected election event.

Figures 3.12, 3.13 and 3.14 display the distributions of sentiment polarity scores for neutral, negative and positive tweets associated with each election event. Given that the majority of tweets are neutral and often comprise tweets (and retweets) from news platforms, Figures 3.12, 3.13 and 3.14 also provide the distributions of only the negative and positive sentiments for each event to permit easier inspection of the polarity scores. Looking across all the election events, there is a bimodal distribution of sentiment polarity scores, with apparent peaks visible in both the positive and negative sides of the scale. (If plotted, the neutral tweets would create a large central spike in all three distributions.) The bimodality in sentiment distributions suggests that different user groups responded differently to the election outcomes, as might be expected in two-party political contests.

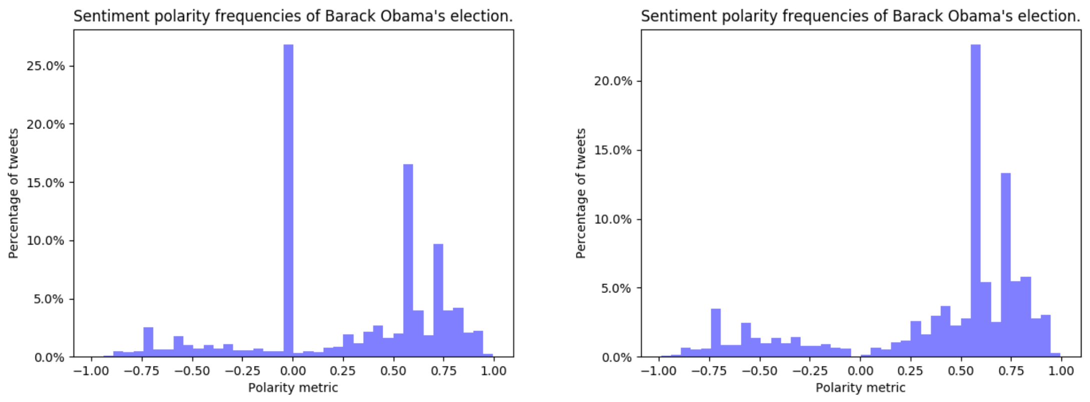


Figure 3.12: Sentiment distribution of the 2012 US presidential elections. About 34% of tweets about the presidents election were positive against only 8.2% of negative tweets. The percentage of neutral tweets is included on the left plot and excluded form the right one.

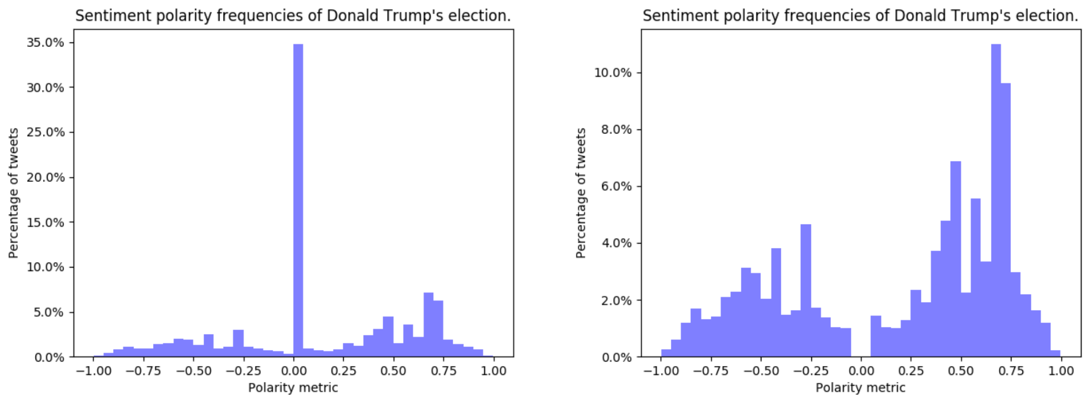


Figure 3.13: Sentiment distribution of the 2016 US presidential elections. About 25.3% of tweets about the presidents election were positive against 14.1% of negative tweets. The percentage of neutral tweets is included on the left plot and excluded form the right one.

For the US2012 data set we observe (Fig. 3.12) that the election of president Barack Obama was mostly received very well by the Twitter user population. The percentage of tweets with positive sentiment was 34% against only 8.2% of negative tweets. In 2012, President Obama won the election with 51.1% of the total votes and 332 electoral votes against Mitt Romney's 47.2% of the total votes and 206 electoral votes. This outcome in US2012 is perhaps more convincing and less controversial that the outcomes in US2016 and UK2019.

In the 2016 US election (Fig. 3.13), President Donald Trump won with 46.1% of the total votes and 304 electoral votes, against Hilary Clinton's 48.2% of total votes and 227 electoral votes. In our analysis of the US2016 data set, we infer that overall Trump's election was received positively by a big part of Twitter user population that tweeted about the election outcome, with 25.3% of the event-associated tweets being positive, but also that a substantial part of the user population (14.1%) reacted negatively.

Finally, in the UK General Election of 2019, the Conservative Party won the elections with

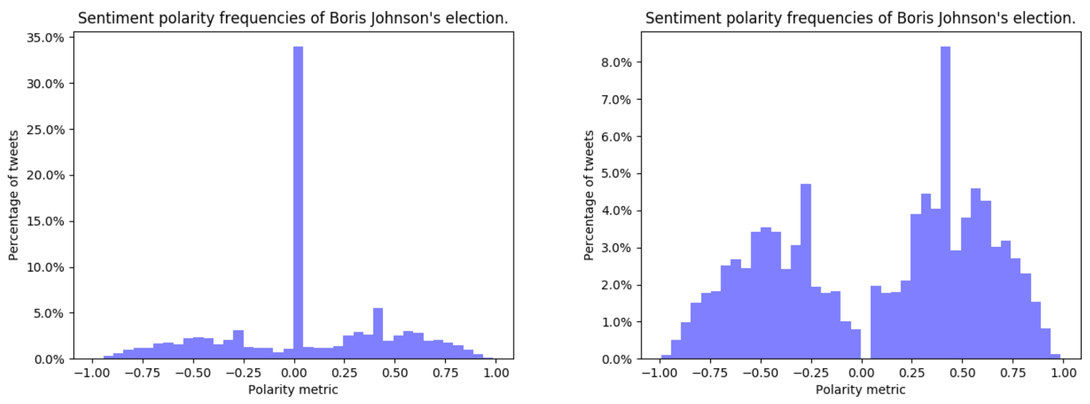


Figure 3.14: Sentiment distribution of the 2019 UK general elections. About 22.9% of tweets about the conservatives election were positive against 16.7% of negative tweets. The percentage of neutral tweets is included on the left plot and excluded form the right one.

43.6% of the total votes, against the Labour Party which came second with 32.2$ of the total votes. Tweets associated to this election event (Fig. 3.14) show the most balance between positive/negative sentiment, with 22.9% of tweets being positive and 16.7% of tweets being negative. It is tempting to speculate about the relationship between political polarisation during election campaigns and the balance of positive and negative sentiment expressed in social media discussions on Twitter; however, no firm conclusions can be drawn on this issue from our analysis and this topic is left for future work.

3.4.2 Monitoring the Covid-19 outbreak discourse on Tweeter

On this his work we analysed tweets from the Corona Virus Tweets data set [87] from January 1 until the end of June 2020. We used the Twitter API to monitor the real time Twitter feed for coronavirus related tweets using about 90 different keywords and hashtags that are commonly used while referencing the pandemic. Table 3.8 presents the keywords and hashtags that were used to acquire the ids of the tweets using the API. After acquiring the ids of the COVID-19 related tweets we once again used the Tweeter API to retrieve json representations of each tweet containing information such as the text of the tweet and the user posted it.

The main goal is to detect salient events on this time period, identify the topic of each event and extract their co-responding sentiments and emotions.

The NED methodology detected 130 peaking events from January to June of 2020. Here we will discuss 15 of the most significant events and provide their summaries as well as the sentiment and emotions detected for each one of them. To determine which events among the detected ones were more significant we took into consideration the weighted degree of each detected peaking entity and the top 15 named entities with the highest weighted degree. The events show how the coronavirus was spreading around the globe at that time as well as political and social developments that were caused by or heavily related to the pandemic. In Figure 3.15 a timeline of the events is presented

COVID-19 Keywords and Hashtags
corona, #corona, coronavirus, #coronavirus, covid, #covid, covid19, #covid19, covid-19, #covid-19, sarscov2, #sarscov2, sars cov2, sars cov 2, covid_19, #covid_19, #ncov, ncov, #ncov2019, ncov2019, 2019ncov,#2019ncov, pandemic, #pandemic #2019ncov, 2019ncov, quarantine, #quarantine, flatten the curve, flattening the curve, #flatteningthecurve, #flattenthecurve, hand sanitizer, #handsanitizer, #lockdown, lockdown, social distancing, #socialdistancing, work from home, #workfromhome, working from home, #workingfromhome, ppe, n95, #ppe, #n95, #covidiots, covidiots, herd immunity, #herdimmunity, pneumonia, #pneumonia, chinese virus, #chinesevirus, wuhan virus, #wuhanvirus, kung flu, #kungflu, wearamask, #wearamask, wear a mask, vaccine, vaccines, #vaccine, #vaccines, corona vaccine, corona vaccines, #coronavaccine, #coronavaccines, face shield, #faceshield, face shields, #faceshields, health worker, #healthworker, health workers, #healthworkers, #stayhomestaysafe, #coronaupdate, #frontlineheroes, #coronawarriors, #homeschool, #homeschooling, #hometasking, #masks4all, #wfh, wash ur hands, wash your hands, #washurhands, #washyourhands, #stayathome, #stayhome, #selfisolating, self isolating

Table 3.8: Keywords and Hashtags used to monitor tweets related with the COVID-19 pandemic.

with a brief description for each one of them given below. Those events were detected based on peaks on their weighted degree on specific days. In most cases there was not a significant change or peak on the number of tweets posted on that day only on the weighted degree on some named entities. However on four days there was a significant change on the number of tweets posted alongside with the peaks on the named entities. Those dates are the 9th of March 2020 where the number of tweets posted on one day doubled, the 17th of March 2020, the 19th of March of 2020 which was the day with the record amount of 7.65 million tweets posted, almost three times more than the previous day and the 18th of April 2020.

Hong Kong confirms first coronavirus death.

On February 4 of 2020 a 39 year old man in Hong Kong died from the coronavirus. He had underlying health problems and had visited Wuhan one month before his death where he came in contact with other infected individuals[10]. Some of the entities with the highest weighted degrees were *Hong Kong, coronavirus death, Wuhan* and *39-year-old-man* (Figure 3.16a). The sentiment for this event was negative and the average score on the Vader classifier was -0.375 (Figure 3.16b). The dominant emotion of the tweets discussing this event was fear with 48% of tweets reflecting this emotion followed by surprise with 31% (Figure 3.16c). The total number of tweets related to this event were 4434. The total number of tweets retrieved for this day were 157803. The event contained $\sim 2.8\%$ of the total tweets. The number of unique users discussing about this event was

[10]https://www.france24.com/en/20200205-china-coronavirus-world-health-organisation-epidemic-hong-kong-macau

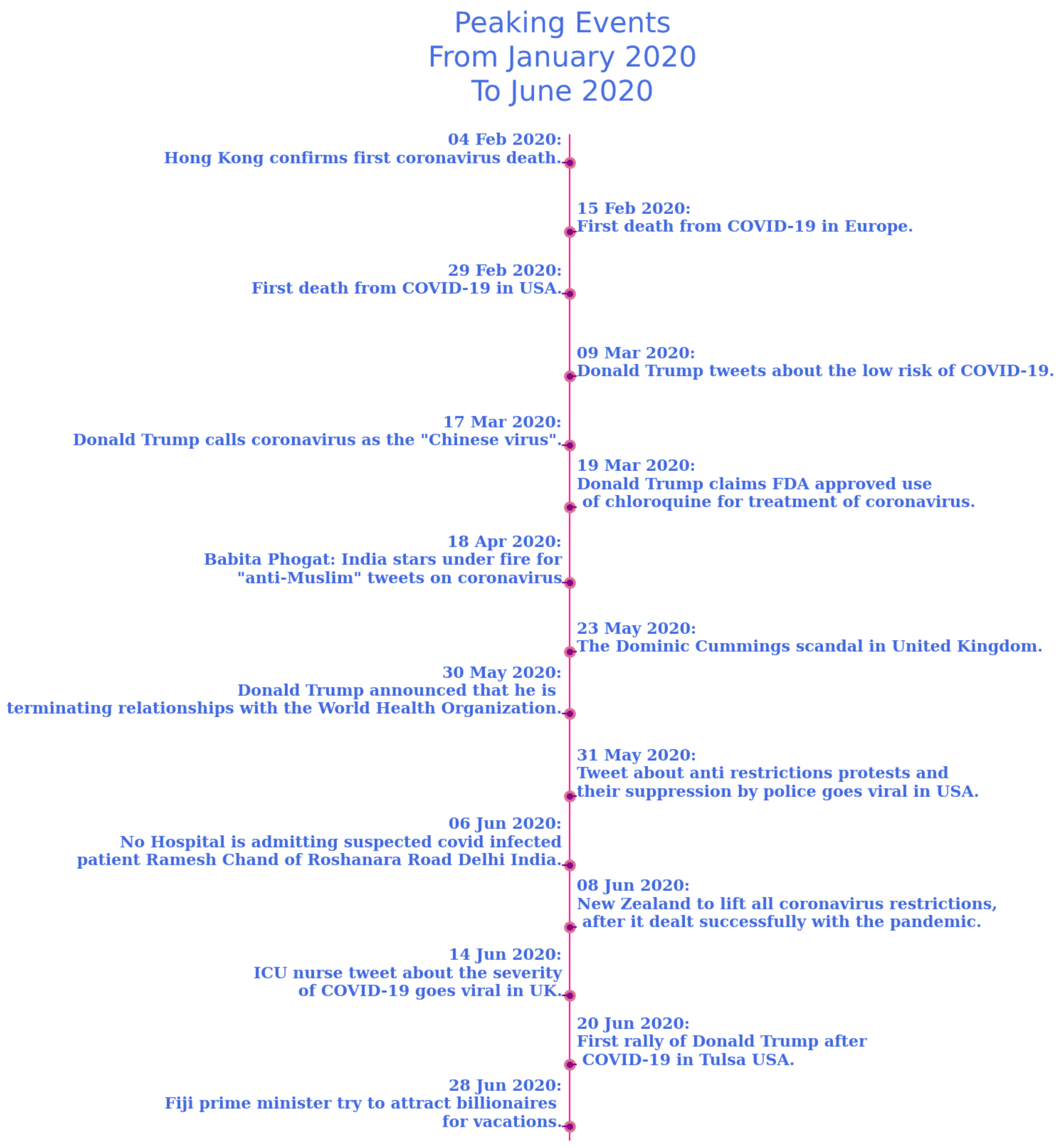

Peaking Events
From January 2020
To June 2020
04 Feb 2020:
Hong Kong confirms first coronavirus death.
15 Feb 2020:
First death from COVID-19 in Europe.
29 Feb 2020:
First death from COVID-19 in USA.
09 Mar 2020:
Donald Trump tweets about the low risk of COVID-19.
17 Mar 2020:
Donald Trump calls coronavirus as the "Chinese virus".
19 Mar 2020:
Donald Trump claims FDA approved use of chloroquine for treatment of coronavirus.
18 Apr 2020:
Babita Phogat: India stars under fire for "anti-Muslim" tweets on coronavirus.
23 May 2020:
The Dominic Cummings scandal in United Kingdom.
30 May 2020:
Donald Trump announced that he is terminating relationships with the World Health Organization.
31 May 2020:
Tweet about anti restrictions protests and their suppression by police goes viral in USA.
06 Jun 2020:
No Hospital is admitting suspected covid infected patient Ramesh Chand of Roshanara Road Delhi India.
08 Jun 2020:
New Zealand to lift all coronavirus restrictions, after it dealt successfully with the pandemic.
14 Jun 2020:
ICU nurse tweet about the severity of COVID-19 goes viral in UK.
20 Jun 2020:
First rally of Donald Trump after COVID-19 in Tulsa USA.
28 Jun 2020:
Fiji prime minister try to attract billionaires for vacations.

Figure 3.15: Timeline of the most significant events detected between January and June 2020.

2897 and the most shared tweet [11] had about one thousand retweets.

First death from COVID-19 in Europe.

The first death on Europe caused by COVID-19 was reported on the 15th of February 2020. It was an 80 year old man from China's Hubei province. He arrived in France on 16 January and was placed in quarantine in hospital in Paris on 25 January[12]. This was also the first casualty outside Asia. The entities with the highest weighted degrees on the knowledge graph were *Europe, First* and *coronavirus death* (Figure 3.17a). The sentiment for this event was very negative and the average score on the Vader classifier was -0.547 (Figure 3.17b). The emotions of fear and surprise were more prevalent on the tweets responding to this event with 51% fear and 44% surprise (Figure 3.17c). The total number of tweets related to this event were 3059. The total number of tweets retrieved for this day were 103540. The event contained $\sim$ 2.9% of the total tweets. The number of unique users discussing about this event was 2346 and the most shared tweet [13] had about two thousand retweets.

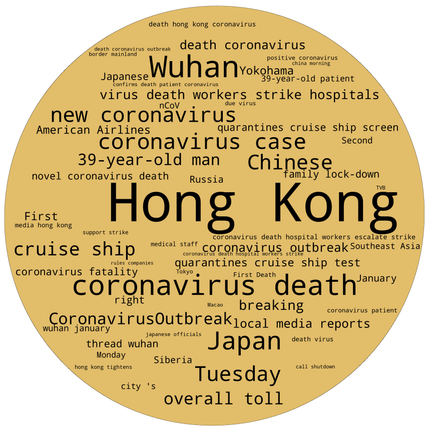
First death from COVID-19 in USA.

On the 29th of February 2020 the first death caused by COVID-19 was reported in Washington USA. The victim was a man in his 50s who had underlying health conditions. There was no indication that the patient had close contact with an infected person or a relevant travel history that would have exposed the patient to the virus[14]. The most significant entities on the knowledge graph were *Washington, the United States, COVID-19* and *health officials* (Figure 3.18a). The overall sentiment of the tweets associated with this event was negative with average score on the Vader classifier of -0.422 (Figure 3.18b). There were three emotions that were classified on the events tweets, Fear with 36%, Sadness with 31% and Surprise with 30% (Figure 3.18c). The total number of tweets related to this event were 11246. The total number of tweets retrieved for this day were 493812. The event contained $\sim$ 2.2% of the total tweets. The number of unique users discussing about this event was 8470 and the most shared tweet [15] had 58 retweets.

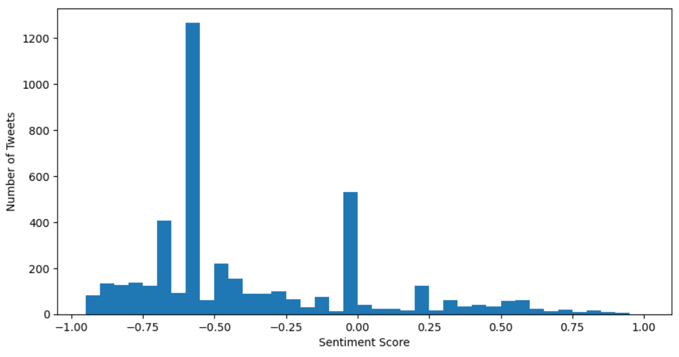
Donald Trump tweets about the "low" risk of COVID-19.

On the 9th of march 2020 president Donald Trump posted a tweet arguing that the corona virus is less dangerous than the common flu[16], [17]. The entities with the highest weighted degrees on the knowledge graph were *Trump, Coronavirus, Americans* and *CDC* (Figure 3.19a). The sentiment

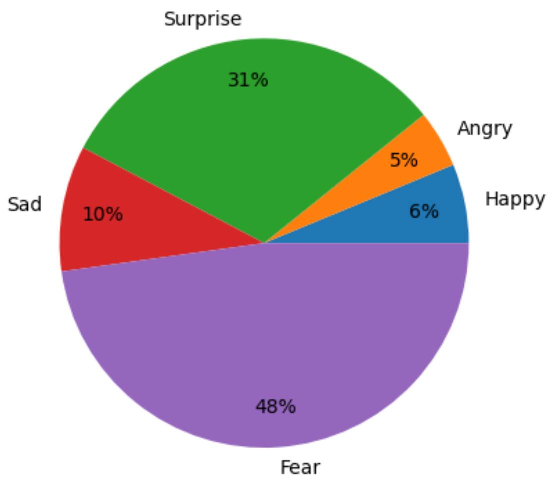
[11]https://twitter.com/spectatorindex/status/1224525608702464000
[12]https://www.bbc.com/news/world-europe-51514837
[13]https://twitter.com/BBCBreaking/status/1228622316835348480
[14]https://twitter.com/i/web/status/1233819054105878528
[15]https://twitter.com/i/web/status/1233820487417057281
[16]https://www.mediaite.com/opinion/10-trump-tweets-from-march-2020-that-have-aged-very-poorly-one-year-after-covid-was-declared-a-pandemic/
[17]https://www.vox.com/policy-and-politics/2020/3/9/21171582/coronavirus-trump-tweets-stock-market-denial



(a)

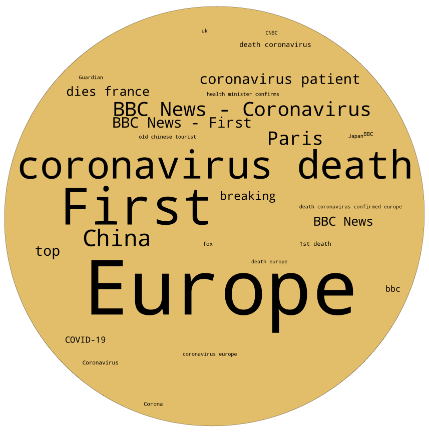


(b)

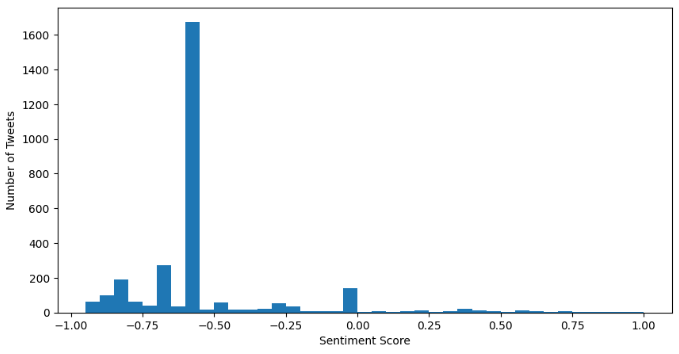


(c)

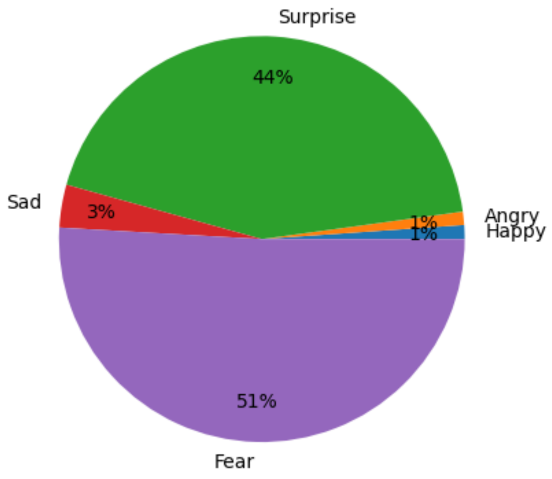
Figure 3.16: Event summary word-cloud, sentiment distribution and emotion percentages of tweets discussing the first death from COVID-19 in Hong Kong. Date: 4/2/20



(a)

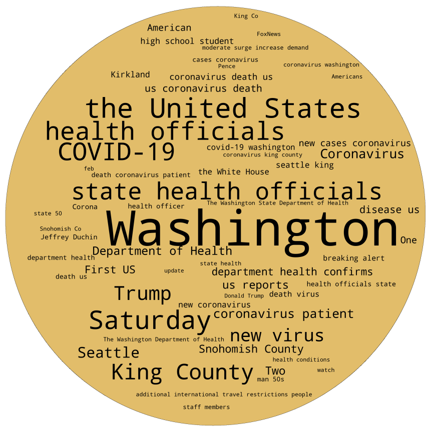


(b)

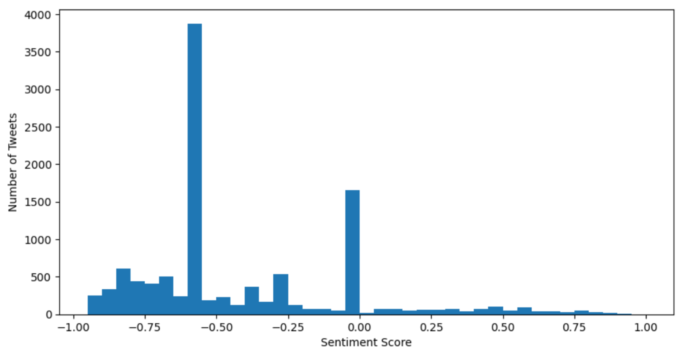


(c)

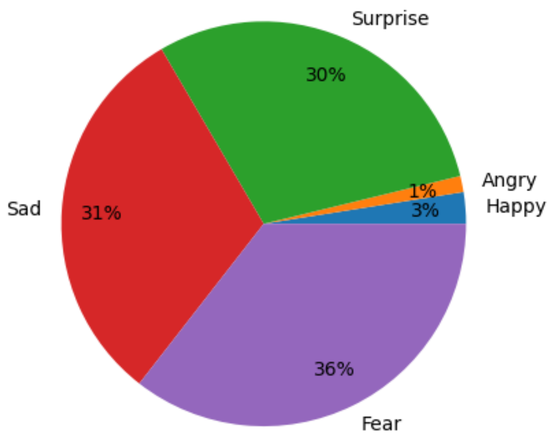
Figure 3.17: Event summary word-cloud, sentiment distribution and emotion percentages of tweets discussing the first death from COVID-19 in Europe. Date: 15/2/20



(a)



(b)



(c)

Figure 3.18: Event summary word-cloud, sentiment distribution and emotion percentages of tweets discussing the first death from COVID-19 in USA. Date: 29/2/20

of the responding tweets to this event is leaning to negative but they are polarized. The average sentiment score is -0.224 (Figure 3.19b). The main emotions for this event were Surprise with 36%, Fear with 35% and Sadness with 20% (Figure 3.19c). The total number of tweets related to this event were 105719. The total number of tweets retrieved for this day were 913033. The event contained $\sim$ 11.5% of the total tweets. The number of unique users discussing about this event was 63864 and the most shared tweet [18] had 63 retweets.

