Imperial College London

Department of Computing

Mining the Social Web: Community Detection in Twitter, and its Application in Sentiment Analysis

by

Nasser Zalmout

Submitted in partial fulfilment of the requirements for the MSc Degree in Computing(Computational Management Science) of Imperial College London

September 2013

To my loving family. If it wasn't for you I wouldn't be here, writing these words.

Acknowledgment

I would like to present my deepest gratitude for my supervisor, Dr. Moustafa Ghanem. His guidance, wisdom, and patience throughout the project's work were essential for the completion of this thesis. The insights he provided me with, not only have helped me enriching the content of this thesis, but enhanced my understanding of the entire world of research in general.

I would like to also thank my family members for their continuous support. If it wasn't for them, I wouldn't be writing these words now. And finally I would like to present my gratitude to my friend Wasfi, his constant motivation kept me going, even at the toughest times.

Abstract

The current outbreak and widespread of microblogging and social networking websites are reshaping many aspects of the modern day social interaction. Social Media today has exceeded the limits of entertainment or simple social interaction contexts. Social media now is better described as a living organism, that has a structure and a soul. The Social Media now reflects the pulse of the people; it reacts to their emotions, and interacts with their opinions. Analyzing and monitoring the content of Social Media can bring about some valuable insights, of which the conventional media means weren't able to convey.

This project tackles the concept of community detection within Social Media means, Twitter in particular. The virtual communities that the people on Social Media tend to group themselves into can provide precious sources of information regarding the patterns of communication and knowledge propagation. The project identifies the dimensions of similarity and interaction between any pair of users and provides the tools to calculate them. These dimensions are used as building blocks to construct the network structure to be used to detect the communities.

The project presents a number of algorithms and software systems implementations that approach the issue of community detection in Twitter. The parameters and factors affecting the community detection process are then investigated in depth, to help tuning the procedure to provide the most efficient results.

The problem of sentiment analysis is then studied in the context of the detected communities. The project implements a procedure in which the detected communities are utilised to enhance the overall performance of the sentiment analysis process. The sociological principles of *Influence and Homophily* will provide the theoretical background to support the analysis. These principles revolve upon the fact that people who communicate more than others, tend to share somewhat similar opinions.

Finally, the project approaches the issues of *Dynamicity* and *Evolution* in Social Media, by providing a classifier that can be used to classify an unknown user or set of Tweets to one of the available detected communities. The characteristics of the detected communities are used as features for the training process. These classified communities can then be used for the sentiment analysis as explained.

All of the introduced approaches and methods are backed by comprehensive experiments and evaluation procedures.

”You cannot live for yourselves; a thousand fibres connect you with your fellow men, and along those fibres, as along sympathetic threads, run your actions as causes, and return to you as effects.”

Henry Melvill

Contents

1 Introduction and Motivation **11**
1.1 Internet and Social Media: The Digital Communication Era . . . 11
1.1.1 Social Media in Numbers . . . 11
1.1.2 Detecting Virtual Communities . . . 11
1.2 Sentiment Analysis . . . 12
1.3 Objectives and Contributions . . . 12
1.4 Thesis Report Structure and Scope . . . 13

2 Literature Review **15**
2.1 Social Network Analysis (SNA) . . . 15
2.1.1 Social Network . . . 15
2.1.2 Structural Holes and Opinion Leaders Detection . . . 17
2.1.3 Dimensions of Connection . . . 17
2.1.4 Recent Challenges Facing SNA . . . 17
2.2 *Homophily* and *Influence* . . . 18
2.3 Community Detection . . . 18
2.3.1 Clustering . . . 19
2.3.2 Graph Cuts and Graph Partitioning . . . 20
2.3.3 Network-Centric Community Detection Algorithms . . . 20
2.3.4 Hierarchy-Centric Community Detection Algorithms . . . 21
2.3.5 Node-Centric Community Detection Algorithms . . . 22
2.3.6 Group-Centric Community Detection Algorithms . . . 22
2.4 Textual Mining . . . 23
2.4.1 Clustering and Classification . . . 23
2.4.2 Reasons for text mining . . . 23
2.4.3 Text mining procedures . . . 24
2.5 Sentiment Analysis . . . 24
2.5.1 Polarity Classification . . . 24
2.5.2 Subjectivity Detection . . . 25
2.5.3 Features/Aspects extraction . . . 25
2.5.4 General Approaches . . . 26
2.5.5 Supervised and Unsupervised approaches . . . 26
2.5.6 Classification Based on Relationship Information . . . 27
2.5.7 Challenges: . . . 27
2.5.8 Sentiment Analysis for Social Media Content . . . 28
2.6 Discussion . . . 28

3 Tools and Software Packages 29
3.1 Twitter Data Crawling Tools . . . 29
3.2 Data Storage Tools . . . 30
3.3 Network Analysis Tools . . . 31
3.4 Network Storage Tools . . . 32
3.4.1 Scientific Library for Python (SciPy) . . . 33
3.5 Natural Language Toolkit (NLTK) . . . 33
3.6 Sentiment Analysis: Sentistrength . . . 34

4 Community Detection, Methodology 35
4.1 Overview . . . 35
4.1.1 Community Definition . . . 35
4.1.2 Multidimensional Heterogeneity . . . 35
4.1.3 Community Detection Dataflow . . . 36
4.2 User Profiling . . . 37
4.3 Building the Network . . . 38
4.3.1 Building the Similarity Matrix . . . 39
4.3.2 Building the Interaction Matrix . . . 41
4.3.3 Aggregation . . . 43
4.4 Community Detection . . . 43
4.4.1 Input . . . 43
4.4.2 Processing: Community Detection Algorithms . . . 43
4.4.3 Output . . . 44
4.4.4 *Dendrogram* Graph . . . 44
4.5 Performance Issues . . . 45
4.6 Summary . . . 46

5 Community Detection, Analysis and Evaluation 48
5.1 Overview . . . 48
5.2 Case study A: Members of the Parliament (MPs) Dataset, Outbound Only . . . 49
5.2.1 Dataset Collection . . . 49
5.2.2 Dataset Statistics . . . 49
5.3 Evaluation Metrics . . . 50
5.3.1 Evaluation by Set-Matching: $F - score$ and $Purity$ . . . 50
5.3.2 Evaluation by Set-Matching, based on Counting-Pairs Approach: $B - Cubed$ . . . 52
5.4 Supervised Evaluation Approach . . . 52
5.4.1 Overview . . . 52
5.4.2 Experimental Setup . . . 53
5.4.3 Methodology . . . 53
5.4.4 Experiment 1: Analysing the Different Metrics and Parameters . . . 55
5.4.5 Experiment 2: Aggregating Similar Clusters . . . 58
5.4.6 Experiment 3: The Effect of the Different Dimensions Over Performance . 60
5.4.7 Experiment 4: Removing the Most-Common-Entities . . . 65
5.5 Unsupervised Evaluation