Donald Trump call coronavirus as the "Chinese virus".

On the 17th of March the US president posted on Twitter: "The United States will be powerfully supporting those industries, like Airlines and others, that are particularly affected by the Chinese Virus. We will be stronger than ever before!"[19] The World Health Organization advised that such terms should be avoiding since they discriminate and stigmatise Chinese citizens in the US. Some

[18]https://twitter.com/PalmerReport/status/1236957906362302464

[19]https://www.theguardian.com/world/2020/mar/17/trump-calls-covid-19-the-chinese-virus-as-rift-with-coronavirus-beijing-escalates

of the entities with the highest weighted degrees were *Chinese, China, Wuhan* and *Trump* (Figure 3.20a). The overall sentiment about this event was again leaning to negative but there is still a significant amount of tweets that are positive about this event. The average sentiment of the event is -0.178 (Figure 3.20b). The most dominant emotions are Surprise with 31%, Fear with 30% and sadness with 24% (Figure 3.20c). The total number of tweets related to this event were 89008. The total number of tweets retrieved for this day were 2470369. The event contained $\sim$ 3.6% of the total tweets. The number of unique users discussing about this event was 70199 and the most shared tweet [20] had 66 retweets. Donald Trumps tweet was removed since at the time of retrieving the tweets of our data set his account was banned. For that reason his tweet was not part of our analysis, but the user can now refer to his tweet [21] since his account was unsuspended.

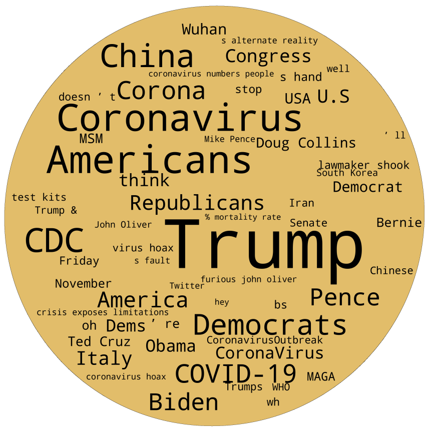
Donald Trump claims that FDA approved the use of chloroquine for treatment of COVID-19.

On the 19th of March US President Donald Trump claimed during a White House briefing on Thursday that the American Food and Drug Administration had approved the "very powerful" drug chloroquine to treat coronavirus. He claimed that the drug have shown "very encouraging early results" and that it could be available "almost immediately"[22]. The entities with the highest weighted degrees on the knowledge graph were *Trump, COVID-19, FDA* and *Chloroquine* (Figure 3.21a). The sentiment about this event is negative with average Vader score -0.164 (Figure 3.21b). The dominant emotion of the event is Surprise with 46% followed by Fear with 24% and Sadness with 18% (Figure 3.21c). The total number of tweets related to this event were 474181. The total number of tweets retrieved for this day were 7649110. The event contained $\sim$ 6.1% of the total tweets. The number of unique users discussing about this event was 238094 and the most shared tweet [23] had 17263 retweets.

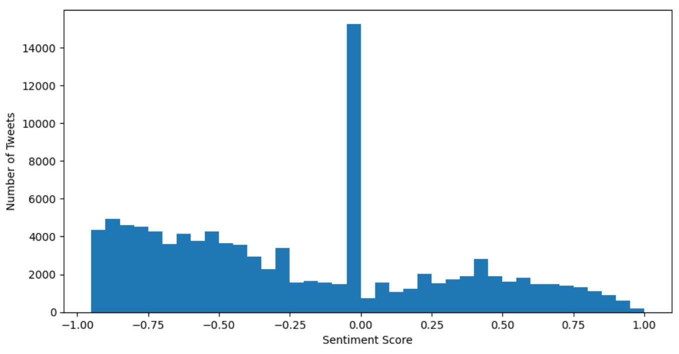
Babita Phogat: India stars under fire for "anti-Muslim" tweets on coronavirus.

An international wrestler and the sister of a Bollywood star have been criticised for tweets blaming Muslims over the spread of coronavirus in India[24]. The hashtag #SuspendBabitaPhogat became viral on the 18th of April 2020 calling on Twitter to ban wrestler Babita Phogat. The entities with the highest weighted degrees on the knowledge graph were *India, Corona, COVID-19* and *Muslims* (Figure 3.22a). The overall sentiment of the tweets discussing this event was slightly positive with average Vader score 0.0542 (Figure 3.22b). The emotion of most tweets of this event was Fear with 41% followed by Sadness with 23% and Surprise with 22% (Figure 3.22c). The total number of tweets related to this event were 73689. The total number of tweets retrieved for this day were

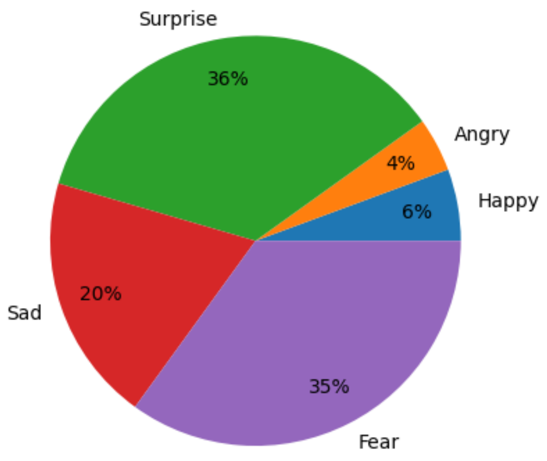
[20]https://twitter.com/i/web/status/1239739789424771073
[21]https://twitter.com/realDonaldTrump/status/1239685852093169664
[22]https://edition.cnn.com/2020/03/19/politics/fact-check-chloroquine-trump-fda/index.html
[23]https://twitter.com/i/web/status/1240643730421559297
[24]https://www.bbc.com/news/world-asia-india-52320900



(a)

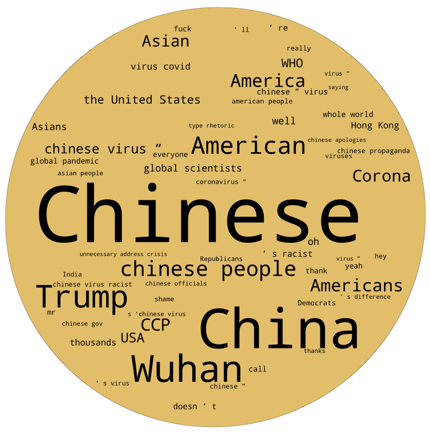


(b)

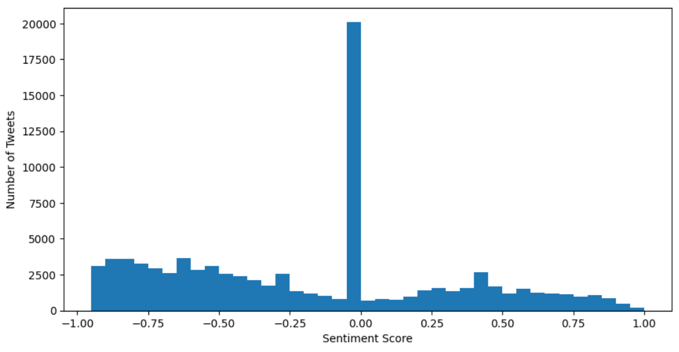


(c)

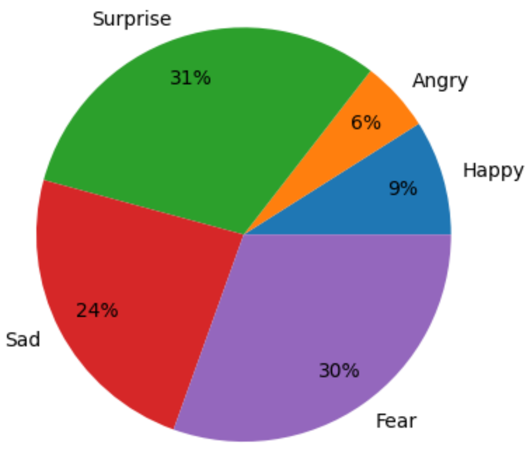
Figure 3.19: Event summary word-cloud, sentiment distribution and emotion percentages of tweets discussing Donald Trump's tweet about the "low" risk of COVID-19. Date: 9/3/20



(a)



(b)



(c)

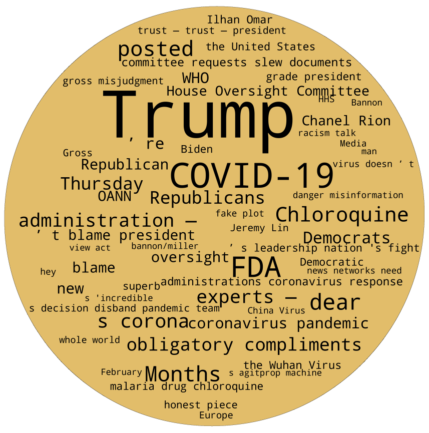
Figure 3.20: Event summary word-cloud, sentiment distribution and emotion percentages of tweets discussing Donald Trump's tweet calling corona virus as the "Chinese virus". Date: 17/3/20

1797517. The event contained $\sim$ 4.1% of the total tweets. The number of unique users discussing about this event was 50233 and the most shared tweet [25] had 785 retweets.

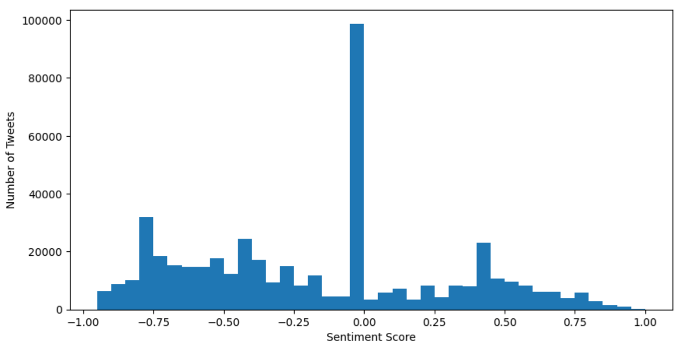
The Dominic Cummings scandal.

On the 23rd of May 2020 an event about the British political strategist Dominic Cummings have emerged. The scandal was about a journey that Cummings, then-chief adviser of Prime Minister Boris Johnson, and his family made from London to County Durham after the start of a national lockdown in March 2020 while they were experiencing symptoms of COVID-19[26]. The entities with the highest weighted degrees on the knowledge graph were *Cummings, cabinet mugs adapt, Johnson, restrictions* and *part story* (Figure 3.23a). The overall sentiment is slightly negative with an average Vader score of -0.022 (Figure 3.23b). The most significant emotion on this event is Fear with 41%, followed by Surprise with 29% and Sadness with 19% (Figure 3.23c). The total number

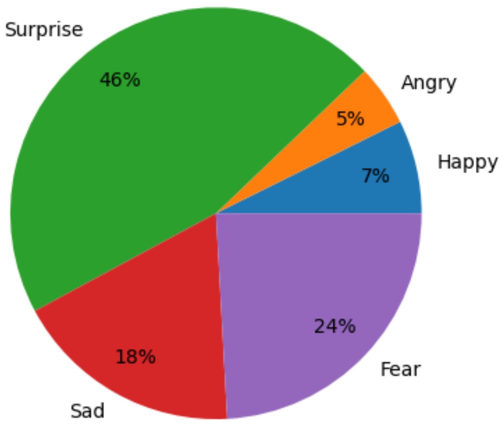
[25]https://twitter.com/i/web/status/1251555263007596545
[26]https://en.wikipedia.org/wiki/Dominic_Cummings_scandal



(a)



(b)



(c)

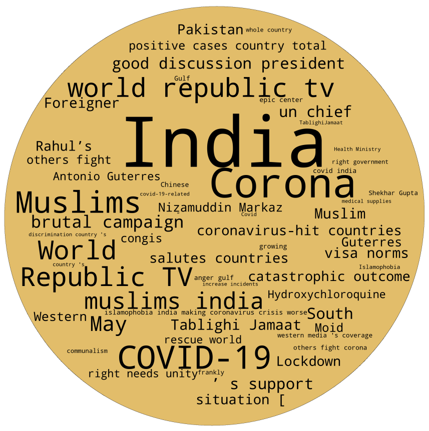
Figure 3.21: Event summary word-cloud, sentiment distribution and emotion percentages of tweets discussing Donald Trump's statement for treating COVID-19 with chloroquine. Date: 19/3/20

of tweets related to this event were 157109. The total number of tweets retrieved for this day were 2891531. The event contained $\sim 5.4\%$ of the total tweets. The number of unique users discussing about this event was 134707 and the most shared tweet [27] had 6154 retweets.

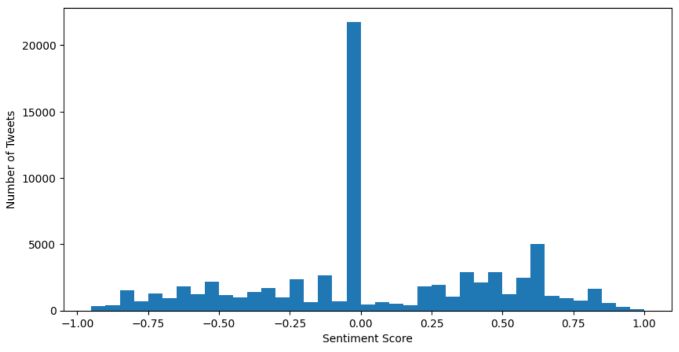
Donald Trump announced that he is terminating relationships with the World Health Organization.

On the 30th of May 2020 US President Donald Trump has announced that he is terminating the country's relationship with the World Health Organization (WHO). The president has accused the WHO of failing to hold China to account over the COVID-19 pandemic. "China has total control over the World Health Organization," the president said while announcing measures aimed at punishing Beijing[28]. The entities with the highest weighted degrees on the knowledge graph were *China, us president, the World Health Organization* and *donald trump terminates relationship* (Figure 3.24a). The sentiment of the vast majority of tweets about this event were neutral because

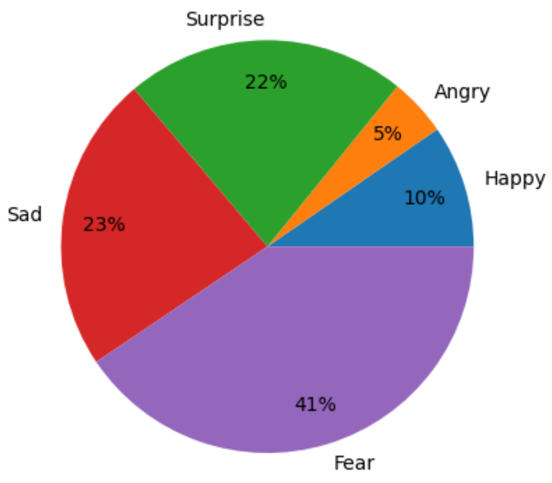
[27]https://twitter.com/i/web/status/1264234986657452043
[28]https://www.bbc.com/news/world-us-canada-52857413



(a)



(b)



(c)

Figure 3.22: Event summary word-cloud, sentiment distribution and emotion percentages of tweets discussing the #SuspendBabitaPhogat hashtag. Date: 18/4/20

most of them were tweets from news sites accounts that were reporting about this event and retweets of those reports. The average Vader score of the sentiment was -0.001 (Figure 3.24b). The dominant emotion on this event was Surprise with 94% followed by Anger with 5% (Figure 3.24c). The total number of tweets related to this event was 226772. The total number of tweets retrieved for this day was 2442116. The event contained ∼ 9.3% of the total tweets. The number of unique users discussing about this event was 225613 and the most shared tweet [29] had 5740 retweets.

Tweet about anti restrictions protests and their suppression by police goes viral in USA.

A tweet about protests and their suppression went viral on the 31st of May 2020[30]. They tweet was "when covid came w warning, the US didn't have enough masks, PPE, or tests. when the peaceful protests came, they were ready with tear gas, rubber bullets, flash guns, and tacticle gear for cops

[29]https://twitter.com/i/web/status/1266608201773899783
[30]https://twitter.com/i/web/status/1266899927134806018

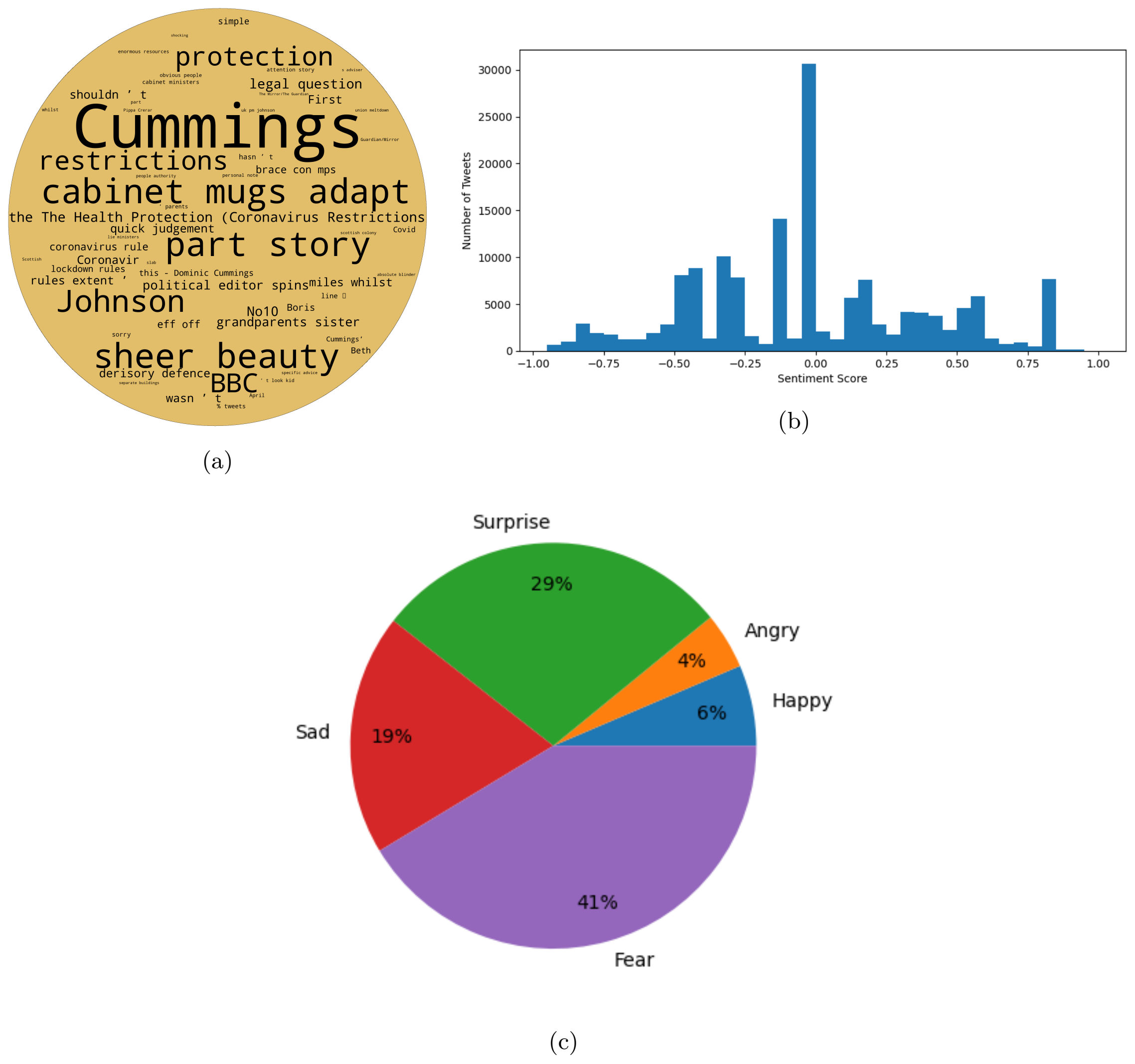


Figure 3.23: Event summary word-cloud, sentiment distribution and emotion percentages of tweets discussing the Dominic Cummings scandal. Date: 23/5/20

in multiple cities. it really shows where the priorities in this country are." and got retweeted about 100 thousand times and got more than 300 thousand likes. Some of the entities with the highest weighted degrees were *PPE, w warning* and *peaceful protests* (Figure 3.25a). The sentiment for this event was positive with average Vader score 0.318 (Figure 3.25b). The most dominant emotions of this event were Surprise and Happiness both with 32% (Figure 3.25c). The total number of tweets related to this event were 89479. The total number of tweets retrieved for this day was 2649409. The event contained $\sim 3.4\%$ of the total tweets. The number of unique users discussing about this event was 86813 and the most shared tweet [31] had 83585 retweets.

No Hospital is admitting suspected corona virus patient Ramesh Chand in Delhi India.

On the 6th of June 2020 a tweet goes viral about a suspected corona virus patient that no hospital is admitting in Delhi India. The tweet was "No Hospital in Delhi is admitting 55 year old Suspected

[31] https://twitter.com/i/web/status/1266922385602682882

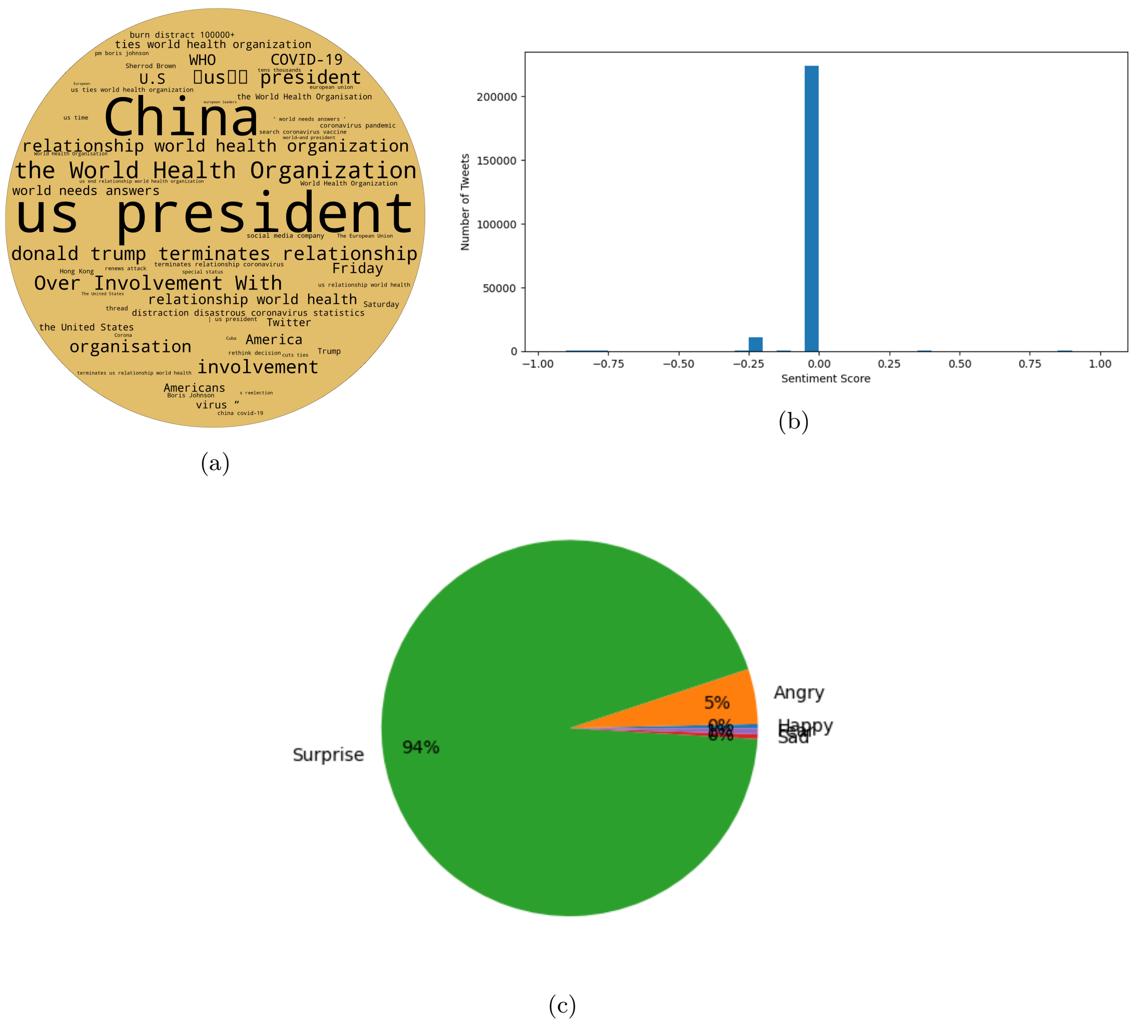


Figure 3.24: Event summary word-cloud, sentiment distribution and emotion percentages of tweets discussing Donald Trump's announcement of terminating relationships with the world health organization. Date: 30/5/20

Corona patient Ramesh Chand (9650345021) of Roshanara Road Delhi that needs immediate hospitalization"[32]. The entities with the highest weighted degrees on the knowledge graph were *Delhi, Ramesh Chand, roshanara* and *No Hospital* (Figure 3.26a). The average sentiment for this event was -0.376 (Figure 3.26b). The most significant emotions for this event were Fear with 53% and Sadness with 29% (Figure 3.26c). The total number of tweets related to this event was 100704. The total number of tweets retrieved for this day was 2538446. The event contained ~ 3.9% of the total tweets. The number of unique users discussing about this event was 100532 and the most shared tweet [33] had 2873 retweets.

New Zealand plans to lift all coronavirus restrictions, after it dealt successfully with the pandemic.

On the 8th of June 2020 New Zealand has lifted almost all of its coronavirus restrictions after reporting no active cases in the country. All of New Zealand moved to alert level one, the lowest

[32]https://twitter.com/socialjurist/status/1269128787293331458
[33]https://twitter.com/i/web/status/1269128787293331458

of the four tier alert system. Social distancing was no more required and there were no limits on public gatherings, however borders remained closed to foreigners[34]. The most significant entities on the knowledge graph were *New Zealand, Jacinda Ardern* and *coronavirus restrictions* (Figure 3.27a). The sentiment of this event is positive with average Vader score 0.190 (Figure 3.27b). The emotions for this event were Fear with 40%, Surprise with 31% and Happiness with 19% (Figure 3.27c). The total number of tweets related to this event were 86435. The total number of tweets retrieved for this day was 2415117. The event contained $\sim$ 3.6% of the total tweets. The number of unique users discussing about this event was 74612 and the most shared tweet [35] had 4766 retweets.

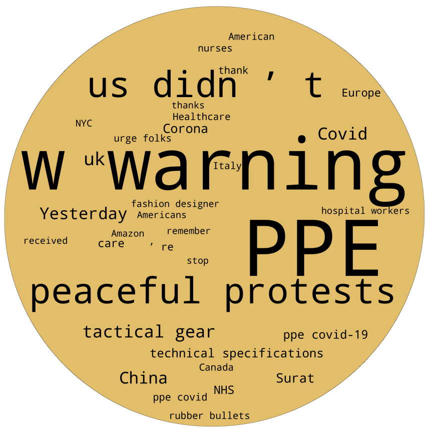
ICU nurse tweets about the severity of COVID-19 goes viral.

On 14 June 2020 an intensive care unit (ICU) nurse has shared on Twitter an insight into her daily routine of taking care of COVID-19 patients since the outbreak of the disease[36]. On two tweets she wrote: "COVID 19 is the worst disease process I have ever worked with in my 8 years as an ICU nurse. When they say recovered they don't tell you that that means you may need a lung transplant. Or that you may come back after d/c with a massive heart attack or stroke bc COVID makes your blood thick as hell. Or that you may have to be on oxygen for the rest of your life. COVID is designed to kill. It is a highly intelligent virus and it attacks everything. We will run out of resources if we don't continue to flatten the curve. I'm exhausted." The entities with the highest weighted degrees on the knowledge graph were *ICU disease process, covid* and *patients ages* (Figure 3.28a). The sentiment about this event was very negative with average Vader score -0.533 (Figure 3.28b). The dominant emotion for this event was Surprise with 59% followed by Fear 20% and Sadness 19% (Figure 3.28c). The total number of tweets related to this event were 68142. The total number of tweets retrieved for this day was 2445992. The event contained $\sim$ 2.8% of the total tweets. The number of unique users discussing about this event was 66771 and the most shared tweet had 59047 retweets.

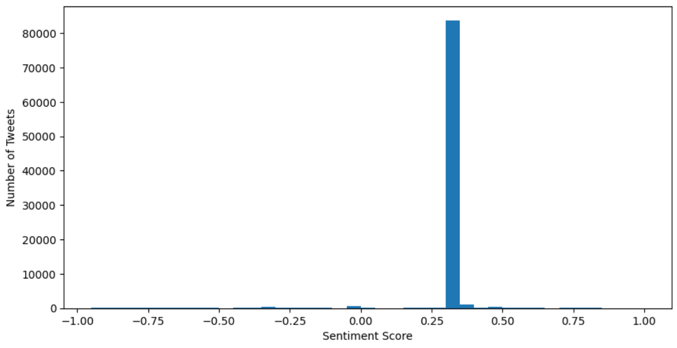
Donald Trump's rally in Tulsa Oklahoma USA.

On June 20, 2020, Donald Trump held a rally for his 2020 presidential re-election campaign in Tulsa, Oklahoma. Amidst the COVID-19 pandemic, the event marked his first public campaign event since March 2020[37]. Critics and health officials warned that as a large public gathering in a confined indoor space, there was a high probability that new COVID-19 infections could occur. More critic also occur on the media when a young couple allegedly sighted on the rally with their

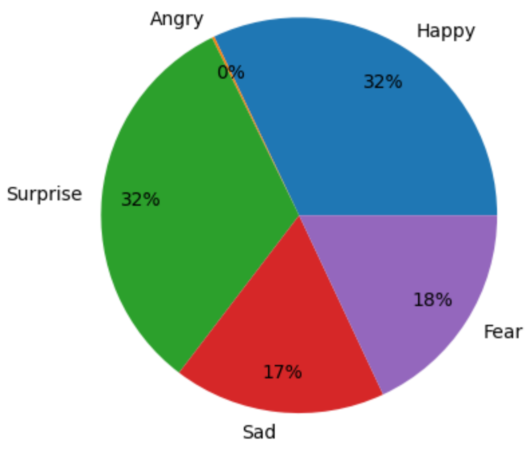
[34]https://www.bbc.com/news/world-asia-52961539
[35]https://twitter.com/i/web/status/1269931896265465857
[36]https://www.lindaikejisblog.com/2020/6/covid-19-is-the-worst-disease-process-ive-ever-worked-icu-nurse-laments.html
[37]https://en.wikipedia.org/wiki/2020_Trump_Tulsa_rally



(a)

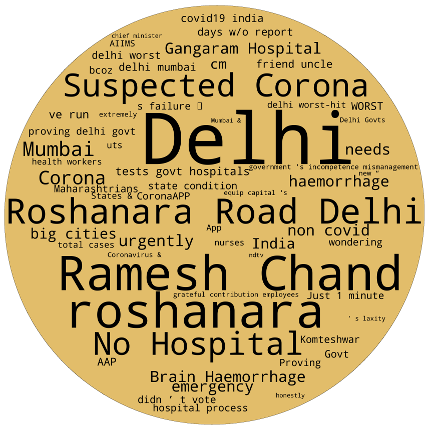


(b)

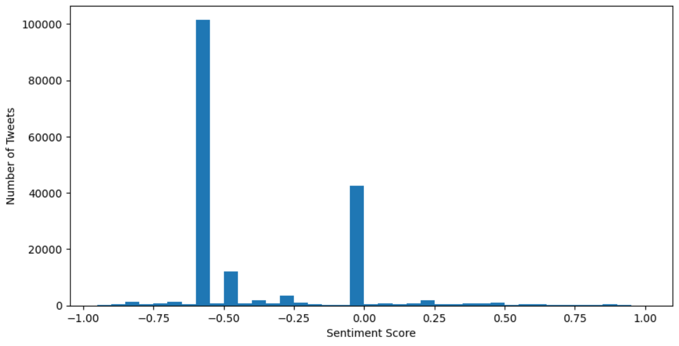


(c)

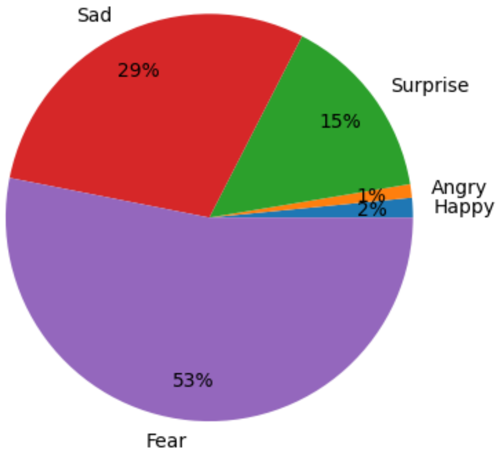
Figure 3.25: Event summary word-cloud, sentiment distribution and emotion percentages of tweets about anti restrictions protests and their suppression by the police. Date: 31/5/20



(a)



(b)



(c)

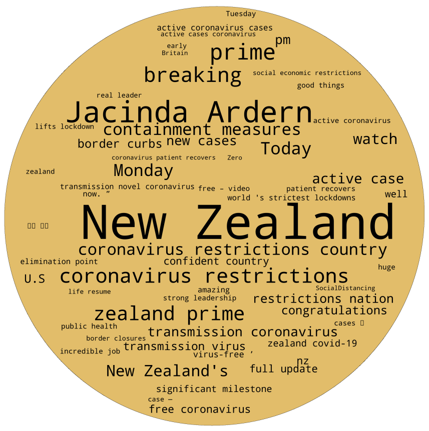
Figure 3.26: Event summary word-cloud, sentiment distribution and emotion percentages of tweets about no hospital admitting corona virus patient Ramesh Chand in Delhi India. Date: 6/6/20

newborn baby[38]. The most significant entities on the knowledge graph were *Trump, Six, COVID-19, Newborn Baby* and *the Tulsa Rally* (Figure 3.29a). The sentiment about this event was mostly neutral with average Vader score -0.021 (Figure 3.29b). The dominant emotion for this event was Sadness with 79% followed by Fear with 11% (Figure 3.29c). The total number of tweets related to this event were 155240. The total number of tweets retrieved for this day was 2900228. The event contained $\sim$ 5.4% of the total tweets. The number of unique users discussing about this event was 137073 and the most shared tweet [39] had 25129 retweets.

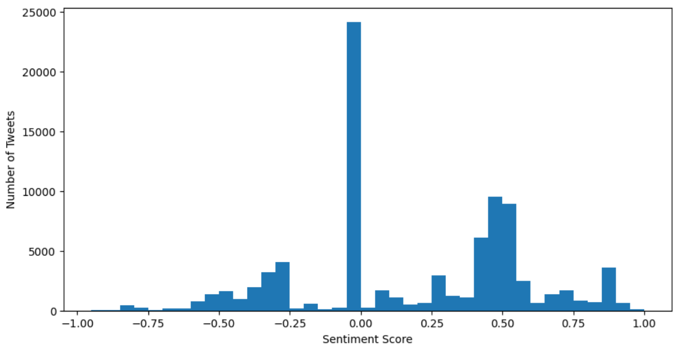
Fiji prime minister tries to attract billionaires for vacations.

The prime minister of the Republic of Fiji, Frank Bainimarama, has announced the country is looking to attract "VIPs" to help restore Fiji's paralysed tourism-dependent economy. Bainimarama said the country would also welcome travellers arriving by yacht who were prepared to spend

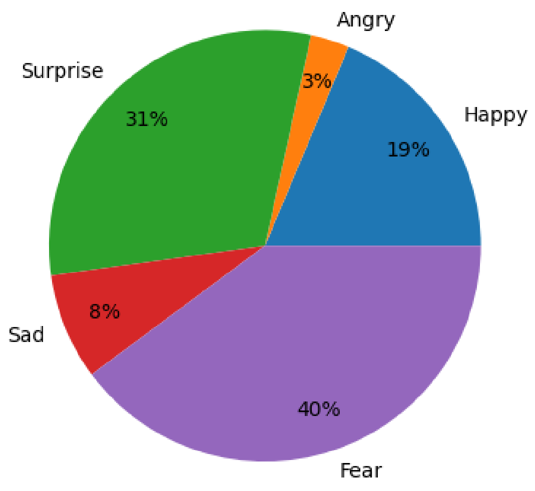
[38]https://nypost.com/2020/06/20/woman-brings-newborn-to-trumps-tulsa-rally/
[39]https://twitter.com/i/web/status/1274770425055268864



(a)



(b)



(c)

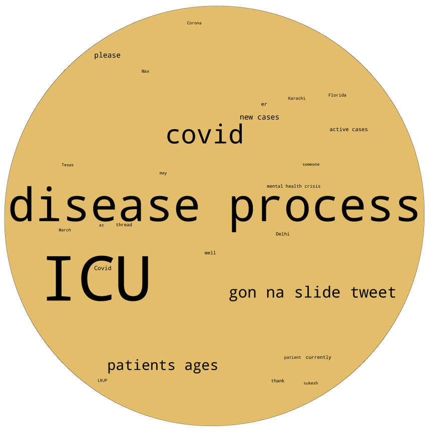
Figure 3.27: Event summary word-cloud, sentiment distribution and emotion percentages of tweets about the lift of restrictions for the pandemic in New Zealand. Date: 8/6/20

14 days at sea – or make up the balance in quarantine in Fiji[40]. The entities with the highest weighted degrees on the knowledge graph were *Fiji, fiji covid-19* and *COVID-19* (Figure 3.30a). The average sentiment score is -0.472 (Figure 3.30b). Finally the dominant emotions are Fear with 42%, Surprise with 28% and Sadness with 21% (Figure 3.30c). The total number of tweets related to this event were 94868. The total number of tweets retrieved for this day was 3089873. The event contained ∼ 3.1% of the total tweets. The number of unique users discussing about this event was 94500 and the most shared tweet [41] had 94368 retweets.

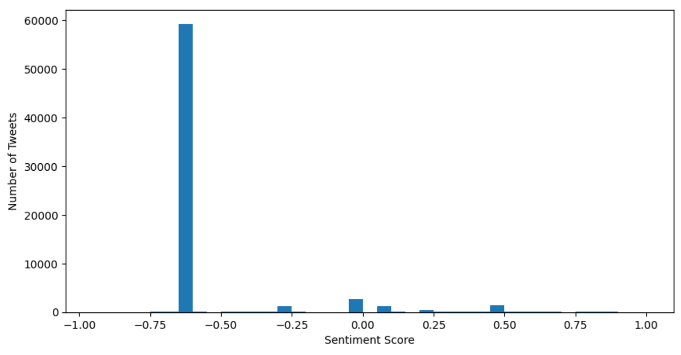
3.5 Discussion

In this work we suggest that NED performs better than other methods for detecting events in news streams because it focuses on named entities (highly relevant to news events) and, in particular,

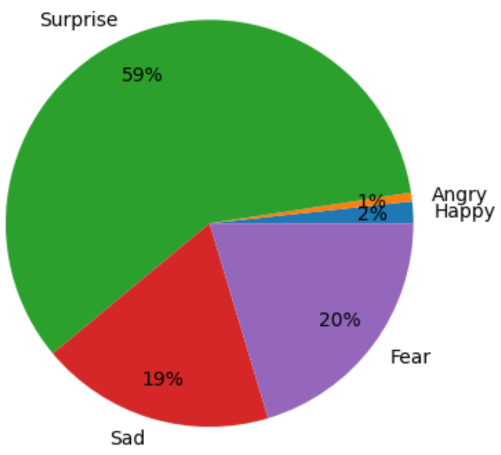
[40]https://www.theguardian.com/world/2020/jun/28/escape-the-pandemic-in-paradise-fiji-opens-its-borders-seeking-billionaires
[41]https://twitter.com/i/web/status/1277081610806890499



(a)

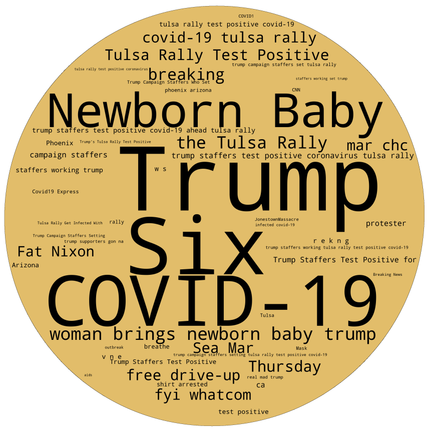


(b)

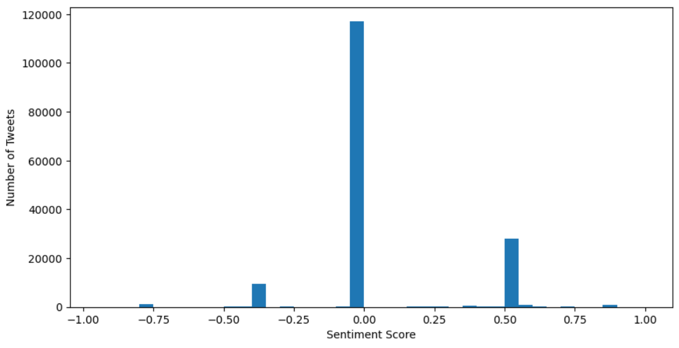


(c)

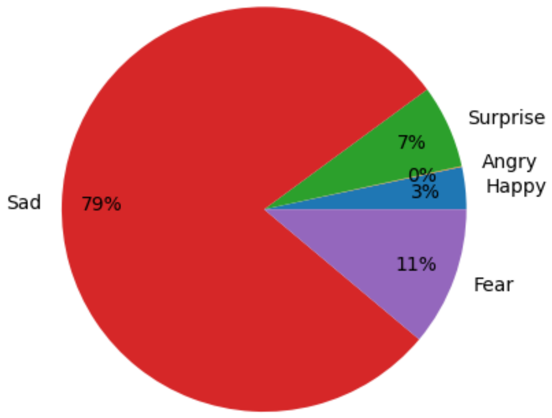
Figure 3.28: Event summary word-cloud, sentiment distribution and emotion percentages of tweets about the ICU nurse tweets going viral. Date: 14/6/20



(a)

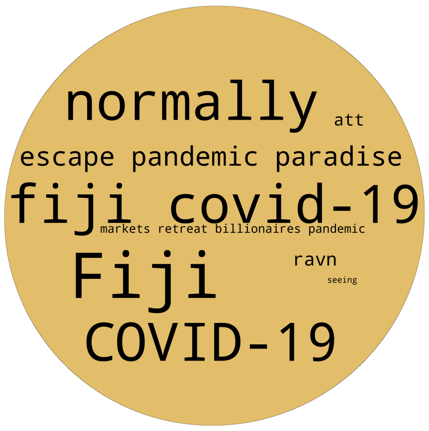


(b)

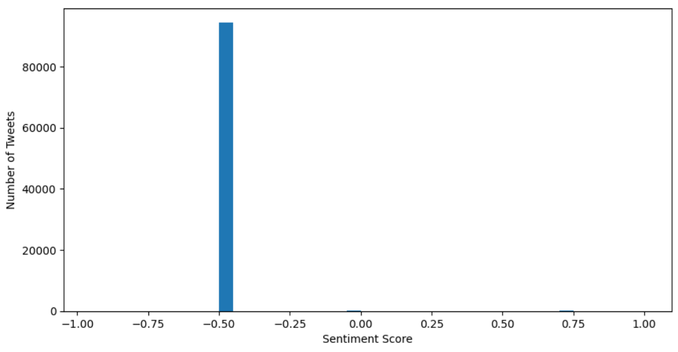


(c)

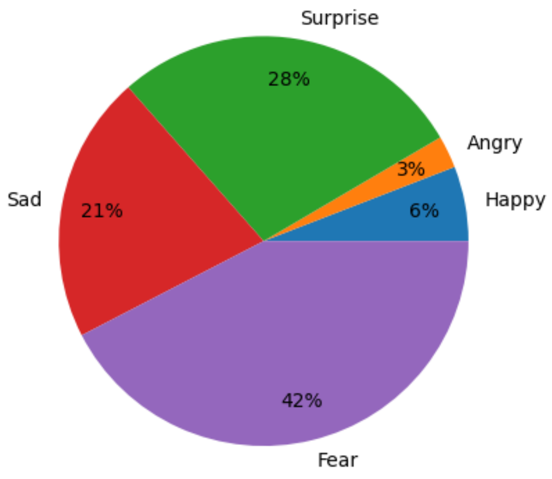
Figure 3.29: Event summary word-cloud, sentiment distribution and emotion percentages of tweets about Donald Trump's election rally at Tulsa Oklahoma. Date: 20/6/20



(a)



(b)



(c)

Figure 3.30: Event summary word-cloud, sentiment distribution and emotion percentages of tweets about Fiji prime minister inviting billionaires for vacations. Date: 28/6/20

it identifies ‘peaking entities’ which show an increased level of prominence within the dynamic knowledge graph. This helps to remove irrelevant articles from further processing. Here such articles constitute ‘noise’ in the document stream and do not contribute anything to the detection and description of the trending topics or events. The approach removes a substantive proportion of articles in this way, in some cases around 50% of them.

Another advantage of our method is the use of named entities and noun phrases to form communities of n-grams that describe an event. This approach effectively avoids ambiguous situations where the same keywords are used in documents that describe different events. By treating named entities with their full name we can separate n-grams that share a keyword. For example n-grams such as Theresa May and the month May, or David Cameron and David Davis, can introduce ambiguity and negatively affect the results of the topic detection methods.

By utilizing the entity co-appearance network the NED method captures relational information about entity interactions. Thereby it can efficiently detect peaking entities based on an approach where an entity is important not only because it appears many frequently itself, but also because it co-occurs with other important entities. The use of relational information allows event detection to access holistic patterns in articles.

By using named entities and noun phrases to generate a description of each detected topic/event, the NED method produces an easy-to-understand summary of the event. We suggest that this method performs qualitatively as well or better than the other methods tested here, though this is a subjective evaluation and others may disagree.

We also demonstrated the benefit of sentiment analysis as a method of event characterisation using three case studies of political elections. The NED methodology is able to extract the sentiment of the engaged user population about each of the detected events. For the events in question, analysis of sentiment clearly shows that while election outcomes generate a lot of positive sentiment (possibly explained by a greater tendency of users to tweet about an outcome they feel happy about), the distribution of sentiment is bimodal. This suggests a scenario where different user groups have opposite affective responses to the election outcome, as might be expected in a major election event in a two-party democracy (and potentially exacerbated by the recent increase in political polarisation). These findings suggest potential applications for this methodology in understanding social processes more widely, seeking first to find events of public interest (using our network-based event detection method to integrate user posts from large populations) and then to characterise their impact on public sentiment and well being.

Finally we utilized our methodology to detect the main events of the discourse about the COVID-19 pandemic on Tweeter for the time period between January to June of 2020. The emotions classifier was not validated or compared with other classifiers in terms of accuracy, although it performs really well in terms of computation performance. In future work we are planning on investigating whether this classifier is good enough for this task or if there are any better alternatives.

From the detected events we conclude that most of the peaking events about the coronavirus have a negative sentiment. Also the most dominant emotions were Fear and Surprise even in positive events such as New Zealand's termination of restrictions where someone would expect the Happiness emotion to be more prevalent. Another insight we can observe is that many COVID-19 related events are also related with politics which makes sense since the pandemic was the cause of new legislation and restrictions. In particular events where President Donald Trump was involved, his tweets went viral and polarized public opinion as we can see on Figures 3.19b, 3.20b and 3.21b.

Given that we evaluated our methodology on data sets consisting of both news articles and tweets we are confident that NED will return objectively important events. In the case where the user wishes to use data from other sources such as forum or Reddit posts, other blogging or micro blogging platforms, or use documents written in other languages than English, we suggest that the user should apply the following adjustments on our methodology. For steps 1, 4 and 7 of our methodology (figure 3.1) the user should consider using other classifiers that perform better on the chosen data although the performance of the classifiers we chose for those tasks is satisfactory even on heterogeneous data. However specific classifiers for the data sets that the user wants to work with might produce better results. For the case where the user wishes to use our methodology for documents written in different languages than English new classifiers should be used for steps 1, 4 and 7. From our experience so far with the NED methodology we believe that steps 2, 3, 5 and 6 do not need any adjustments when working with unknown data although this claim is hard to prove (if not impossible). In any case the only way for a user to be sure that our methodology (and in our opinion any other methodology he or she might use for event detection) produces objectively important events is to evaluate it against a manually annotated sub set of the data.

For the analysis on the COVID-19 data set, although we did not run an evaluation on the results our methodology produced, we are confident that those results are significant. The main reasons for that are that first, we have already run an evaluation test against other state of the art methodologies on tweeter data and NED outperformed most of them, second the components we are using, in step 1 for example, are also evaluated by their creators and they also outperform similar state of the art techniques. Lastly by manually examining the results that our method returned about the COVID-19 outbreak, we are confident enough, at least in our opinion, that those events were significant if not the most significant events during the time period we are examining. However we can not claim with certainty that other methods would converge to the same results and the only way to test that is by applying other methodologies to the same data set and compare their results with ours. This task was not one of the purposes of our work and we consider it out of the scope for this thesis, although we are considering to apply such tests in future work.

To evaluate how fast our methodology performs when applied to big amounts of data we measured the time it took for NED to process the COVID-19 data set. The system we run our methodology was a personal computer with an Intel Core i5-8250U CPU at 1.60GHz with four cores using only one core, 8 gigabytes of ddr4 ram, 1 terabyte SSD and we used the Ubuntu 20.04.5 LTS operating system. There were on average about 1.3 million tweets per day and the processing time for this average was 9042 seconds or about 2.51 hours. In the worst case when on one day (19/3/2020) about 7.65 million tweets were posted, it took our method about 53200 seconds or about 14.8 hours to process. In total our data set contains about 239 million tweets and it took our method about 462.8 hours or about 19.3 days to process. Given the available hardware we strongly believe that the performance of our method makes it feasible to process big amounts of tweets generated from a streaming source. Regarding the memory consumption of our method the maximum amount of ram required by it was about 700 megabytes. The most demanding part of our methodology in terms of processing time is the named entity recognition step. However given that named entity recognition can be applied on each document independently it can be parallelized very easily improving the performance of our method drastically. Even on the CPU we used we can improve the processing performance of the method almost four times by simply parallelizing the named entity detection step.

Of course the NED methodology has some disadvantages as well. One of the weaknesses of our methodology is that some of the event clusters might be merged together. For example in figure 3.11 the clusters labeled as North Korea Nuclear Threat and Freddie Grey Trial contain entities and noun phrases that are related with other events such as the Olympic games in Rio in 2016 or the Abigail Fischer University admission trial. This is a weakness of our methodology and currently it is not addressed by it. Possible reasons that this is happening are when two events have common named entities, when the number of documents reporting an event is small and when some documents are reporting more than one events. The solution for this weakness will be addressed on future work but a possible direction for that is to take into account the proximity of two entities inside the text. For example co-appearing on the same paragraph or sentence. However this weakness seems to be mitigated when the number of documents increases which is the case on real life news streams in contrast with our test data set. For instance the collection of documents containing the detected events of figure 3.11 consisted of 299 documents. Table 5.1 displays the annotations for the articles published at 24/6/2016. The annotations were produced and validated manually by one of the authors of this thesis. Although we would prefer to annotate and validate the data set with multiple annotators we did not had the resources and time for such task. This affects the quality of the annotated data and one should consider to use them with caution. However the same data were used for the evaluation of all methods that were compared with ours so ideally we assume that all would be affected more or less the same by their quality.

3.6 Conclusion

This work presents a prototype news event detection methodology, based on natural language processing and network analysis. Events are located by finding peaks in the prominence of named entities, based on node-level time series in the entity knowledge graph, characterized by community detection in KeyGraphs linking entities and noun-phrases and sentiment analysis.

Our evaluation suggests that NED provides a significant improvement against other state of the art methods. This is supported on two qualitatively different data sets (news articles and tweets). The combination of named entities and social network analysis techniques, such as community detection, seems to be very effective in detecting and tracking topics in document streams, and provides a more comprehensive description of each detected event, compared with the rest of the evaluated methods.

Future work will seek to improve the NED method. One avenue for exploration is whether it is useful to further split the communities identified in the knowledge graphs and KeyGraphs into sub-communities. This may improve the resolution of event detection by splitting communities that include two different events but share a lot of common nodes. Another area to investigate related to community detection is whether assignment of entities to multiple communities will give a better representation of the event that each community describes. This is possible using methods for detecting overlapping communities [176]. A final area of investigation will be to test the utility of other network metrics as indicators of changing network structure that might more effectively detect news events. Such metrics might include node-level statistics such as various centrality measures, or perhaps meso-level features such as dynamic community structure.

Chapter 4

Community evolution on Stack Overflow

4.1 Introduction

The software and computing industries change rapidly as new technologies and platforms are introduced. The ecosystem of programming languages, methodologies and developers is highly complex, with tools and behaviours continually adapting to internal and external forces. Widespread adoption of web-based and/or "cloud" computing paradigms has altered not only the software itself (in terms of languages and architectures) but also how software is created. The sharing and re-use of code via platforms such as GitHub [1], Bitbucket [2], and Source Forge [3] enable more effective collaboration, while the easy availability of programming expertise online on sites such as Stack Overflow [4] and Quora [5] has changed how individuals seek help with programming challenges. Whereas once an individual might seek an answer in a text book or from an office colleague, it has become the norm to seek help online.

Question-and-answer websites such as Stack Overflow are widely used by programmers and researchers, forming a large repository of valuable knowledge related to the software development, computing, and data science industries. Software developers rely on such websites to acquire knowledge, solve problems, seek snippets of code for reuse, improve their own code, and discuss technical concepts. Stack Overflow also helps individuals gain visibility to establish professional standing and reputation. At time of writing, the Stack Overflow community has more than 14 million registered users who have asked 21 million questions and received 31 million answers;

[1]https://github.com/
[2]https://bitbucket.org
[3]https://sourceforge.net/
[4]https://stackoverflow.com/
[5]https://www.quora.com

many more unregistered users utilise the publicly displayed questions/answers provided by others without contributing new content. Approximately 70% of questions receive an answer [6] making the platform one of the best places to seek help or discuss emerging technology issues.

Stack Overflow is written by many and read by many. To help ensure good content is easily visible, Stack Overflow provides a voting system where users that contribute high quality answers or interesting questions are assigned positive votes and thematic tags (e.g. ‘C++’, ‘Python’, ‘Machine-Learning’) by other users. Thereby a ‘reputation’ score is formed that identifies the most knowledgeable users for specific fields. The reputation score is also used to give privileged roles to high-ranked users, such as up-voting, editing and moderating the community. Moderation of the community is quite strict; only well-documented, on-topic and correctly tagged questions/answers are accepted. These features ensure that content is of high quality, but also provide a rich resource for social analysis of the platform.

The social dimensions to online programming platforms such as GitHub and Stack Overflow are an intrinsic part of their function. Various studies have tried to understand social factors, for example, seeking to identify influential users [75, 188] or understand their general properties as socio-technical systems [161]. The social component also provides useful information about wider trends in the software industry, with user activity reflecting the shifting popularity of different technologies.

Here we analyse the evolution of the Stack Overflow user community over a relatively long period (2008-2020). By tracking the usage of different tags by individual users, we are able to provide insight into the clusters of topics that are the focus of clusters of users, and observe trends in the adoption of new programming languages and technologies. It is reasonable to assume that trends on Stack Overflow, revealed by analysis of users, tags and various platform metrics, are reflective of wider trends in the software industry. Thus we use Stack Overflow as a lens with which to study attention to different technologies, reveal technology clusters defined by the user groups that utilise them, and observe the movement of people between different technological clusters over time.

The core of our methodology is the construction of networks that link users to each other based on the tags that define their (shared) expertise. Within these networks we use community detection algorithms to identify sub-communities representing groups of users focused on particular technology clusters, using the set of tags associated with users to characterise each sub-community. By analysing a temporal sequence of such networks we are able to explore the concurrent evolution of the programming community and underpinning technologies over time. We examine how the various sub-communities relate to each other and identify different technologies with common applications. By monitoring the movement of users between communities over time, we show that the majority of users either remain in the same community or didn’t acquire any score, and

[6] https://stackexchange.com/sites#technology

that only a small fraction of users migrate between communities. The rise and fall of different technologies, revealed by the number of users who are interested in them and the way technologies are clustered, provides insight into the dynamics of the tech industry during a period of rapid change.

The next section describes some relevant Background, including a brief description of the operation of the Stack Overflow platform and some related work using similar data. In the Data set & Methods section, we describe the data collection and pre-processing steps, as well as the processes used to generate the findings given in the Results section. The Discussion section concludes the chapter.

4.2 Background

4.2.1 Stack Overflow

Stack Overflow is a self-moderating online Question & Answer forum. Stack Overflow questions are generally hard, requiring expertise and domain knowledge to provide a good answer.

Stack Overflow's success is largely due to the engaged and active user community that collaboratively manages the site. Content is heavily curated by the community. Duplicate or similar questions are quickly identified as such and merged with existing questions. Posts (questions or answers) that are unhelpful or irrelevant are removed. As a result of this self-regulation, content on Stack Overflow tends to be of high quality.

The quality of each post is collaboratively evaluated using a voting system. Each question or answer can receive up-votes or down-votes from users, with the sum of votes (up-votes minus down-votes) acting as its overall voting score. The votes awarded to a user's posts is accumulated in their 'reputation', another type of score associated with individual users and intended to identify expert users.

Reputation brings moderation privileges. Each user gets the ability to up-vote a post when their reputation score reaches 15 points and the ability to down-vote a post when their reputation reaches 125 points. On Table 4.1 we present all the available privileges on the platform as well as the required reputation for each one and the percentage of the total users that poses them.

Another key mechanism on the Stack Overflow site is the use of tags to identify the content or theme of each post. When a user asks a question, the platform prompts them to add a small number of content tags (at least one and at most five).

A user can refer to the website's help center (https://stackoverflow.com/help/) where he can learn more about Stack Overflow and its rules.

Reputation	**Privilege**	**Description**	**Percentage**
25000	Access to site analytics	Access to internal and Google site analytics	0.05%
20000	Trusted users	Expanded editing, deletion and undeletion privileges	0.07%
15000	Protect questions	Mark questions as protected	0.1%
10000	Access to moderator tools	Access reports, delete questions, review reviews	0.16%
5000	Approve tag wiki edits	Approve edits to tag wikis made by regular users	0.35%
3000	Cast close and reopen votes	Help decide whether posts are off-topic or duplicates	0.6%
2500	Create tag synonyms	Decide which tags have the same meaning as others	0.71%
2000	Edit questions and answers	Edits to any question or answer are applied immediately	0.88%
1500	Create tags	Add new tags to the site	1.14%
1000	Established user	You've been for a while; see vote counts	1.61%
1000	Create gallery chat rooms	Create chat rooms where only specific users may talk	1.61%
500	Access review queues	Access first posts and late answers review queues	2.81%
250	View close votes	View and cast close/reopen votes on your own questions	4.37%
200	Reduce adds	Some add are automatically disabled	4.77%
125	Vote down	Indicate when questions and answers are not useful	6.42%
100	Edit community wiki	Collaborate on the editing and improvement of wiki posts	7.62%
100	Create chat rooms	Create new chat rooms	7.62%
75	Set bounties	Offer some of your reputation as bounty for a question	8.9%
50	Comment everywhere	Leave comments on other people's posts	11.17%
20	Talk in chat	Participate in the site's chat rooms	18.49%
15	Flag posts	Bring content to the attention of the community via flags	19.44%
15	Vote up	Indicate when questions and answers are useful	19.44%
10	Remove new user restrictions	Post more links, answer protected questions	25.28%
10	Create wiki posts	Create answers that can be easily edited by most users	25.28%
5	Participate in meta	Discuss the site itself; bugs, feedback, and governance	26.20%
1	Create posts	Ask a question or contribute an answer	100%

Table 4.1: Users distribution by Stack Overflow privileges. Percentage of users possessing privileges on Stack Overflow. Every user poses all the privileges that require less or equal reputation than he/she currently has. More information can be found here: (ht-tps://stackoverflow.com/help/privileges).

4.2.2 Related work

The research literature includes various studies that analyse question-and-answer websites, typically using data from Stack Overflow, Quora [7] or Yahoo! Answers [8]. These studies can be loosely grouped into three categories: studies of network structure, studies of content, and studies of information retrieval. We cover these in turn below.

Studies of network structure explore the relationships between entities, such as users, posts or tags, that are associated with question-and-answer websites. Communities can be identified from network structure and analysed to detect key actors, as well as the main interests and typical behaviours of the users. Silvestri et al [152] describe a methodology for linking user accounts between platforms (across Stack Overflow, Github and Twitter) based on user attributes and platform specific services, examining different account matching strategies. Wang, Liu & Fan [168] introduce a methodology to discover similar users in online communities based on the tags they share. Beyer and Pinzger [18] introduce an approach to group tag synonyms to meaningful topics by creating networks, and investigating community detection algorithms to build meaningful groups of tags. Rosen & Shihab [139] analyze more than 13 million posts from Stack Overflow to examine topics of discussion amongst mobile application developers, finding that this community is focused on app distribution, mobile APIs, data management, sensors/context, mobile tools, and user interface development. Fu, Yu & Benson [57] analyze the tag networks of several Stack Exchange communities and develop a simple generative model that creates random bipartite graphs of tags and questions. Halavais et al [69] investigate whether individual badge-seeking behaviour is motivated by exposure to others' achievements, concluding that 'general' badges are closely related to tenure on the site, while more numerous 'tag' badges are associated with socially influenced differentiation. Papoutsoglou, Kapitsaki & Angelis [127] introduce a methodology for modeling the effect of the badge reward system on the personal traits of Stack Overflow users based on data recorded before and after the award time, employing the 'Big Five' personality Factors.

Studies of content on online question-and-answer communities typically analyse the content and metadata of questions and answers. Calefato et al [27] investigate how Stack Overflow users can increase the chance of getting their answer accepted when writing an answer or making comments, finding that information presentation, timing and affect all have an impact on the success of a post. Schenk & Lungu [149] use the geospatial metadata associated with each Stack Overflow user to understand how different geographic regions contribute to the knowledge market represented by the community. They find that Europe and North America are the principal (and roughly equal) contributors, with Asia as a distant third (mainly India), followed by Oceania, which even in fourth position still contributes more than South America and Africa combined. Morrison & Murphy-Hill [110] aim to support career planning and staff development by identifying age-related

[7] https://www.quora.com/
[8] https://answers.yahoo.com/

trends in Stack Overflow data, observing that user reputation scores increase with age well into their fifties, that programmers in their thirties tend to focus on fewer areas than those younger or older in age, and that there is not a strong correlation between age and reputation scores for any specific knowledge areas. Ragkhitwetsagul et al [135] investigate the quality of code Stack Overflow answers contain, how often it gets adopted (cloned) by the community, and how often it is reviewed/modified by the answer author. Their research shows that a significant amount of reused Stack Overflow code ($\sim$ 66%) is outdated, of which ($\sim$ 6.6%) was buggy and harmful for reuse. Vasilescu et al. [163] similarly investigate the relation between Stack Overflow questions and answers, and the software development process, as reflected by code changes committed to the GitHub code repository. They find that Stack Overflow activity rates correlate with activity in GitHub.

Studies of information retrieval related to question-and-answer communities typically adopt a perspective where a question is viewed as a ‘query’ and answers as ‘results’. De Sousa, Campos & Maia [41] make use of ‘crowd knowledge’ from Stack Overflow to recommend information to developers. Ranked lists of question/answer pairs are recommended in response to a query based on textual similarity and the question/answer vote scores. Liu et al [97] predict ‘searcher satisfaction’ by analysis of a large number of web searches that result in a visit to a popular question-and-answer site (Yahoo! Answers). They identify query clarity, query-to-question match, and answer quality, as key factors in predicting searcher satisfaction. Xu et al [179] use Stack Overflow as their source of question-and-answer threads, achieving good results with an attention-based model that predicts which answer will be preferred by the user posting the original question. Zhang et al [188] propose a methodology for duplicate question detection in question-and-answer communities, adopting a classification approach based on text vectorisation and neural networks.

Data Collection & Methods

Our methodology uses interactions between users and tags on Stack Overflow to explore trends in software development and technology usage over time. The assumption is that the tags attached to posts by a user, and the reputation score they acquire from posts using those tags, form a profile for each user that defines their interests and expertise. These profiles can be used to link pairs of users based on the similarity of their expertise. Pairwise links can be aggregated to form a network of users within which community structure reflects groupings of users and technologies. These networks can be studied over time to explore trends.

This section describes how the data were collected and some pre-processing steps that were used to, first, associate content tags to ‘answer’ posts (this information is not provided in the data files), and second, assign scores to each user based on the tags they used in a given time period. Then we describe the main parts of the network creation and analysis methods, including how each community was characterised based on its dominant tags. Overall, this analysis pipeline permits the Stack Overflow developer community to be studied over time and to thereby reveal trends in software development and technology usage.

4.2.3 Data collection and pre-processing

For our analysis we obtained all questions, answers [9], votes [10] and tags [11] on the Stack Overflow website between its inception on 31st July 2008 and 31st December 2020. All the data were retrieved from the Archive.org platform [12], which hosts the entire history of every Stack Exchange community, including the tags used to annotate questions and the votes of each question and answer. This is an anonymized dump of all user-contributed content on the Stack Exchange network. Each site is formatted as a separate archive consisting of XML files zipped via 7-zip using bzip2 compression and updated every three months. Each site archive includes Posts, Users, Votes, Comments, PostHistory and PostLinks.

All user content contributed to the Stack Exchange network is cc-by-sa 4.0 [13] licensed, intended to be shared and remixed. The acquisition, processing and presentation of these data fully complies with the terms and conditions of the Stack Exchange network.

The XML files were quite large. The *posts* file was ∼70 gigabytes, the *votes* file was ∼16.7 gigabytes, and the *tags* file smaller at ∼5 megabytes. From the creation of the platform on 31st July 2008 until December 2020 there were ∼ 46 million posts, of which ∼18 million were questions and ∼27 million were answers.

The Stack Overflow platform awards reputation scores depending the vote type [14]. If an answer gets accepted the author of the question gets 5 score points and the author of the answer 15 points. If a question gets up-voted the author gets 10 points and if an answer gets up-voted the author gets 10 points. If a post gets down-voted the author gets -2 points, as also does the user who cast the down-vote.

The first step of our analysis was to parse the data files to group together posts that were created in each month. For some analyses, we used a monthly time unit, for others yearly, making a 1-month bin size suitable as input for both.

Before user networks could be created, we first associated each user with a set of tags reflecting their interests and expertise. The up-votes each user receives on a post are also associated with the tags assigned to the post. For example if a user receives 100 reputation on a post the co-responding tags will be assigned to the user as well and each one of them will receive the same reputation. This creates a ranking of related tags for each user. Our goal is to utilise those tags to form relations

[9] https://archive.org/download/stackexchange/stackoverflow.com-Posts.7z
[10] https://archive.org/download/stackexchange/stackoverflow.com-Votes.7z
[11] https://archive.org/download/stackexchange/stackoverflow.com-Tags.7z
[12] https://archive.org/download/stackexchange
[13] https://creativecommons.org/licenses/by-sa/4.0/
[14] https://stackoverflow.com/help/whats-reputation

between users based on how similar their tags are. Those tags are based on the reputation score the users acquired from posts containing those tags.

The data archive provides the list of tags associated with each question post, but does not provide this information for answer posts. Therefore, we annotated each answer post with the same tags as its parent question post, using the *ParentId* attribute to recover the corresponding question record. To do that efficiently with the large files, we created indexes for each question post pointing to the time period the question was posted, then used these to load only the questions of each time period, indexed on their post *Id*, then retrieved each question's tags and added them to an expanded record for each answer post.

At the end of processing, the product was a data dictionary for each month with the user IDs as keys and a ranked list of their top tags based on the reputation score they acquired.

4.2.4 Graph creation and community detection

For our analysis, we created graphs where the nodes are the users and the edges between them represent how related two users are in their technology focus, based on how similar tags those users have. For calculating the weight of each edge we first normalized the score of each user's tags creating a vector where the sum of all tag scores is 1 and then applied the Manhattan distance between the user's vectors (equation 4.1). This way we get 0 for identical users and 2 for users with completely different tags. We convert the distance into similarity by subtracting it from two. Our similarity metric determines which tags mostly represent each user and by comparing the users vectors we can calculate how similar two users are. We chose to use the Manhattan distance after some experimentation with other metrics we came up by our selves but did not work properly and because its computational performance was appropriate for the amount of data we had to process.

$$EdgeWeight(U_1, U_2) = 2 - \sum_{t \in Tags} |ScoreU_1(t) - ScoreU_2(t)| \tag{4.1}$$

Where U_1, U_2 are the users and $Tags$ the union of their tags.

The motivation behind this graph creation approach is to link users that have similar tags. Using this similarity-based edge creation approach we cluster users that have similar technology interests. Alternatively, we might have linked users that interacted directly on the StackOverflow platform, e.g. by providing an answer to a query. This kind of user interaction is subsumed within our similarity-based edges, since both questions and answers are associated with the same tags. Our approach additionally allows users with similar interests to be linked, even if they have never interacted directly; this aspect serves to provide a more coherent network structure. We then monitor if our methodology produces a meaningful structure, detect user communities and annotate them based on the tags of the users belonging to each community. Our priority here is to reveal which tags are grouped together on each community and, by annotating each community, to understand which technologies are used on each software development field.

The approach we took for dynamic community discovery was the Instant Optimal Community Detection approach [140] which considers that the detected communities on every time step (one year in our case) are independent. This approach consists of two stages. The Identify stage where communities are detected on each step of the evolution and the Match stage where the detected communities on each step are aligned with the communities of the other steps. For each calendar year in the period 2008- 2020, we created one graph containing the top 100 thousand users based on their reputation that were active during that year to sparsify the networks and their tag-based relationships (For years 2008 ($\sim$ 17000 users) and 2009 ($\sim$ 65000 users) we included all active users because Stack Overflow launched on 31st July 2008 and got bigger after 2009).

We then applied the InfoMap [142] community detection algorithm to identify community structure within these graphs. Infomap is a pattern-based community detection algorithm and is based on the concept of patterns of random movement (walks) among the nodes of a network. The main intuition of this method is that a community can be defined as a group of nodes where a random walker is more likely to be trapped in. This concept can be treated as an increased flow circulation pattern between the nodes of the same community. We chose this algorithm based on its performance on networks with sizes around 100 thousand nodes [54, 89, 183]. We used the implementation of Edler, Eriksson and, Rosvall [15].

4.2.5 Community identification and persistence over time

To characterise each community of users, we utilised their reputation scores on the tags that were used to create its edges. This was done by summing the scores for tags shared by the users within each community (i.e. that were used to create the graph edges adjacent to that community). The summed scores were used to create a ranking of tags based on the total associated reputation score acquired by the community users. Using the top tags of each community ranking we are able to characterise the community by its core topics and technologies, and identify the part of the software industry that it represents.

For each year, we detected a number of communities in the user-tag graph. Characterisation using tags suggested that many of the bigger communities were present in almost every year, while smaller communities tended to be more volatile, appearing and disappearing. For the bigger communities we wanted to investigate their consistency in terms of the users that constitute them in each year. Our intuition is that a persistent community should appear as a pair of communities in consecutive years that have a large number of users in common and a low number of users in common with any other community. For each community in each year, we calculated the percentage of users in common with each of the previous year's communities, using the percentage overlap between the most-similar communities as a measure of their persistence. Persistence was also

[15]https://mapequation.github.io/infomap/

assessed qualitatively by the similarity of dominant tags over time. On chapter 4.3.6 Community Dynamics we describe the evolution of each detected community independently including when was its birth, when it was it's death if it stopped existing or other transformations from chapter 2.2.4 Community Life Cycle on Evolving Networks that might happened to it.

4.3 Results

4.3.1 Usage of Stack Overflow over time

Figure 4.1 displays the number of questions and answers posted on Stack Overflow for every month since August 2008 until December 2020, as well as the number of active users (defined as those users that created a post – question or answer – or received scores from up-votes on a historic post). The amount of posts increased from 2008 until 2014, with more answers than questions (perhaps to be expected since each question can have more than one answer). We observe a significant drop in the number of answers and questions posted after 2014. The number of active users making posts stabilises after 2015 while the number of users receiving up-votes continues to increase. During the same period, there is a decrease in the number of answers, while the number of questions reaches an equilibrium.

The reasons behind the drop in activity after 2014 are out of the scope of our work, but a possible reason is the creation at that time of the Stack Overflow Meta community [16]. Stack Overflow Meta is a forum where users discuss the workings and policies of Stack Overflow, rather than discussing programming itself. It is separated from the main question-and-answer site to reduce noise, while providing a legitimate space for people to ask how and why the Stack Overflow site works the way it does. A lot of questions more suitable for the Meta community were moved away from the main platform by the moderators, giving a plausible explanation for the significant drop in 2014.

There is a discussion on the Meta community [17] about the changes in user activity and what this means for the operation of Stack Overflow; we encourage the reader to refer to it. What is probably happening is that the community can handle a specific amount of questions each month. If the number of questions exceeds a threshold (about 180 thousand questions) the community gets saturated and many questions are not answered. This discourages users, especially new ones, from asking new questions and answer on other peoples questions, resulting on a decrease in answers and stable amount of questions. The amount of users creating new posts is increasing until 2014 and then it seems to stabilize on around 180 thousand users. The amount of users receiving score is increasing since 2008 and reaches about 400 thousand users. This means that older posts are

[16]https://meta.stackoverflow.com

[17]https://meta.stackoverflow.com/questions/256003/on-large-communities-decaying-over-time-being-nice-or-mean-and-stack-overflow/256084#256084

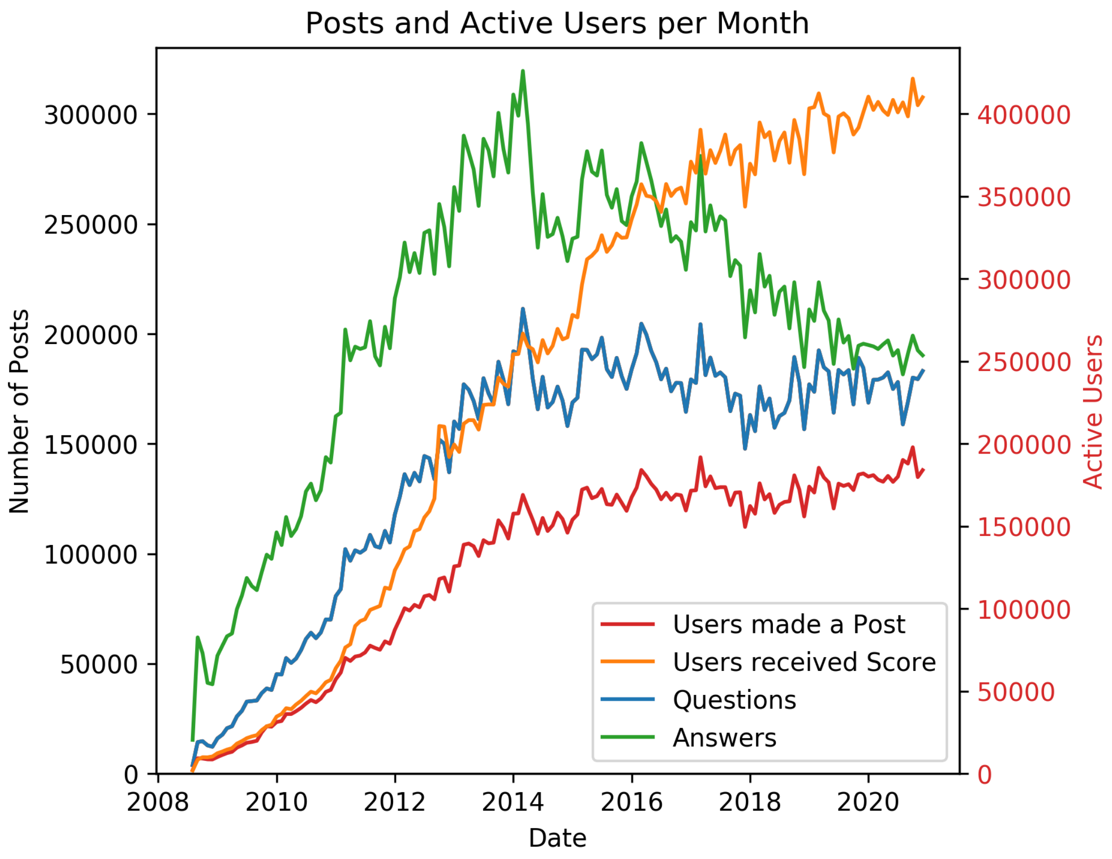


Figure 4.1: Number of posts and active users on Stack Overflow in each month from August 2008 to December 2020. Plot displays the number of questions and answers posted (left axis), and the number of users creating posts or receiving score via up-votes (right axis).

receiving votes from the users thus posts can be useful to the community for a long period of time, even tho the software development industry is rapidly evolving and changing. Finally we have checked and did not notice any data missing during the drop we are discussing about. The drop has been also detected by many other sources such as the stack overflow meta website we are referring on the footnote above and this discussion[18] as well.

4.3.2 Popularity of tools and technologies

To observe trends in the popularity of different tools and technologies in the software industry, we tracked the accumulated monthly votes given by Stack Overflow users to posts labelled with three categories of tags. Each time a question or answer received a vote, we incremented the score for every tag associated with the question. Scores for each tag were aggregated over monthly time periods. These scores show which tags (and by extension, which topics or technologies) are getting positive attention from the software development community.

Figure 4.2 shows tags representing popular programming languages. Trends show some similar patterns to those for overall numbers of posts (see above), with almost every tag showing a drop in attention from 2014, and most languages showing decreasing interest after 2017, similar to the number of answers (see Figure 4.1). However, the trends of languages relative to each other are interesting. The only languages that consistently increase are Python and R, perhaps reflecting the adoption of these languages for data analysis. Python becomes the top-ranked language by the end of 2020, after growing significantly to replace Javascript.

Figure 4.3 shows tags representing popular web programming frameworks. An interesting observation is the rise of AngularJS between 2012 and 2016, then its subsequent replacement by Angular. These two frameworks are open-source JavaScript frameworks and Angular is considered a direct alternative to AngularJS; this analysis suggests that Angular is replacing AngularJS for web development. Another framework that is rapidly growing is ReactJS. Well-known web frameworks like ASP.NET and Ruby-on-Rails seem to decline, while Django and Spring steadily increase since the beginning of the Stack Overflow platform.

Figure 4.4 shows tags relating to operating systems. The most dominant tag is Android followed by iOS, showing a bias to discussion of programming for mobile platforms. Surprisingly the Linux tag receives more attention than the Windows tag, suggesting that although the majority of personal computers in the world run the Windows operating system, the community of software developers on Stack Overflow is more interested in Linux.

[18]https://meta.stackoverflow.com/questions/320223/what-happened-to-stack-overflow-in-2014/320234#320234

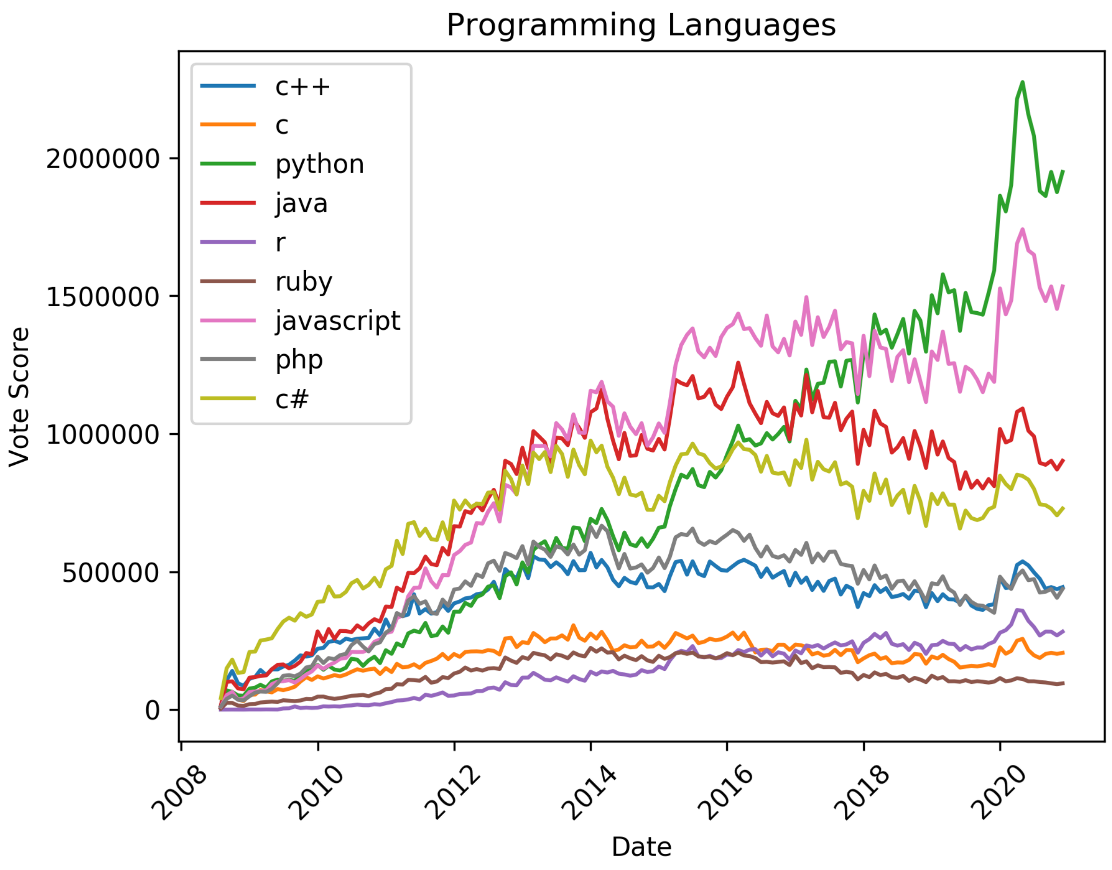


Figure 4.2: Trends in tag popularity: Programming languages.

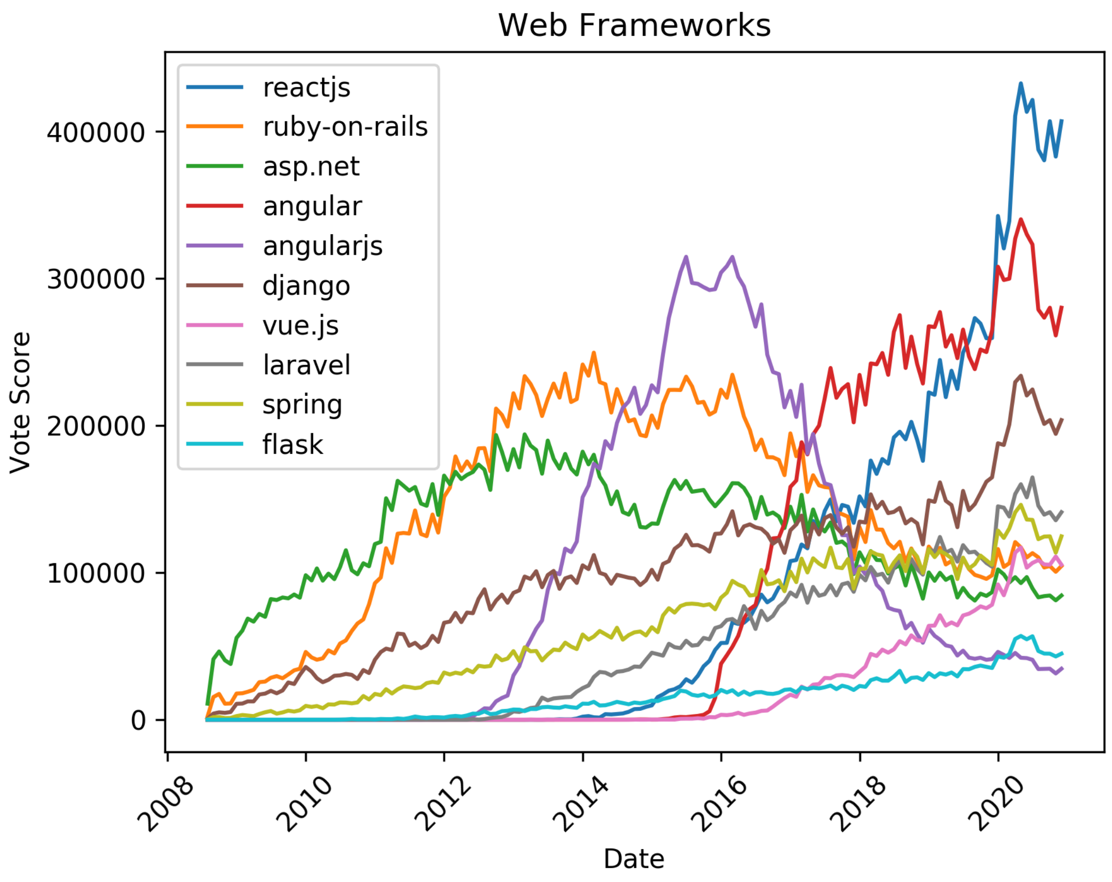


Figure 4.3: Trends in tag popularity: Web programming frameworks.

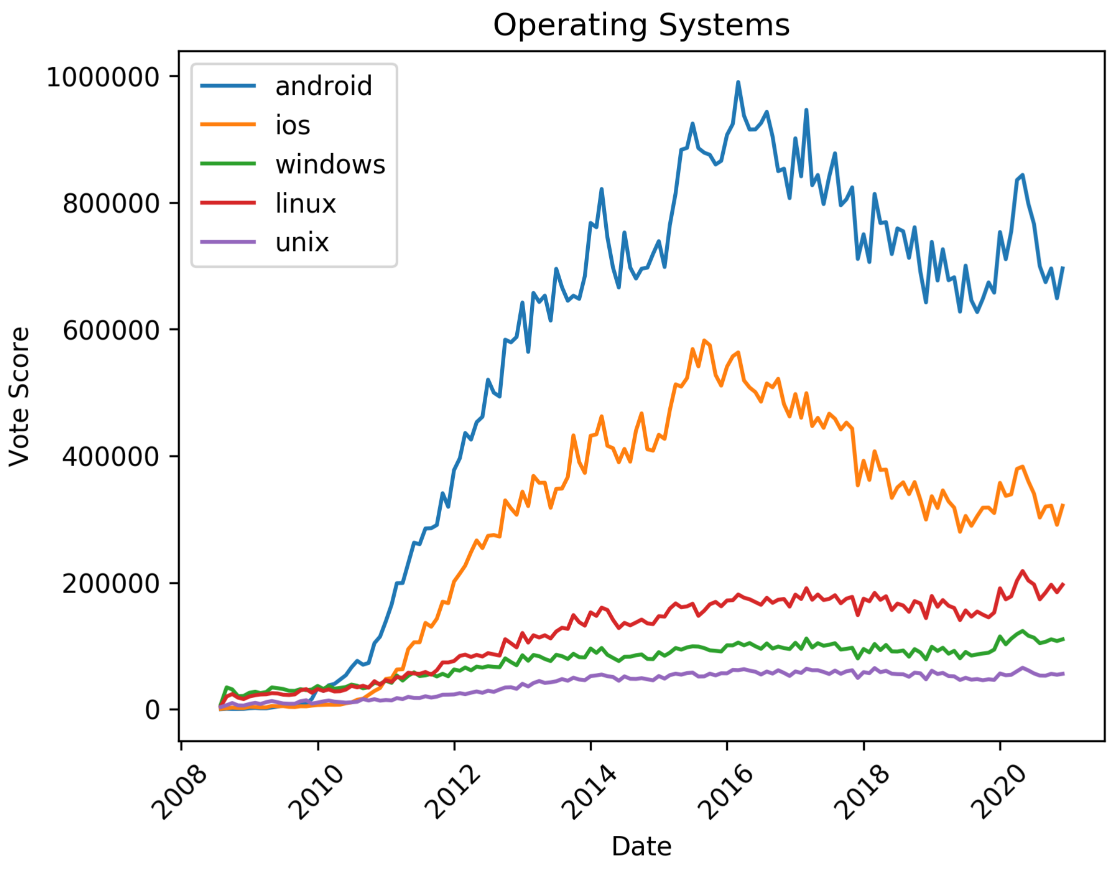


Figure 4.4: Trends in tag popularity: Operating systems.

4.3.3 User communities on Stack Overflow

Initially we created a graph using all the available data from 2008 until 2020. This graph is shown in Figure 4.5 (left), which displays the nodes/edges using the ForceAtlas2 continuous graph layout algorithm [80] from the Gephi visualization tool. Community detection was used to identify the 18 largest communities formed by Stack Overflow users, based on all interactions between 2008 and 2020; these are shown by node colours and annotated with a community label derived from their dominant tags. Figure 4.5 (right) shows the most frequent tags associated with posts by the users forming each community. Here each community is represented by a circle, using the same colour scheme as the left-hand plot for comparison, and sized proportionally to its number of members. Within each community (circle) the dominant tags are presented with size proportional to their usage. Table 4.4 displays network metrics for the detected communities. We can see that almost all communities have low density and about 6 of them have modularity higher than 0.4. this is a good indicator that those communities could be further split into sub-communities. We will further investigate this on future work. For calculating the modularity of each community separately we gathered the nodes of each community labeled by the community detection algorithm and connected them with edges from the original graph that were initially connecting those nodes. That way for each community we created a sub-graph and calculated its modularity.

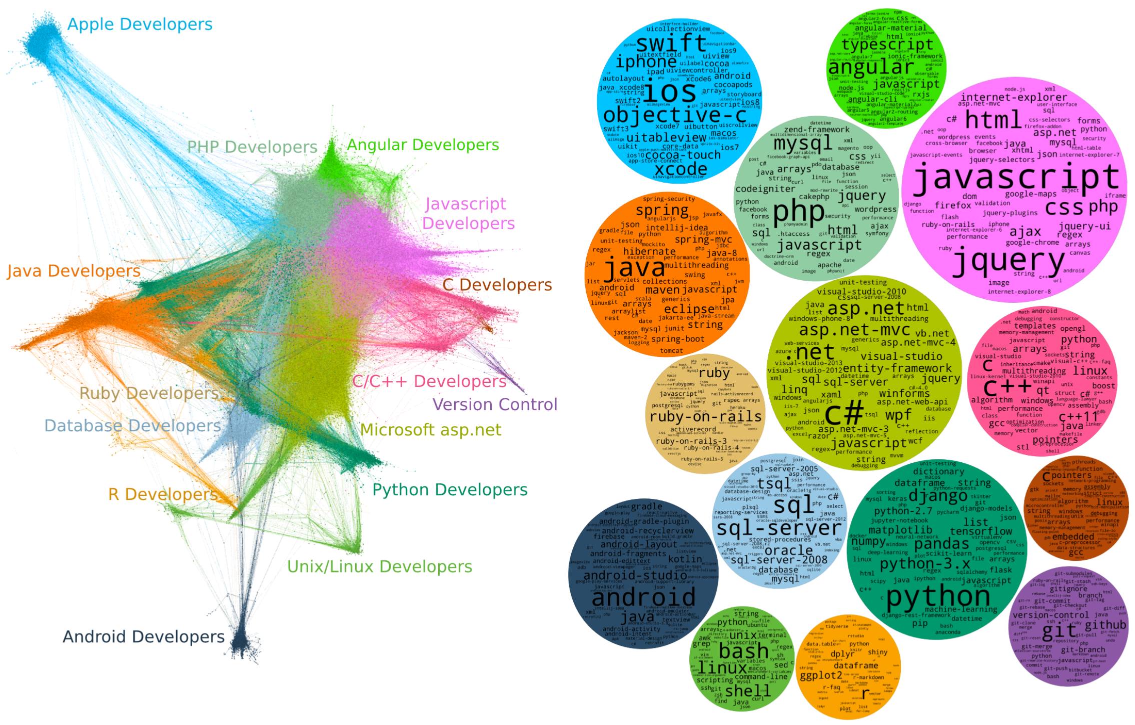


Figure 4.5: Stack Overflow community graph. User-tag interaction graph created from all available data between 2008 and 2020. Left: Communities of users within the Stack Overflow platform. Right: Tag clouds corresponding to each community showing the most frequent tags used by the users in that community.

The detected communities correspond to identifiable sub-fields of the software development industry, such as web developers or Android developers. The plots provide insights about the number of developers working in a specific sub-field, as well as the specific technologies that are popular in each sub-field. For example, some of the top tags of the Android community are android-layout, android-activity, android-intent, android-studio and sqlite. This provides information about which components, tools, IDEs and database engines are discussed on Stack Overflow in the context of the Android platform.

4.3.4 Studying user communities over time

To monitor how the developer communities evolve we created a user community graph similar to Figure 4.5 for every year between 2008 and 2020 inclusive. We then applied the Infomap community detection algorithm to each annual graph. For every detected community, its technological focus or field was determined using the dominant tags used by its members. Figures 4.6 and 4.7 present community tag clouds for the major communities found from 2008 (when the Stack Overflow website was launched) until 2020.

Our analysis shows a number of different communities present in each year. In total, we identified 16 user communities for the time period between the August 2008 and December 2020. There were also more communities detected by the InfoMap algorithm but their size was insignificant (less than 1% of the nodes in the complete graph), so these are omitted from further analysis. The

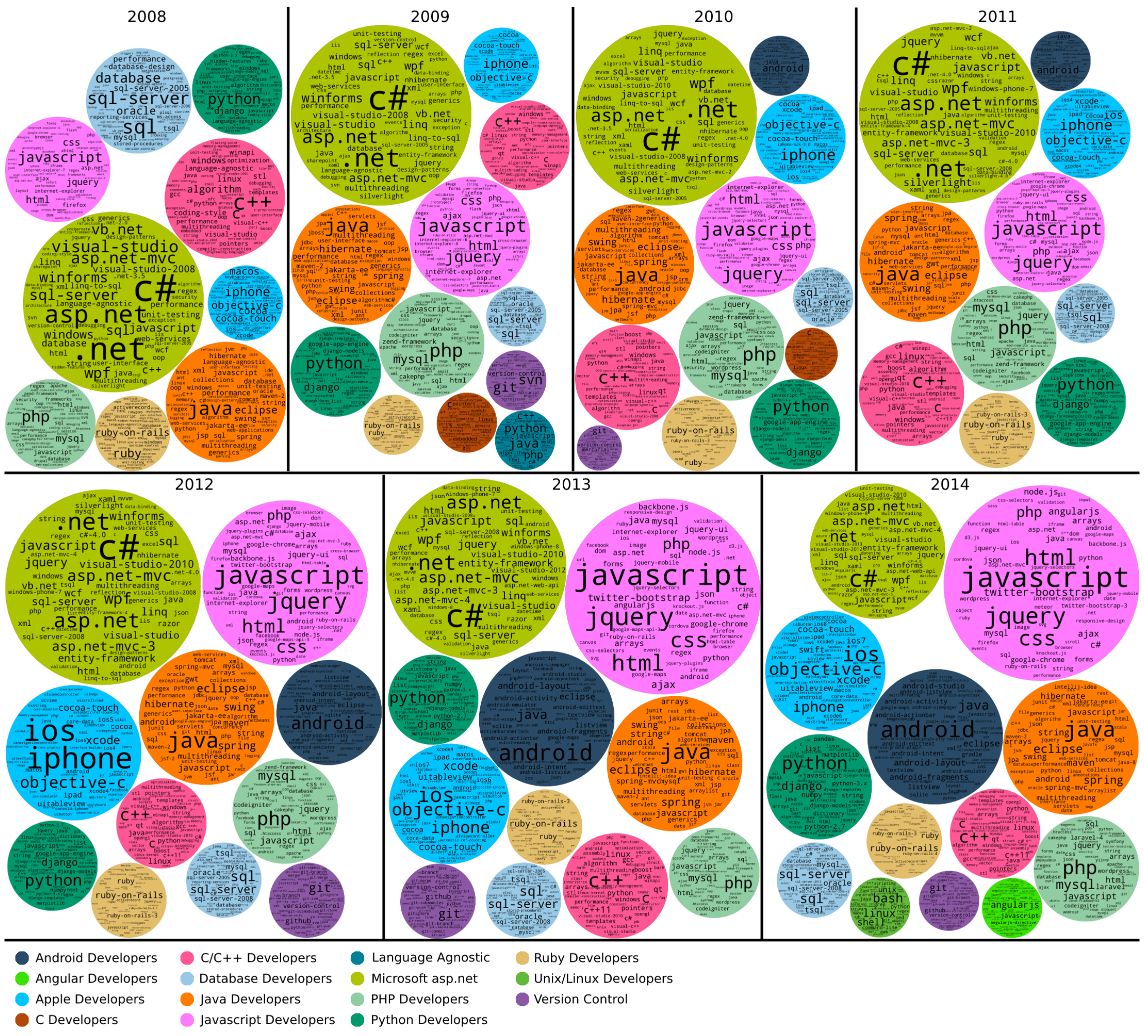


Figure 4.6: Evolution of the Stack Overflow communities from 2008 to 2014. Each circle shows a tag cloud with the most popular tags in a community identified in a given year. The size of the circles is somewhat proportional to the actual size of the communities, with some adjustments for layout.

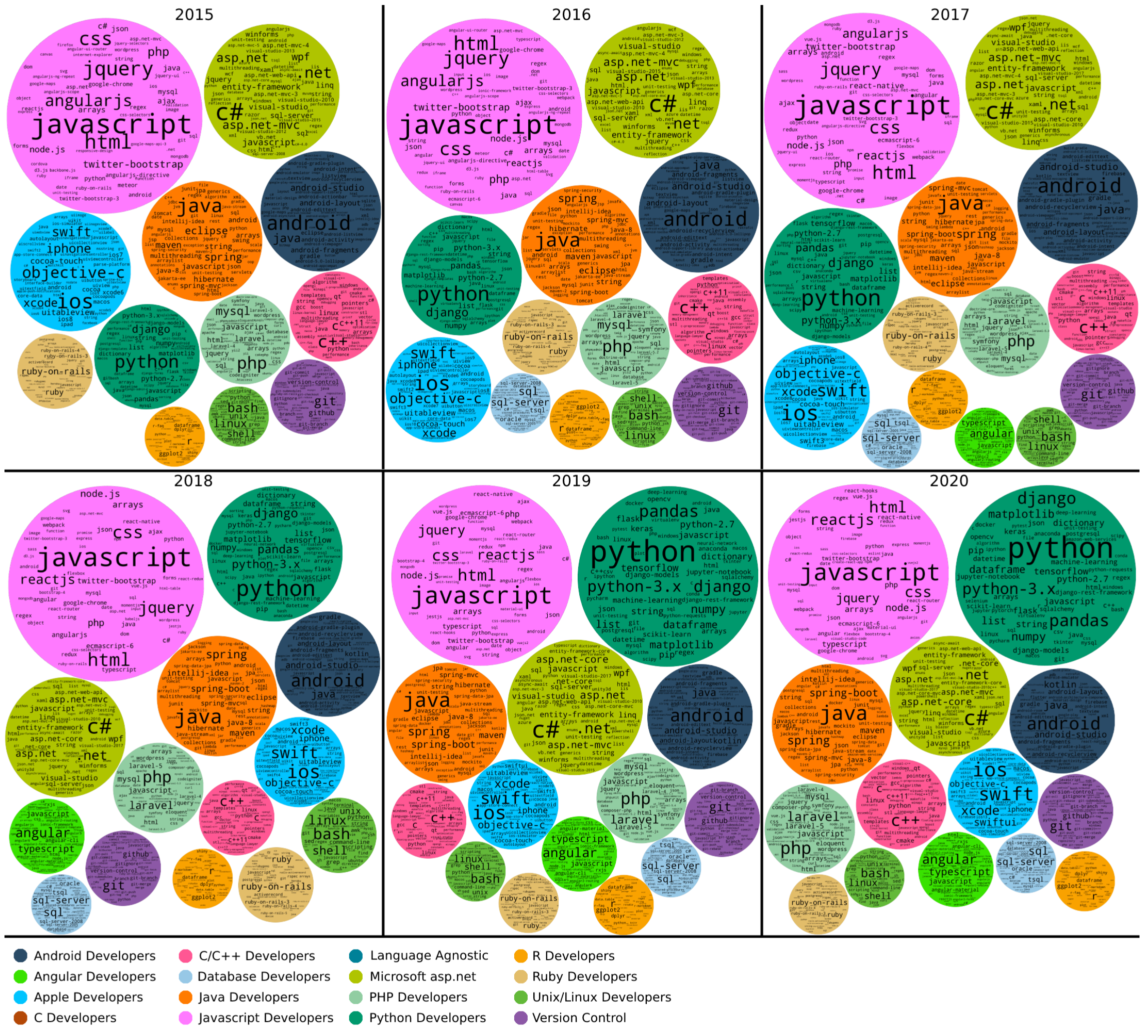


Figure 4.7: Evolution of the Stack Overflow communities from 2015 to 2020. Each circle shows a tag cloud with the most popular tags in a community identified in a given year. The size of the circles is somewhat proportional to the actual size of the communities, with some adjustments for layout.

time-resolved graphs show the same/similar communities as the time-aggregated graph of Figure 4.5; the exceptions are the Laravel community, which is not identified in the yearly graphs, and the Testing Developers community, which is only seen in the 2008 graph.

Each community can be characterised by the dominant tags associated with the members of each community. Table 4.2 shows the dominant tags for the 16 communities found across all graphs that were created, including the time-aggregated graph created with the accumulated data from 2008 to 2020 and the time-resolved annual graphs. Focusing on the users gives a view of the persistence of user populations working with particular technologies, how users acquire score, and how users migrate between technologies. Focusing on the tags associated with each community gives a view of the technology clusters that make up the wider programming and software development ecosystem, including which technologies are typically clustered together in practise, how new technologies arise and are getting adopted, while old technologies fall out of use.

4.3.5 Community persistence over time

To further test whether the apparent persistence of communities shown in Figures 4.6 and 4.7 is genuine, we examined user behaviour in terms of whether they stay active in the same community (i.e. continue to ask or answer questions related to their current community), migrate to another community (i.e. begin asking/answering questions in another community), or go inactive (i.e. stop asking/answering questions anywhere on the Stack Overflow platform). For clarity, we define an inactive user as one that stops receiving any additional score from the platform for a given time period; this does not necessarily mean that the user stops using the Stack Overflow website (perhaps as a silent viewer), but that they both stopped creating posts and did not receive votes/responses to earlier posts. This definition views "activity" as a combination of the contribution a user makes to the platform and the acknowledgement (reputation points) they receive from it.

Figure 4.8 shows the average movement rates of users in/out/within the Stack Overflow platform. The top node shows new users moving into existing communities. The bottom node shows users staying active within an existing community, or moving from existing communities to either inactivity or to other communities. On average, $\sim$49.07% of the users of a community stay in the same community the next year, $\sim$13.41% migrate to another community, and $\sim$37.52% of the users go inactive. Also for each year (except 2008) we observed that $\sim$40.18% of the current users of a community were completely new to the platform.

Figure 4.9 shows the average percentage overlap between every pair of communities between consecutive years. The persistence of the user membership of each community is given by entries on the diagonal; the similarity of each community to itself from year to year. The off-diagonal entries indicate the overlap of users between each pair of communities, indicating the flow of users between communities. Between-community flows are typically small.

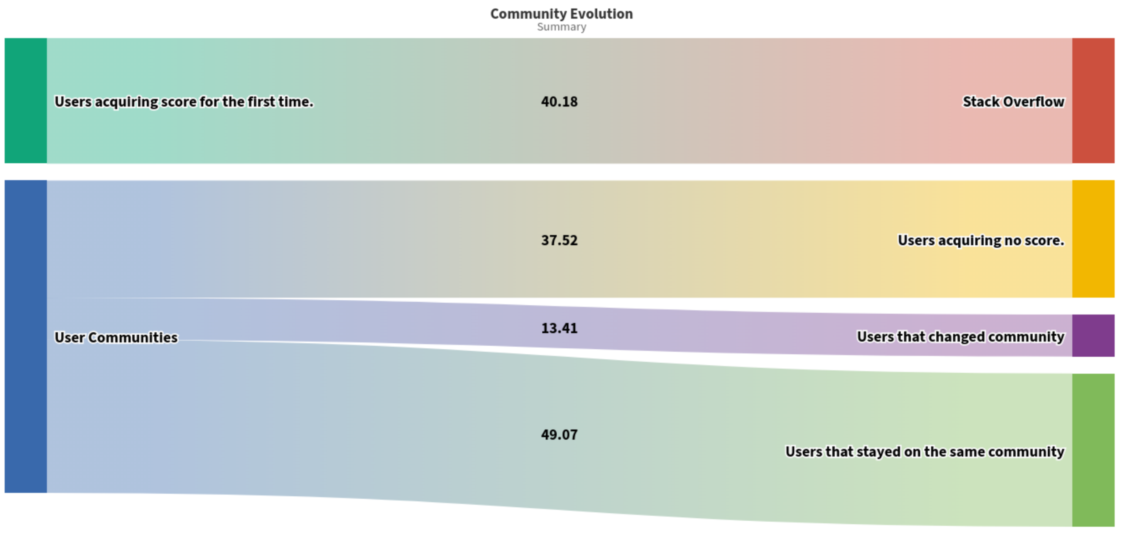


Figure 4.8: Movement of users in/out/within Stack Overflow. The top node indicates the flow of new users into the platform (users that were not present in the previous year). The second node indicates flows of users that become inactive (i.e. do not acquire any score on the next year), migrate to another community within the platform, or stay in the same community in consecutive years.

The identified communities we discuss have large percentage self-similarity and can therefore be seen as persistent from year to year. Note that the percentages shown exclude the (average) 37.52% of users that go inactive in each year, so the percentages displayed in Figure 4.9 relate to the average of 62.48% of users that kept acquiring vote score..

Another way to confirm the consistency of the formed communities is a qualitative comparison of the dominant tags for each community from year to year. Figures 4.6 and 4.7 show that the dominant tags for each identified community are coherent between years. Note that consistency does not require the exact same tags to be present in each year, but allows for tags that are associated with the same software field. For example, in the Apple developers community the objective-c tag gets smaller as time passes, while the swift tag is growing; this reflects the decision by Apple to introduce a new programming language for application development in their platforms.

4.3.6 Community dynamics

Early in history of the platform, (e.g. in the 2008 community graph (Figure 4.6)) we see that only a few communities were formed with the two largest being The Microsoft asp.net and the Java developers communities. Later in the platforms development, many more communities formed and these are more specialised. Figure 4.10 shows the sizes of the different communities over time. Table 4.2 shows the major tags in each community. Table 4.3 displays the tags that are related with the top tag of each detected community given by the Stack Overflow website. We can observe that those tags are very similar with the tags of table 4.2. Below we describe the main events for the 11 largest and most persistent communities over time, which are also shown in Figure 4.10 and characterised based on community graphs and tags.

Community Name	**Top Tags**
Android Developers	android, java, android-layout, android-studio, android-fragments, android-activity, listview, android-intent, xml, eclipse, json, sqlite, android-recyclerview, gradle, android-edittext
Angular Developers	angular, javascript, typescript, html, css, rxjs, angularjs, node.js, ionic-framework, angular-material, ionic2, observable, angular-cli, android, jquery
Apple Developers	ios, objective-c, iphone, xcode, swift, uitableview cocoa-touch, ipad, cocoa, macos, uiview, android, uiviewcontroler, core-data, ios7
C Developers	c, linux, embedded, pointers, gcc, arrays, string, windows, algorithm, assembly, function, unix, struct, performance, memory
C/C++ Developers	c++, c, arrays, c++11, linux, java, python, string, pointers, algorithm, c#, gcc, multithreading, templates, windows
Database Developers	sql-server, mysql, database, tsql, oracle, sql-server-2008 c#, php, stored-procedures, join, java, postgresql, select, sql-server-2012
Java Developers	java, eclipse, spring, maven, intellij-idea, spring-boot, java-8, hibernate, spring-mvc, string, json, javascript, jpa, arrays, amdroid
Javascript Developers	javascript, html, css, jquery, reactjs, node.js, arrays, twitter-bootstrap, ecmascript-6, typescript, redux, express, angularjs, ajax, webpack
Language Agnostic	python, java, php, c++, c, javascript, c#, linux, string, algorithm, arrays, language-agnostic, .net, regex, performance
Microsoft asp.net	c#, .net, asp.net, javascript, asp.net-mvc, sql, sql-server, jquery, html, linq, visual-studio, winforms, entity-framework, css, wpf
PHP Developers	php, laravel, mysql, javascript, laravel-5, html, arrays, jquery, wordpress, eloquent, css, sql, laravel-4, json, string
Python Developers	python, django, pandas, python-3.x, numpy, list, matplotlib, tensorflow, python-2.7, dataframe, string, pip, dictionary, javascript, machine-learning
R Developers	r, dataframe, ggplot2, dplyr, plot, data.table, shiny, list, regex, r-markdown, tidyverse, function, loops, matrix, string
Ruby Developers	ruby-on-rails, ruby, ruby-on-rails-3, javascript, activerecord,jquery, ruby-on-rails-4, rspec, git, html, css, devise, postgresql, heroku, mysql
Unix/Linux Developers	bash, linux, shell, unix, python, awk, c, java, c++, git, regex,grep, javascript, sed, arrays
Version Control	git, github, version-control, git-branch, gitignore, branch, javascript, git-merge, git-commit, git-push, git-stash, ssh, git-submodules, bash, commit

Table 4.2: Community characterisation. The tags with highest scores for each community are shown. The table displays 16 communities which is the union of the communities detected on the accumulated graph from 2008 until 2020 and all yearly graphs. The ranking is a result of aggregating all tag scores for each year since 2008 for each detected community. A description for each tag can be found on Stack Overflow (https://stackoverflow.com/tags).

Related Tag	**Top Tags**
Android	java, android-studio, android-layout, kotlin, android-fragments, firebase, listview, xml, android-intent, sqlite, android-activity, android-recyclerview, json, eclipse, cordova $\times$ 21624
Angular	typescript, javascript, html, angular-material, rxjs, ionic-framework, css, node.js, observable, firebase, ionic2, angular-cli, angular6, json, angular2-routing
Apple	swift, objective-c, iphone, xcode, uitableview, android, ipad, cocoa-touch, core-data, uiview, uiviewcontroller, firebase, javascript, uicollectionview, react-native
C	c++, arrays, linux, pointers, gcc, struct, string, sockets, function, malloc, windows, multithreading, file, scanf, linked-list
C/C++	c++11, qt, c, templates, arrays, boost, pointers, windows, vector, opencv, winapi, multithreading, linux, visual-c++, string
Database	mysql, sql, php, java, sql-server, c#, sqlite, android, postgresql, database-design, python, oracle, mongodb, javascript, django
Java	android, spring, swing, spring-boot, eclipse, hibernate, arrays, maven, multithreading, json, xml, string, spring-mvc, jpa, mysql
Javascript	jquery, html, css, reactjs, node.js, php, angularjs, ajax, arrays, json, typescript, angular, vue.js, regex, asp.net
Language Agnostic	algorithm, math, oop, data-structures, design-patterns, java, programming-languages, performance, c#, arrays, python, geometry, regex, sorting, c++
Microsoft asp.net	c#, asp.net, winforms, vb.net, wpf, visual-studio, linq, asp.net-mvc, entity-framework, wcf, .net-core, sql-server, multithreading, windows, asp.net-core
PHP	mysql, javascript, html, laravel, jquery, wordpress, arrays, ajax, codeigniter, sql, json, symfony, laravel-5, forms, regex
Python	pandas, python-3.x, django, numpy, dataframe, list, python-2.7, matplotlib, tensorflow, dictionary, tkinter, flask, selenium, regex, arrays
R	ggplot2, dataframe, dplyr, shiny, plot, data.table, loops, tidyverse, function, for-loop, matrix, r-markdown, list, regex, rstudio
Ruby	ruby-on-rails, ruby-on-rails-3, ruby-on-rails-4, rubygems, activerecord, rspec, arrays, regex, javascript, hash, sinatra, devise, heroku, postgresql, json
Unix/Linux	shell, linux, bash, c, awk, sed, grep, python, java, c++, macos, scripting, regex, perl, terminal
Version Control	git, svn, github, mercurial, tfs, merge, visual-studio, branch,t ortoisesvn, bitbucket, repository, dvcs, xcode, eclipse, perforce

Table 4.3: Related Tags from the Stack Overflow website. The related tags with each one of the communities we detected taken from the Stack Overflow website. A description for each tag can be found on Stack Overflow (https://stackoverflow.com/tags).

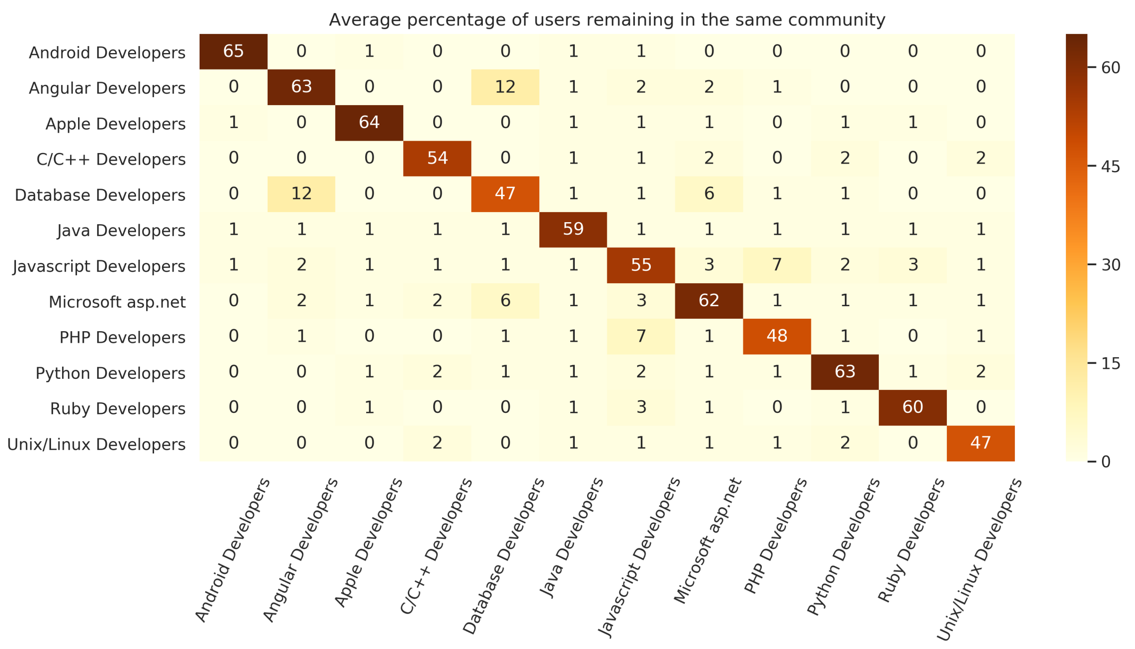
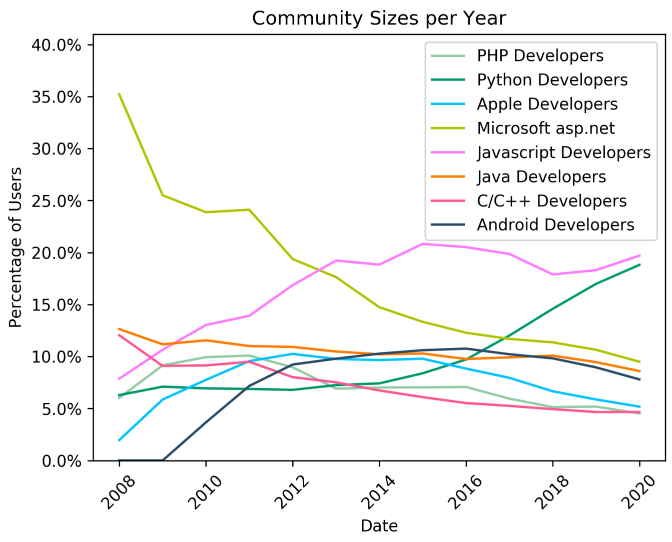


Figure 4.9: Heat map of year-to-year pairwise similarity of community membership. The heat map displays the average percentage overlap of users belonging to each pair of communities in consecutive years between 2008 and 2020. Diagonal entries indicate self-similarity and community persistence. Off-diagonal entries indicate flows between communities.

Community Name	**Density**	**Modularity**	**Avg. Degree**	**Avg. Weighted Degree**
Android Developers	0.01	0.383	22.4	284.9
Angular Developers	0.01	0.263	28.7	356.5
Apple Developers	0.006	0.41	23.4	289.5
C Developers	0.004	0.392	6.1	76.3
C/C++ Developers	0.003	0.681	9.16	114.2
Database Developers	0.019	0.322	53.2	680.8
Java Developers	0.009	0.425	27.1	340.7
Javascript Developers	0.006	0.351	47.4	590.2
Microsoft asp.net	0.007	0.345	61.3	762.6
PHP Developers	0.014	0.21	95.8	1195.6
Python Developers	0.005	0.61	19.2	227.7
R Developers	0.005	0.421	9.5	117.2
Ruby Developers	0.01	0.304	19.9	247.3
Unix/Linux Developers	0.004	0.736	7.9	99.2
Version Control	0.005	0.411	6.8	79.5

Table 4.4: Network metrics of the detected communities.



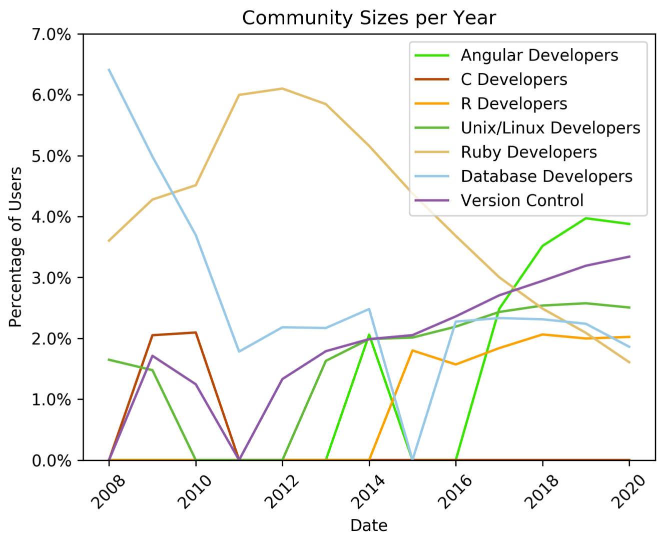


Figure 4.10: Evolution of community sizes over time. The time series present the percentage of all users on Stack Overflow that each detected community contains in each year from 2008 to 2020. Plots for the 15 largest communities are shown, smaller communities are omitted for clarity.

Android Developers. The birth of this community was in 2010 and continues until 2020. Users discuss basic components of the Android operating system (layout, fragments, activities), programming languages and developing tools. Top tags include: android, java, android-layout, android-studio, android-fragments, android-activity and listview. Java is the main language for Android development which explains the second-place ranking of this tag. The Android Developers community reached its peak on 2016 where 10% of the total users were part of this community (Figure 4.10).

Apple Developers. This community was born in the beginning of the Stack Overflow platform and continues until 2020. Users discuss developing applications on the iOS operating system for mobile devices and the macOS system for personal computers. Dominant tags include ios, objective-c, iphone, xcode, swift, uitableview cocoa-touch and ipad. In Figures 4.6 and 4.7, we can observe that the objective-c tag starts to decline after 2016 and the swift tag emerges. The Swift programming language was introduced by Apple in 2014 as a replacement for objective-c in all Apple platforms.

C/C++ Developers. This community continues since the beggining of the Stack Overflow website. Users primarily discuss the two languages and related tools, as well as other programming languages such as Python or Java to a much lesser extent. Some of the top tags of this community are c++, c, arrays, c++11, linux, java, python, string, pointers, gcc. GCC (GNU Compiler Collection) is the standard (and most well known) compiler for C and C++ on Linux and supports many other languages and platforms as well.

Database Developers. The Database Developers community was born on 2008 which was also its peak including 6% of the total users. The community almost persists until 2020 with the exception of 2015. Users on this community are discussing about database related topics and technologies such as sql-server, mysql, join and select operations.

Java Developers. This is another community that was born in 2008 and continues until 2020. Users discuss the language itself (versions, libraries), data structures and data types, as well as environments and platforms. Prevalent tags include: java, eclipse, spring, maven, intellij-idea, spring-boot, java-8 and hibernate(Table 4.2). Eclipse is one of the most well known integrated development environments (IDE) for Java.

Javascript Developers. This community was born in 2008 and grows in every year until 2015 where its size stabilizes on 20% of the total users of the network. This community is defined by use of the Javascript language and related frameworks (node.js, jquery, ReactJS). Users discuss the language and related tools: javascript, ReactJS, node.js, html, css, jquery, react-native, arrays, typescript, and ecmascript-6.

Microsoft asp.net Community. The asp.net community is the biggest community from 2008 until 2012. It remains big until 2013 and is still noteworthy until 2020. Some of the top tags of this community, shown on table 4.2, are c#, .net, asp.net, javascript, asp.net-mvc, sql,

sql-server, and jquery. Those tags show the focus of this community on web development using the .NET ecosystem developed by Microsoft.

PHP Developers. This community was born in 2008 and continues until 2020. The pick of its popularity was on 2010 and 2011 including 10% of the network users. The community top tags are php, laravel, mysql, javascript, laravel-5, html, arrays, jquery, and wordpress.

Ruby Developers. This community was born in 2008 and continues until 2020. The Ruby community is a web development community focused on the Ruby programming language and the associated web framework Ruby-on-Rails. Top tags of the community are ruby-on-rails, ruby, ruby-on-rails-3, javascript, activere-cord,jquery, ruby-on-rails-4 and rspec (Table 4.2). Community size peaks in 2012 at about 6% of all platform users then declines to about 2% of users in 2020 (Figure 4.10).

Unix/Linux Developers. This community concerns the Unix and Linux operating systems. Users discuss the operating systems, associated tools and programming languages. Top tags include: bash, linux, shell, unix, python, awk, c, java, c++, and git. Bash is a command language for the Unix shell; AWK is a domain-specific language used as a data extraction and reporting tool; Git is a version-control system for tracking changes in source code during software development.

Python Developers. This community was born in 2008 and continues until 2020. Top tags are python, python-3.x, python-2.7, numpy, list, pandas, dictionary, string, arrays, matplotlib. Numpy, Pandas and Matplotlib are Python libraries for data analysis. In 2020, the community reached its peak including almost 20% of the network users.

4.4 Discussion

In this study, we analysed a large amount of data from the Stack Overflow platform to study the communities formed by users around different technologies in the software development and computing industries. By associating users with the tags by which they gathered most ‘reputation’, we formed user profiles that reveal their technology use and expertise. Aggregating these profiles into user-technology graphs (in which two users are connected by the number of tags they share) we were able to detect communities of users. Examining the popular tags within each community allowed us to characterise each community based on its dominant technologies/tools. By repeating the network formation and community detection process, we examined the evolution of the Stack Overflow community over an extended period from 2008 until 2020. To the extent that Stack Overflow is reflective of the wider software and computing industry, trends uncovered in this analysis reflect trends in the digital industries.

Our study reveals a number of insights about the Stack Overflow platform. First, the platform shows strong community structure, with different user populations engaged in discussions around different technology clusters. We were able to detect implicit communities of users specializing in different aspects of software development (e.g. Web Developers, Python Developers, Linux/Unix Developers, and so on). One finding from this is the central position of the World Wide Web and related technologies on software development in the period 2008 to 2020. On average 30% of platform users were members of the Web Developers community in each year in this period, while there were also a number of other web-related communities focused on particular languages or web frameworks (e.g. C#-ASP.NET, Python-Django). Our analysis also explored the movement of active users between communities or in/out of the platform. We found that every year about 40% of the users acquiring score are completely new to the website. Users belonging to a community are likely to either stay in the same community or receive no score for the given year, rather than migrate to a different community within the platform. Finally, we observed an overall pattern whereby the Stack Overflow has changed slowly from a 'generalised' discussion, with a small number of larger and more diverse communities, to a more 'specialised' mode with a larger number of smaller communities each with a particular focus.

Our findings provide significant insights for professionals working in the computing and software development industries. The communities we identify represent clusters of technologies, and by extension, skills and technical competence. A human resource manager might use this information to tailor job advertising and hiring campaigns, or create appropriate job offers. Evaluation of potential employees would benefit from improved competency profiling processes [111], enabled by using the observed technology clusters to get a better understanding of skills associated with different job roles. A student or a software enthusiast can choose the right technology to start his/her career in software development by examining the size and growth of different communities. Businesses in the tech industries can also benefit from such analysis by being able to better assess a new market; knowing the groupings of technologies and programming languages, and the growth/decline of these groupings, might guide better investment decisions. Software developers can recognize emerging or decaying technologies and adapt to this rapidly evolving industry. Also the grouping of different technologies into communities provides a guide of which groupings of technologies perform better together, providing useful information to system architects.

From an academic perspective, to our knowledge this study presents the first large-scale analysis of community structure within the Stack Overflow platform. Here we showed that weighted networks describing the indirect interactions between users (mediated through their shared usage of the same tags) can be constructed, then interrogated using community detection methods to identify meaningful groups of users. This approach can be expanded in future work to examine (e.g.) clustering of technologies, individual trajectories along a career path, evolution of technological trends, amongst other themes. Stack Overflow is an intriguing example of a complex sociotechnical system [164]. The full user-technology interaction network is a large object with rich structure; the community detection process we used to identify meso-scale patterns within the network is only one kind of analysis that might be applied. While here we tracked network dynamics using a simple one-year time increment, it would be possible to perform a more complex temporal analysis to explore trends at a variety of scales. Stack Overflow is also a rich resource for the study of collaborative work and crowd-production of knowledge. Here we have shown that users tend to stay within a particular segment of the knowledge base. Such findings might help inform efforts to design better systems to manage online collaborative knowledge production. Social scientists can observe how users adopt new skills, learn new ones and abandon skills they already had. They can also observe the life cycle of communities of individuals in the field of software development and monitor how users respond to new technologies or technologies that are getting obscure. Lastly, by investigating how consistent the detected communities are in terms of common users for each year and in terms of software development field they correspond, we demonstrated how community detection methods (infomap in our case) can produce meaningful communities using real life data that were rapidly changing over time.

Chapter 5

Discussion

On Chapters 3 and 4 we have seen how we can utilize evolving networks to acquire useful information from heterogeneous document streams such as news articles, and Twitter posts. We were also able to see how we can discover implicit relations and communities in online social platforms such as Stack Overflow, monitor their evolution over time and detect their main characteristics.

To motivate our investigation in applications of evolving networks we defined the following set of research questions in Chapter 2.

- RQ1: Can we detect breaking news events from heterogeneous input streams by utilizing social networks analysis and evolving networks?
- RQ2: How can we extract the summary of the context of a detected event?
- RQ3: Can we extract useful insights from online communities using social network analysis on evolving networks?

Working on those questions we initially discovered and later demonstrated that research on social interactions between entities on social networks and platforms can significantly benefit by utilizing evolving networks instead of static networks. Evolution over time is a fundamental aspect of social media and social networks, thus studying evolving networks can reveal more information about this aspect. We will now summarise the main findings from chapters in this thesis in the context of these research questions.

5.1 RQ1: Can we detect breaking news events from heterogeneous input streams by utilizing social networks analysis and evolving networks?

Detecting events on news streams is a challenging task. The volume, velocity, diversity of generated news content around the globe is enormous and the task of handling this amount of information sets significant restrictions on what analysis can be performed. On top of that, in most cases proper processing of the incoming information should take place in order to identify the main linguistic entities that are needed to perform event detection. Such actors could be named entities, noun phrases and parts of speech. By utilizing networks we can generate representations where each actor of interest is a network node and relations between actors are network edges. By doing that we drastically reduce the size of data we have to process and create a structured representation of them. Also the main advantage of this approach is that it provides a better measurement technique of "popularity" of an actor on the news. By calculating centrality metrics on the nodes and edges, such as weighted degree, betweeness centrality or closeness centrality, we can easily detect "important" actors in the news stream not by just counting their frequency of appearance on the text but also by taking into account with whom they appear, relate or react. So for example an important actor is not only an actor that appears frequently, but also one that is related to, or interacts with other important actors. For event detection, creating an evolving network and monitoring its metrics over time, utilizing a moving window and setting a threshold to detect peaks of activity, provides a straight forward and easy to implement methodology to detect peaking events on a heterogeneous stream of news documents. Overall social network analysis on evolving networks is a potent approach for this task since it provides a way to reduce the load of data we have to process, contributes better metrics to measure "popularity" or "importance" and makes easy to monitor and detect peaks on the evolution of actors "popularity".

5.2 RQ2: How can we extract the summary of the context of a detected event?

Extracting the context of the detected events can provide us with further information about the event and also generate a comprehensive summary that describes it. There are many approaches that can handle this task like LDA [20] or other topic modelling methods, but as we have shown in Chapter 3 we can achieve better results by applying social network analysis on a knowledge graph created by named entities, nouns and noun phrases. During our research on this field, we discovered and demonstrated that named entities are suitable for monitoring and detecting events on news streams, while nouns and noun phrases combined with named entities are more suitable for describing those events. By applying community detection on the knowledge graph we can extract the summary of each event from the formed communities of the network. Each community represents an event and the nodes of the community generate its summary.

Another useful dimension when we want to describe an event and discover its context is the sentiment that people associated with it. In many cases, even if we understand what a specific event is about it is not always obvious how the public feels about it. News articles aren't a good source to extract such information since the language used for reporting news often has a neutral sentiment. However, the majority of tweets made by ordinary users very often express their personal opinion, emotions and sentiment when they talk about an event. In our methodology, we used the Vader [76] classifier to extract the sentiment of each detected event from the tweets that discuss it. That way, for each event we get a sentiment distribution between -1 (negative sentiment) and 1 (positive sentiment) that represents the overall sentiment the people discussing this event have. We tested this methodology on three data sets of tweets. The first one was a collection of tweets from the US Presidential Election in 2012 by Aiello et al. [4] containing approximately 3.6 million tweets. The second one was a set of tweets collected during the US Presidential Election in 2016 by Littman et al. [96] containing approximately 5.4 million tweets. The last one is a collection of tweets collected by our research group during the UK General Election in 2019 which contains 4 million tweets. Finally we can extract more insights about the context of a detected event by identifying the emotions related to it, beyond just positive negative or neutral, for example Happiness, Sadness, Surprise, Anger and Fear. For example, in Figure 3.16 we can see the summary, sentiment distribution and emotions percentages of tweets discussing the first death from COVID-19 in Hong Kong.

5.3 RQ3: Can we extract useful insights from online communities using social network analysis on evolving networks?

Social network analysis on evolving networks can also provide valuable insights about how communities form on online social platforms and how they evolve over time. An example of this is presented in Chapter 4. On the Stack Overflow website programmers and computer scientists discuss software development problems they encounter in their work. Depending on the topics that each person discusses, a network can be created where each user of the platform is a node and the common interests between two individuals form a weighted edge. In this work, we investigated the communities in this kind of networks for each year from 2008 to 2020. The majority of the detected communities were clusters of developers of a specific programming language, framework or field (for example, web developers). The major communities were the Microsoft asp.net developers community which was the biggest community on the platform from 2008 until 2012, the Javascript developers community which was prevalent on Stack Overflow form 2013 until 2018 and the Python developers community which emerged on 2019 until 2020. Other important communities were the developers of the Apple ecosystem (iOS, MacOS), Android developers, Java developers and C/C++ developers. For the Microsoft asp.net community the popular tags were .net, C#, asp.net and asp.net-mvc. For the Javascript developers community the most popular tags were javascript, jquery, html, and css which also indicates that the javascript language is widely used in web development and applications. The most popular tags in the Python developers community were python, django, pandas and python-3.x. Finally, we were able to monitor how the most popular tags in three fields of software development were evolving over time. Such fields were programming languages with python, javascript and java being the most popular languages, web frameworks where reactjs, angular, angularjs and django were the most popular and operating systems where android and ios were the most popular.

This method provided insights about the software development community. Such insights could be useful to human resources professionals that are looking for developers on specific positions. The dominant tags for each detected community provide information on what skills are most used by the industry on several positions. For example if a company is looking for a web developer, they could look for individuals that have experience on *Javascript, ReactJS, JQuery* and *Node.js.* Also individuals that want to train as software developers could discover which fields are most popular and what skill they could combine to be more competitive in that field. Finally, from the evolving network we can also find out which technologies are currently popular, which are growing in popularity and which are declining. That way professionals could train on skills that will rise in demand in the future and avoid skills that are becoming irrelevant on the industry.

5.4 Future Work

There are a number of ways that the NED methodology could be improved and extended. Given that for the summarization of the detected events community detection is derived from a knowledge graph, it would be interesting to explore if the detected communities contain one or more events. Removing a community from the original graph and measuring the modularity of the associated sub-graph could indicate if we could extract more communities that describe different events or sub-events within a wider narrative. Another possibility is to experiment with the parameters of different community detection algorithms such as the resolution of the Louvain algorithm [21] which finds on more or fewer detected communities depending on its value. Natural language processing is an essential part of the NED methodology both for detecting named entities and nouns and noun phrases. Although the classifiers used to address this task are some of the most well performing, creating custom classifiers for different kinds of text could significantly benefit the performance of the method. For example, text from tweets is different than text from news articles or blog posts since it usually contains jargon, typos and abbreviations. Further investigation could also be done in choosing other network measurements for monitoring how entities popularity evolves over time. Such measurements could be node level such as closeness or betweeness centralities or maybe meso-level features such as dynamic community structure.

Another direction we would like to explore is to repeat the work of Chapter 3 about the COVID-19 pandemic for a wider time period. By monitoring the peaking events for a bigger time period we can better understand what the users of Twitter were discussing about and how the sentiment and emotions about the pandemic were evolving over time. There are a number of challenges completing this task since the amount of the input data as well as the amount of the produced results require better planning and visualization approaches. Those approaches could include an interactive tool that could display all detected events based on their date and more information about them such as the keyword summaries, sentiment distribution and emotions pie charts.

In the work done on the Stack Overflow website we were able to detect communities of software developers, monitor how they evolve over time and identify key elements of those communities. Stack Overflow is a question and answer website that belongs to a larger group of QnA websites called Stack Exchange. This group contains websites similar to Stack Overflow but with different themes[1], for example, mathematics, video games, role playing games or operating systems. For future work it would be interesting to perform the same analysis on some of them to get better insights about each platform. Another direction for future work is to analyse the questions and answers of the website and try to determine when a question is about a bug or a problem about a technology or programming language. This will help us understand whether the reason of a tag accumulating a lot of up-votes on the platform is because the tag is problematic or popular on the developing community.

5.5 Conclusion

In this Thesis, we demonstrated how we can utilize social network analysis on evolving networks to detect events in real time from heterogeneous sources. The NED methodology produced satisfactory results for this task. Networks and natural language processing were proven useful for extracting the context of detected events producing word clouds of keywords describing those events. The NED methodology was also proven effective for processing large collections of real world documents and extracting valuable insights about the COVID-19 pandemic for a duration of six months. Finally, social network analysis on evolving networks can reveal implicit communities of users from online communities such as Stack Overflow and provide valuable insights of how they

[1]https://stackexchange.com/sites

relate with each other and what characteristics they have in common.

Bibliography

[1] J. Abraham, D. Higdon, J. Nelson, and J. Ibarra. Cryptocurrency price prediction using tweet volumes and sentiment analysis. *SMU Data Science Review*, 1(3):1, 2018.

[2] C. Aggarwal and K. Subbian. Evolutionary network analysis: A survey. *ACM Computing Surveys (CSUR)*, 47(1):1–36, 2014.

[3] M. Ahmad, S. Aftab, I. Ali, and N. Hameed. Hybrid tools and techniques for sentiment analysis: A review. *Int. J. Multidiscip. Sci. Eng*, 8(3):29–33, 2017.

[4] L. M. Aiello, G. Petkos, C. Martin, D. Corney, S. Papadopoulos, R. Skraba, A. Göker, I. Kompatsiaris, and A. Jaimes. Sensing trending topics in twitter. *IEEE Transactions on Multimedia*, 15(6):1268–1282, 2013.

[5] M. A. Al-Garadi, K. D. Varathan, S. D. Ravana, E. Ahmed, G. Mujtaba, M. U. S. Khan, and S. U. Khan. Analysis of online social network connections for identification of influential users: Survey and open research issues. *ACM Computing Surveys (CSUR)*, 51(1):1–37, 2018.

[6] R. Albert and A.-L. Barabási. Topology of evolving networks: local events and universality. *Physical review letters*, 85(24):5234, 2000.

[7] J. Alejandro. Journalism in the age of social media. *Reuters Institute Fellowship Paper*, 5(1-47):1, 2010.

[8] R. Alghamdi and K. Alfalqi. A survey of topic modeling in text mining. *Int. J. Adv. Comput. Sci. Appl.(IJACSA)*, 6(1), 2015.

[9] M. Allahyari, S. Pouriyeh, M. Assefi, S. Safaei, E. D. Trippe, J. B. Gutierrez, and K. Kochut. Text summarization techniques: a brief survey. *arXiv preprint arXiv:1707.02268*, 2017.

[10] J. Allan, J. G. Carbonell, G. Doddington, J. Yamron, and Y. Yang. Topic detection and tracking pilot study final report. ., 1998.

[11] J. Allan, R. Papka, and V. Lavrenko. On-line new event detection and tracking. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 37–45, 1998.

[12] H. Alvari, A. Hajibagheri, and G. Sukthankar. Community detection in dynamic social networks: A game-theoretic approach. In *2014 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2014)*, pages 101–107. IEEE, 2014.

[13] T. Aynaud, E. Fleury, J.-L. Guillaume, and Q. Wang. Communities in evolving networks: definitions, detection, and analysis techniques. In *Dynamics On and Of Complex Networks, Volume 2*, pages 159–200. Springer, 2013.

[14] T. Aynaud and J.-L. Guillaume. Static community detection algorithms for evolving networks. In *8th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks*, pages 513–519, 2010.

[15] T. Aynaud and J.-L. Guillaume. Multi-step community detection and hierarchical time segmentation in evolving networks. In *Proceedings of the 5th SNA-KDD workshop*, volume 11, 2011.

[16] A.-L. Barabâsi, H. Jeong, Z. Néda, E. Ravasz, A. Schubert, and T. Vicsek. Evolution of the social network of scientific collaborations. *Physica A: Statistical mechanics and its applications*, 311(3-4):590–614, 2002.

[17] F. Beck, M. Burch, S. Diehl, and D. Weiskopf. A taxonomy and survey of dynamic graph visualization. In *Computer Graphics Forum*, volume 36, pages 133–159. Wiley Online Library, 2017.

[18] S. Beyer and M. Pinzger. Grouping android tag synonyms on stack overflow. In *Proceedings of the 13th International Conference on Mining Software Repositories*, pages 430–440, 2016.

[19] S. Bird, E. Klein, and E. Loper. *Natural language processing with Python: analyzing text with the natural language toolkit.* " O'Reilly Media, Inc.", 2009.

[20] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022, 2003.

[21] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment*, 2008(10):P10008, 2008.

[22] S. P. Borgatti, A. Mehra, D. J. Brass, and G. Labianca. Network analysis in the social sciences. *science*, 323(5916):892–895, 2009.

[23] C. Borgs, M. Brautbar, J. Chayes, and B. Lucier. Maximizing social influence in nearly optimal time. In *Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms*, pages 946–957. SIAM, 2014.

[24] A. Bóta, M. Krész, and A. Pluhár. Dynamic communities and their detection. *Acta Cybernetica*, 20(1):35–52, 2011.

[25] R. Bourqui, F. Gilbert, P. Simonetto, F. Zaidi, U. Sharan, and F. Jourdan. Detecting structural changes and command hierarchies in dynamic social networks. In *2009 International conference on advances in social network analysis and mining*, pages 83–88. IEEE, 2009.

[26] F. Bu, A. E. Aiello, J. Xu, and A. Volfovsky. Likelihood-based inference for partially observed epidemics on dynamic networks. *Journal of the American Statistical Association*, 117(537):510–526, 2022.

[27] F. Calefato, F. Lanubile, M. C. Marasciulo, and N. Novielli. Mining successful answers in stack overflow. In *2015 IEEE/ACM 12th Working Conference on Mining Software Repositories*, pages 430–433. IEEE, 2015.

[28] T. Camber Warren. The geometry of security: Modeling interstate alliances as evolving networks. *Journal of Peace Research*, 47(6):697–709, 2010.

[29] J. Carbonell and J. Goldstein. The use of mmr, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 335–336, 1998.

[30] R. Cazabet and F. Amblard. Simulate to detect: a multi-agent system for community detection. In *2011 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology*, volume 2, pages 402–408. IEEE, 2011.

[31] R. Cazabet, G. Rossetti, and F. Amblard. *Dynamic community detection.* 2017.

[32] T. Chakraborty, A. Dalmia, A. Mukherjee, and N. Ganguly. Metrics for community analysis: A survey. *ACM Computing Surveys (CSUR)*, 50(4):1–37, 2017.

[33] Z. Chen, K. A. Wilson, Y. Jin, W. Hendrix, and N. F. Samatova. Detecting and tracking community dynamics in evolutionary networks. In *2010 IEEE International Conference on Data Mining Workshops*, pages 318–327. IEEE, 2010.

[34] G. G. Chowdhury. Natural language processing. *Annual review of information science and technology*, 37(1):51–89, 2003.

[35] P. Chunaev. Community detection in node-attributed social networks: a survey. *Computer Science Review*, 37:100286, 2020.

[36] A. Clauset, M. E. Newman, and C. Moore. Finding community structure in very large networks. *Physical review E*, 70(6):066111, 2004.

[37] H. Crane and W. Dempsey. Community detection for interaction networks. *arXiv preprint arXiv:1509.09254*, 2015.

[38] N. Dakiche, F. B.-S. Tayeb, Y. Slimani, and K. Benatchba. Tracking community evolution in social networks: A survey. *Information Processing & Management*, 56(3):1084–1102, 2019.

[39] V. Dankers, E. Bruni, and D. Hupkes. The paradox of the compositionality of natural language: a neural machine translation case study. *arXiv preprint arXiv:2108.05885*, 2021.

[40] A. Darwish and K. I. Lakhtaria. The impact of the new web 2.0 technologies in communication, development, and revolutions of societies. *Journal of advances in information technology*, 2(4):204–216, 2011.

[41] L. B. De Souza, E. C. Campos, and M. d. A. Maia. Ranking crowd knowledge to assist software development. In *Proceedings of the 22nd International Conference on Program Comprehension*, pages 72–82, 2014.

[42] Z. Dhouioui and J. Akaichi. Tracking dynamic community evolution in social networks. In *2014 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2014)*, pages 764–770. IEEE, 2014.

[43] R. Dridan and S. Oepen. Tokenization: Returning to a long solved problem—a survey, contrastive experiment, recommendations, and toolkit—. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 378–382, 2012.

[44] D. Duan, Y. Li, Y. Jin, and Z. Lu. Community mining on dynamic weighted directed graphs. In *Proceedings of the 1st ACM international workshop on Complex networks meet information & knowledge management*, pages 11–18, 2009.

[45] D. Duan, Y. Li, R. Li, and Z. Lu. Incremental k-clique clustering in dynamic social networks. *Artificial Intelligence Review*, 38(2):129–147, 2012.

[46] W. S. El-Kassas, C. R. Salama, A. A. Rafea, and H. K. Mohamed. Automatic text summarization: A comprehensive survey. *Expert Systems with Applications*, 165:113679, 2021.

[47] A. El-Kishky, Y. Song, C. Wang, C. Voss, and J. Han. Scalable topical phrase mining from text corpora. *arXiv preprint arXiv:1406.6312*, 2014.

[48] T. Falkowski, J. Bartelheimer, and M. Spiliopoulou. Mining and visualizing the evolution of subgroups in social networks. In *2006 IEEE/WIC/ACM International Conference on Web Intelligence (WI 2006 Main Conference Proceedings)(WI'06)*, pages 52–58. IEEE, 2006.

[49] T. Falkowski and M. Spiliopoulou. Data mining for community dynamics. *Künstliche Intell.*, 21(3):23–29, 2007.

[50] M. A. Fattah and F. Ren. Ga, mr, ffnn, pnn and gmm based models for automatic text summarization. *Computer Speech & Language*, 23(1):126–144, 2009.

[51] J. R. Finkel, T. Grenager, and C. D. Manning. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 363–370, 2005.

[52] F. Fogelman-Soulié, D. Perrotta, J. Piskorski, and S. Ralf. Evolving networks. *Mining Massive Data Sets for Security: Advances in Data Mining, Search, Social Networks and Text. Mining, and Their Applications to Security*, page 19, 2008.

[53] F. Folino and C. Pizzuti. Multiobjective evolutionary community detection for dynamic networks. In *Proceedings of the 12th annual conference on Genetic and evolutionary computation*, pages 535–536, 2010.

[54] S. Fortunato. Community detection in graphs. *Physics Reports*, 486(3-5):75–174, Feb. 2010.

[55] L. Freeman. The development of social network analysis. *A Study in the Sociology of Science*, 1(687):159–167, 2004.

[56] L. C. Freeman. Centrality in social networks conceptual clarification. *Social networks*, 1(3):215–239, 1978.

[57] X. Fu, S. Yu, and A. R. Benson. Modelling and analysis of tagging networks in stack exchange communities. *Journal of Complex Networks*, 8(5):cnz045, 2020.

[58] G. P. C. Fung, J. X. Yu, P. S. Yu, and H. Lu. Parameter free bursty events detection in text streams. In *Proceedings of the 31st international conference on Very large data bases*, pages 181–192. Citeseer, 2005.

[59] M. Gambhir and V. Gupta. Recent automatic text summarization techniques: a survey. *Artificial Intelligence Review*, 47(1):1–66, 2017.

[60] L. Gauvin, A. Panisson, and C. Cattuto. Detecting the community structure and activity patterns of temporal networks: a non-negative tensor factorization approach. *PloS one*, 9(1):e86028, 2014.

[61] R. George, K. Shujaee, M. Kerwat, Z. Felfli, D. Gelenbe, and K. Ukuwu. A comparative evaluation of community detection algorithms in social networks. *Procedia Computer Science*, 171:1157–1165, 2020.

[62] A. Ghasemian, P. Zhang, A. Clauset, C. Moore, and L. Peel. Detectability thresholds and optimal algorithms for community structure in dynamic networks. *Physical Review X*, 6(3):031005, 2016.

[63] K. Gimpel, N. Schneider, B. O'Connor, D. Das, D. Mills, J. Eisenstein, M. Heilman, D. Yogatama, J. Flanigan, and N. A. Smith. Part-of-speech tagging for twitter: Annotation, features, and experiments. Technical report, Carnegie-Mellon Univ Pittsburgh Pa School of Computer Science, 2010.

[64] B. Goethals. Frequent set mining. In *Data mining and knowledge discovery handbook*, pages 377–397. Springer, 2005.

[65] M. Goldberg, M. Magdon-Ismail, S. Nambirajan, and J. Thompson. Tracking and predicting evolution of social communities. In *2011 IEEE Third International Conference on Privacy, Security, Risk and Trust and 2011 IEEE Third International Conference on Social Computing*, pages 780–783. IEEE, 2011.

[66] R. Görke, P. Maillard, A. Schumm, C. Staudt, and D. Wagner. Dynamic graph clustering combining modularity and smoothness. *Journal of Experimental Algorithmics (JEA)*, 18:1–1, 2013.

[67] R. Görke, P. Maillard, C. Staudt, and D. Wagner. Modularity-driven clustering of dynamic graphs. In *International symposium on experimental algorithms*, pages 436–448. Springer, 2010.

[68] B. Hajian and T. White. Modelling influence in a social network: Metrics and evaluation. In *2011 IEEE Third International Conference on Privacy, Security, Risk and Trust and 2011 IEEE Third International Conference on Social Computing*, pages 497–500. IEEE, 2011.

[69] A. Halavais, K. H. Kwon, S. Havener, and J. Striker. Badges of friendship: Social influence and badge acquisition on stack overflow. In *2014 47th Hawaii international conference on System Sciences*, pages 1607–1615. IEEE, 2014.

[70] T. Hartmann, A. Kappes, and D. Wagner. Clustering evolving networks. In *Algorithm engineering*, pages 280–329. Springer, 2016.

[71] Q. He, K. Chang, and E.-P. Lim. Analyzing feature trajectories for event detection. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 207–214, 2007.

[72] T. Herlau, M. Mørup, and M. Schmidt. Modeling temporal evolution and multiscale structure in networks. In *International Conference on Machine Learning*, pages 960–968. PMLR, 2013.

[73] A.-S. Himmel, H. Molter, R. Niedermeier, and M. Sorge. Enumerating maximal cliques in temporal graphs. In *2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 337–344. IEEE, 2016.

[74] J. Hopcroft, O. Khan, B. Kulis, and B. Selman. Tracking evolving communities in large linked networks. *Proceedings of the National Academy of Sciences*, 101(suppl 1):5249–5253, 2004.

[75] Y. Hu, S. Wang, Y. Ren, and K.-K. R. Choo. User influence analysis for github developer social networks. *Expert Systems with Applications*, 108:108–118, 2018.

[76] C. Hutto and E. Gilbert. Vader: A parsimonious rule-based model for sentiment analysis of social media text. In *Proceedings of the International AAAI Conference on Web and Social Media*, volume 8, 2014.

[77] N. İlhan and Ş. G. Öğüdücü. Predicting community evolution based on time series modeling. In *2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 1509–1516. IEEE, 2015.

[78] A. S. Imran, S. M. Daudpota, Z. Kastrati, and R. Batra. Cross-cultural polarity and emotion detection using sentiment analysis and deep learning on covid-19 related tweets. *Ieee Access*, 8:181074–181090, 2020.

[79] P. Jaccard. The distribution of the flora in the alpine zone. 1. *New phytologist*, 11(2):37–50, 1912.

[80] M. Jacomy, T. Venturini, S. Heymann, and M. Bastian. ForceAtlas2, a Continuous Graph Layout Algorithm for Handy Network Visualization Designed for the Gephi Software. *PLoS ONE*, 9(6):e98679, June 2014.

[81] G. Jeh and J. Widom. Simrank: a measure of structural-context similarity. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 538–543, 2002.

[82] A. G. Jivani et al. A comparative study of stemming algorithms. *Int. J. Comp. Tech. Appl*, 2(6):1930–1938, 2011.

[83] V. Kharde, P. Sonawane, et al. Sentiment analysis of twitter data: a survey of techniques. *arXiv preprint arXiv:1601.06971*, 2016.

[84] J. Kleinberg. Bursty and hierarchical structure in streams. *Data mining and knowledge discovery*, 7(4):373–397, 2003.

[85] D. Knoke and S. Yang. *Social network analysis*. Sage Publications, 2019.

[86] G. Lample, M. Ballesteros, S. Subramanian, K. Kawakami, and C. Dyer. Neural architectures for named entity recognition. *arXiv preprint arXiv:1603.01360*, 2016.

[87] R. Lamsal. Design and analysis of a large-scale covid-19 tweets dataset. *Applied Intelligence*, 51(5):2790–2804, 2021.

[88] A. Lancichinetti and S. Fortunato. Community detection algorithms: a comparative analysis. *Physical review E*, 80(5):056117, 2009.

[89] A. Lancichinetti and S. Fortunato. Community detection algorithms: A comparative analysis. *Physical Review E*, 80(5):056117, Nov. 2009.

[90] A. Lancichinetti, S. Fortunato, and F. Radicchi. Benchmark graphs for testing community detection algorithms. *Physical review E*, 78(4):046110, 2008.

[91] P. Lee, L. V. Lakshmanan, and E. E. Milios. Incremental cluster evolution tracking from highly dynamic network data. In *2014 IEEE 30th International Conference on Data Engineering*, pages 3–14. IEEE, 2014.

[92] K. Lehnertz, G. Ansmann, S. Bialonski, H. Dickten, C. Geier, and S. Porz. Evolving networks in the human epileptic brain. *Physica D: Nonlinear Phenomena*, 267:7–15, 2014.

[93] E. Li, X. Li, and Z. Liu. Relationships and evolving networks of rural manufacturing clusters: a case study in yucheng county, henan province of china. *Chinese Geographical Science*, 21(3):364–376, 2011.

[94] H. Li and K. Yamanishi. Topic analysis using a finite mixture model. *Information processing & management*, 39(4):521–541, 2003.

[95] E. D. Liddy. Natural language processing. 2001.

[96] J. Littman, L. Wrubel, and D. Kerchner. United states presidential election tweet ids. *Harvard Dataverse*, 2016.

[97] Q. Liu, E. Agichtein, G. Dror, E. Gabrilovich, Y. Maarek, D. Pelleg, and I. Szpektor. Predicting web searcher satisfaction with existing community-based answers. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, pages 415–424, 2011.

[98] S. Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137, 1982.

[99] A. Lopez. Statistical machine translation. *ACM Computing Surveys (CSUR)*, 40(3):1–49, 2008.

[100] D. N. Makau, I. A. Paploski, C. A. Corzo, and K. VanderWaal. Dynamic network connectivity influences the spread of a sub-lineage of porcine reproductive and respiratory syndrome virus. *Transboundary and Emerging Diseases*, 69(2):524–537, 2022.

[101] C. Manning and H. Schutze. *Foundations of statistical natural language processing*. MIT press, 1999.

[102] C. D. Manning, M. Surdeanu, J. Bauer, J. R. Finkel, S. Bethard, and D. McClosky. The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*, pages 55–60, 2014.

[103] N. Masuda and R. Lambiotte. *A guide to temporal networks*. World Scientific, 2016.

[104] C. Matias and V. Miele. Statistical clustering of temporal networks through a dynamic stochastic block model. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 79(4):1119–1141, 2017.

[105] C. Matias, T. Rebafka, and F. Villers. A semiparametric extension of the stochastic block model for longitudinal networks. *Biometrika*, 105(3):665–680, 2018.

[106] W. Medhat, A. Hassan, and H. Korashy. Sentiment analysis algorithms and applications: A survey. *Ain Shams engineering journal*, 5(4):1093–1113, 2014.

[107] S. Melvin, W. Yu, P. Ju, S. Young, and W. Wang. Event detection and summarization using phrase network. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 89–101. Springer, 2017.

[108] M. Mori, T. Miura, and I. Shioya. Topic detection and tracking for news web pages. In *2006 IEEE/WIC/ACM International Conference on Web Intelligence (WI 2006 Main Conference Proceedings)(WI'06)*, pages 338–342. IEEE, 2006.

[109] M. Morini, P. Flandrin, E. Fleury, T. Venturini, and P. Jensen. Revealing evolutions in dynamical networks. *arXiv preprint arXiv:1707.02114*, 2017.

[110] P. Morrison and E. Murphy-Hill. Is programming knowledge related to age? an exploration of stack overflow. In *2013 10th Working Conference on Mining Software Repositories (MSR)*, pages 69–72. IEEE, 2013.

[111] E. Moustroufas, I. Stamelos, and L. Angelis. Competency profiling for software engineers: literature review and a new model. In *Proceedings of the 19th Panhellenic Conference on Informatics*, pages 235–240, Athens Greece, Oct. 2015. ACM.

[112] I. Moutidis and H. T. Williams. Utilizing complex networks for event detection in heterogeneous high-volume news streams. In *International Conference on Complex Networks and Their Applications*, pages 659–672. Springer, 2019.

[113] I. Moutidis and H. T. Williams. Good and bad events: combining network-based event detection with sentiment analysis. *Social Network Analysis and Mining*, 10(1):1–12, 2020.

[114] I. Moutidis and H. T. Williams. Community evolution on stack overflow. *Plos one*, 16(6):e0253010, 2021.

[115] P. J. Mucha, T. Richardson, K. Macon, M. A. Porter, and J.-P. Onnela. Community structure in time-dependent, multiscale, and multiplex networks. *science*, 328(5980):876–878, 2010.

[116] F. Murtagh. A survey of recent advances in hierarchical clustering algorithms. *The computer journal*, 26(4):354–359, 1983.

[117] D. Nadeau and S. Sekine. A survey of named entity recognition and classification. *Lingvisticae Investigationes*, 30(1):3–26, 2007.

[118] M. Newman. *Networks*. Oxford university press, 2018.

[119] M. E. Newman and M. Girvan. Finding and evaluating community structure in networks. *Physical review E*, 69(2):026113, 2004.

[120] R. Njoroge. *Impacts of social media among the youth on behavior change: a case study of University students in selected universities in Nairobi, Kenya*. PhD thesis, University Of Nairobi, 2013.

[121] B. O'Connor, M. Krieger, and D. Ahn. Tweetmotif: Exploratory search and topic summarization for twitter. In *Fourth International AAAI Conference on Weblogs and Social Media*, 2010.

[122] N. Ohsaka, T. Akiba, Y. Yoshida, and K.-i. Kawarabayashi. Dynamic influence analysis in evolving networks. *Proceedings of the VLDB Endowment*, 9(12):1077–1088, 2016.

[123] Y. Ohsawa, N. E. Benson, and M. Yachida. Keygraph: Automatic indexing by co-occurrence graph based on building construction metaphor. In *Proceedings IEEE International Forum on Research and Technology Advances in Digital Libraries-ADL'98-*, pages 12–18. IEEE, 1998.

[124] J. Olive, C. Christianson, and J. McCary. *Handbook of natural language processing and machine translation: DARPA global autonomous language exploitation*. Springer Science & Business Media, 2011.

[125] G. K. Orman and V. Labatut. A comparison of community detection algorithms on artificial networks. In *International conference on discovery science*, pages 242–256. Springer, 2009.

[126] G. Palla, A.-L. Barabási, and T. Vicsek. Quantifying social group evolution. *Nature*, 446(7136):664–667, 2007.

[127] M. Papoutsoglou, G. M. Kapitsaki, and L. Angelis. Modeling the effect of the badges gamification mechanism on personality traits of stack overflow users. *Simulation Modelling Practice and Theory*, 105:102157, 2020.

[128] H.-S. Park and C.-H. Jun. A simple and fast algorithm for k-medoids clustering. *Expert systems with applications*, 36(2):3336–3341, 2009.

[129] L. Peel and A. Clauset. Detecting change points in the large-scale structure of evolving networks. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[130] S. Petrović, M. Osborne, and V. Lavrenko. Streaming first story detection with application to twitter. In *Human language technologies: The 2010 annual conference of the north american chapter of the association for computational linguistics*, pages 181–189, 2010.

[131] P. Pons and M. Latapy. Computing communities in large networks using random walks. In *International symposium on computer and information sciences*, pages 284–293. Springer, 2005.

[132] A.-M. Popescu, M. Pennacchiotti, and D. Paranjpe. Extracting events and event descriptions from twitter. In *Proceedings of the 20th international conference companion on World wide web*, pages 105–106, 2011.

[133] R. Prabowo, M. Thelwall, I. Hellsten, and A. Scharnhorst. Evolving debates in online communication: A graph analytical approach. *Internet Research*, 2008.

[134] D. R. Radev, W. Fan, and Z. Zhang. Webinessence: A personalized web-based multi-document summarization and recommendation system. In *NAACL workshop on automatic summarization*. Citeseer, 2001.

[135] C. Ragkhitwetsagul, J. Krinke, M. Paixao, G. Bianco, and R. Oliveto. Toxic code snippets on stack overflow. *IEEE Transactions on Software Engineering*, 2019.

[136] G. Ranco, D. Aleksovski, G. Caldarelli, M. Grčar, and I. Mozetič. The effects of twitter sentiment on stock price returns. *PloS one*, 10(9):e0138441, 2015.

[137] L. Richardson. Beautiful soup documentation. retrieved 1st april, 2016, 2007.

[138] A. Ritter, S. Clark, O. Etzioni, et al. Named entity recognition in tweets: an experimental study. In *Proceedings of the 2011 conference on empirical methods in natural language processing*, pages 1524–1534, 2011.

[139] C. Rosen and E. Shihab. What are mobile developers asking about? a large scale study using stack overflow. *Empirical Software Engineering*, 21(3):1192–1223, 2016.

[140] G. Rossetti and R. Cazabet. Community discovery in dynamic networks: a survey. *ACM Computing Surveys (CSUR)*, 51(2):1–37, 2018.

[141] G. Rossetti, L. Pappalardo, D. Pedreschi, and F. Giannotti. Tiles: an online algorithm for community discovery in dynamic social networks. *Machine Learning*, 106(8):1213–1241, 2017.

[142] M. Rosvall and C. T. Bergstrom. Maps of random walks on complex networks reveal community structure. *Proceedings of the national academy of sciences*, 105(4):1118–1123, 2008.

[143] M. Rosvall and C. T. Bergstrom. Mapping change in large networks. *PloS one*, 5(1):e8694, 2010.

[144] H. Safdari, M. Contisciani, and C. De Bacco. Reciprocity, community detection, and link prediction in dynamic networks. *Journal of Physics: Complexity*, 3(1):015010, 2022.

[145] G. Salton and M. J. McGill. *Introduction to modern information retrieval.* mcgraw-hill, 1983.

[146] A. Sanfeliu and K.-S. Fu. A distance measure between attributed relational graphs for pattern recognition. *IEEE transactions on systems, man, and cybernetics*, (3):353–362, 1983.

[147] K. Sarkar. Syntactic trimming of extracted sentences for improving extractive multi-document summarization. *Journal of Computing*, 2(7):177–184, 2010.

[148] H. Sayyadi, M. Hurst, and A. Maykov. Event detection and tracking in social streams. In *Third international AAAI conference on weblogs and social media*, 2009.

[149] D. Schenk and M. Lungu. Geo-locating the knowledge transfer in stackoverflow. In *Proceedings of the 2013 international workshop on social software engineering*, pages 21–24, 2013.

[150] J. Scott. Social network analysis. *Sociology*, 22(1):109–127, 1988.

[151] J. Shang, L. Liu, F. Xie, Z. Chen, J. Miao, X. Fang, and C. Wu. A real-time detecting algorithm for tracking community structure of dynamic networks. *arXiv preprint arXiv:1407.2683*, 2014.

[152] G. Silvestri, J. Yang, A. Bozzon, and A. Tagarelli. Linking accounts across social networks: the case of stackoverflow, github and twitter. In *KDWeb*, pages 41–52, 2015.

[153] N. Sobin. *Syntactic analysis: The basics.* John Wiley & Sons, 2010.

[154] M. Spiliopoulou. Evolution in social networks: A survey. In *Social network data analytics*, pages 149–175. Springer, 2011.

[155] D. Sportiche, H. Koopman, and E. Stabler. *An introduction to syntactic analysis and theory.* John Wiley & Sons, 2013.

[156] S. Stieglitz and L. Dang-Xuan. Political communication and influence through microblogging–an empirical analysis of sentiment in twitter messages and retweet behavior. In *2012 45th Hawaii international conference on system sciences*, pages 3500–3509. IEEE, 2012.

[157] C. Sutton and A. McCallum. An introduction to conditional random fields for relational learning. *Introduction to statistical relational learning*, 2:93–128, 2006.

[158] S. Tabassum, F. S. Pereira, S. Fernandes, and J. Gama. Social network analysis: An overview. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 8(5):e1256, 2018.

[159] L. Tang, H. Liu, J. Zhang, and Z. Nazeri. Community evolution in dynamic multi-mode networks. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 677–685, 2008.

[160] W. H. Thompson, P. Brantefors, and P. Fransson. From static to temporal network theory: Applications to functional brain connectivity. *Network Neuroscience*, 1(2):69–99, 2017.

[161] Y. Tian, W. Ng, J. Cao, and S. McIntosh. Geek talents: Who are the top experts on github and stack overflow? *CMC-COMPUTERS MATERIALS & CONTINUA*, 61(2):465–479, 2019.

[162] A. Tumasjan, T. Sprenger, P. Sandner, and I. Welpe. Predicting elections with twitter: What 140 characters reveal about political sentiment. In *Proceedings of the International AAAI Conference on Web and Social Media*, volume 4, pages 178–185, 2010.

[163] B. Vasilescu, V. Filkov, and A. Serebrenik. Stackoverflow and github: Associations between software development and crowdsourced knowledge. In *2013 International Conference on Social Computing*, pages 188–195. IEEE, 2013.

[164] A. Vespignani. Modelling dynamical processes in complex socio-technical systems. *Nature Physics*, 8(1):32–39, Jan. 2012.

[165] T. Viard, M. Latapy, and C. Magnien. Computing maximal cliques in link streams. *Theoretical Computer Science*, 609:245–252, 2016.

[166] S. Vijayarani, R. Janani, et al. Text mining: open source tokenization tools-an analysis. *Advanced Computational Intelligence: An International Journal (ACII)*, 3(1):37–47, 2016.

[167] A. Voutilainen. *Part-of-speech tagging*, volume 219. The Oxford handbook of computational linguistics, 2003.

[168] X. Wang, H. Liu, and W. Fan. Connecting users with similar interests via tag network inference. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, pages 1019–1024, 2011.

[169] Y. Wang, B. Wu, and X. Pei. Commtracker: A core-based algorithm of tracking community evolution. In *International Conference on Advanced Data Mining and Applications*, pages 229–240. Springer, 2008.

[170] C. Wartena and R. Brussee. Topic detection by clustering keywords. In *2008 19th international workshop on database and expert systems applications*, pages 54–58. IEEE, 2008.

[171] S. Wasserman, K. Faust, et al. Social network analysis: Methods and applications. 1994.

[172] C. L. Wayne. Topic detection and tracking (tdt). In *Workshop held at the University of Maryland on*, volume 27, page 28. Citeseer, 1997.

[173] J. Weng and B.-S. Lee. Event detection in twitter. In *Proceedings of the International AAAI Conference on Web and Social Media*, volume 5, 2011.

[174] P. Willett. The porter stemming algorithm: then and now. *Program*, 2006.

[175] F. Wu, W.-L. Zhao, Q. Ji, and D. Zhang. Dependency, centrality and dynamic networks for international commodity futures prices. *International Review of Economics & Finance*, 67:118–132, 2020.

[176] J. Xie, S. Kelley, and B. K. Szymanski. Overlapping community detection in networks: The state-of-the-art and comparative study. *Acm computing surveys (csur)*, 45(4):1–35, 2013.

[177] K. S. Xu and A. O. Hero. Dynamic stochastic blockmodels for time-evolving social networks. *IEEE Journal of Selected Topics in Signal Processing*, 8(4):552–562, 2014.

[178] N. Xu and W. Mao. Multisentinet: A deep semantic network for multimodal sentiment analysis. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 2399–2402, 2017.

[179] S. Xu, A. Bennett, D. Hoogeveen, J. H. Lau, and T. Baldwin. Preferred answer selection in stack overflow: Better text representations... and metadata, metadata, metadata. In *Proceedings of the 2018 EMNLP Workshop W-NUT: The 4th Workshop on Noisy User-generated Text*, pages 137–147, 2018.

[180] X. Xu, N. Yuruk, Z. Feng, and T. A. Schweiger. Scan: a structural clustering algorithm for networks. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 824–833, 2007.

[181] T. Yang, Y. Chi, S. Zhu, Y. Gong, and R. Jin. A bayesian approach toward finding communities and their evolutions in dynamic social networks. In *Proceedings of the 2009 SIAM International Conference on Data Mining*, pages 990–1001. SIAM, 2009.

[182] Y. Yang, T. Pierce, and J. Carbonell. A study of retrospective and on-line event detection. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 28–36, 1998.

[183] Z. Yang, R. Algesheimer, and C. J. Tessone. A Comparative Analysis of Community Detection Algorithms on Artificial Networks. *Scientific Reports*, 6(1):30750, Aug. 2016.

[184] L. Yujian and L. Bo. A normalized levenshtein distance metric. *IEEE transactions on pattern analysis and machine intelligence*, 29(6):1091–1095, 2007.

[185] D. M. Zajic, B. J. Dorr, and J. Lin. Single-document and multi-document summarization techniques for email threads using sentence compression. *Information Processing & Management*, 44(4):1600–1610, 2008.

[186] A. Zakrzewska and D. A. Bader. A dynamic algorithm for local community detection in graphs. In *2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 559–564. IEEE, 2015.

[187] J. Zhang, C. Zong, et al. Deep neural networks in machine translation: An overview. *IEEE Intell. Syst.*, 30(5):16–25, 2015.

[188] W. E. Zhang, Q. Z. Sheng, J. H. Lau, and E. Abebe. Detecting duplicate posts in programming qa communities via latent semantics and association rules. In *Proceedings of the 26th International Conference on World Wide Web*, pages 1221–1229, 2017.

Appendix A

Manual annotation for the 24/06/2016 articles.

Article Annotation	**Number of Articles**
Brexit referendum	79
US Presidential Campaign	36
Orlando mass shooting	15
Freddie Gray trial	5
Stonewall Uprising monument	4
West Virginia floods	4
Mike Flynn death	4
California wildfires	3
Barack Obama executive amnesty	3
stock markets	3
global stock markets	2
Independence Day Resurgence	2
latest news	2
Ralph Stanley death	2
Rio Olympic games laboratory	2
transgender military ban	2
North Korea nuclear threat	2
Viernheim cinema shootings	2
Muslim migration	2
EgyptAir plane crash	2
Abigail Fisher University admission trial	2
VW emissions scandal	2
delegate lawsuit Virginia	2

Bernie Worrell death	2
climate change	1
trash	1
Harlan Zimmerman stock buybcks	1
Moto Guo acne	1
Tony Robbins Unleash the Power Within	1
Dolly Parton New York City	1
CAIR Orlando shooting	1
Trotify bike clap	1
growing up kids	1
NYPD dogs	1
Led Zeppelin lawsuit	1
Marie Claire of Mexico vs Ivanka Trump	1
man escaped using fake leg	1
Todd Solondz modern American life	1
Brooklynite Facebook inbox	1
China harvesting organs from prisoners	1
civilizational jihad	1
Chauncy's Chance GoFundMe	1
Swiss Army Man movie	1
Gabi Grecko Penthouse	1
toddler graduation fight	1
Frank Zappa Eat That question	1
Radio City Music Hall	1
Civil War story	1
phony Auschwitz man	1
banks stress test	1
FIFA president raise	1
Jo Cox assassination	1
Boeing refurbishes Hornets	1
e time girls	1
Fenton Allen attacks judge on trial	1
Game of thrones	1

Aziz Ashari attacks Trump	1
Mark Leonard book	1
EU politics	1
date news	1
fashion news	1
Xavier Helgesen solar energy	1
Kyle Moses commodore relieved of command	1
FBI auction	1
Malaysia Airlines potentially wreckage	1
Derricl Rose Knicks	1
Matthew McConaughey new movie	1
art and depression	1
newsletter	1
Congo vice president sentenced to prison	1
Omar Mateen life	1
George Lucas museum	1
Texas University admission trial	1
quitting drugs	1
Russia Promobot	1
Mashrou Leila band	1
Democrats sat down in the house	1
Lee Nelson Nazi golfballs	1
diet advice	1
Banks secure log in	1
Buenos Aires zoo closing	1
Eli Roth Death Wish	1
Mike Francesa Zack Wheeler	1
Tony Robbins hot coals	1
Gang leader gets in prison for life	1
US Democrats	1
Facebook slew of dead dogs	1
TSA PreCheck	1
microaggresion to women	1

gay marriage	1
man fell on tracks 77th Street	1
backhand return tennis	1
Fodor’s travel guides being sold	1
Rodrigo Duterte Philippines president	1
Jona Rechnitz plane ride	1
targeted individuals	1
marijuana policy	1
YouTube Red	1
US immigration	1
smartphone blindness	1
Zika virus	1
Andrea Dworkin feminism	1
ESPN Johnny Manziel spiral	1
Kelly McGillis gun permit	1
Burger King Mac n’ Cheetos	1
Uber price estimation	1
Louis Vuitton Kimm Jones	1
Boston Celtics NBA draft	1
CNN Lewandowski	1
Invisibilia podcast	1
Chinese Dietary Guidelines	1
New York gay pride	1
John Ashe death	1
Aziz Ansari against Trump	1
Michael Herr death	1
globalization	1
Green Martha Stweart	1
Tarvaris Jackson pulled a gun on his wife	1
Abigail Fischer University admission trial	1
John Kennedy letter	1
Chine mobile patent wars	1
Rocketship John Danner	1

MusclePharm Arnold Schwarzenegger	1
child in the car app	1
Jonathan Rauch why politics doesn't work anymore	1
David Cameron popularity	1
organ transplants	1
Broadway scandals	1
Trump trip to Scotland	1
Shark Week	1
Aaron Persky judge recall	1
Trump marriage contract	1
Democratic party	1
Collaborative robots	1
Canada Express Entry	1
Google symptom cards	1
Paul Bucha Medal of Honor	1
Johnny Manziel domestic violence case	1
Federal Reserve announcement	1
Dior designer	1

Table 5.1: Annotations for the articles posted on the 24/06/2016 from the All The News data set.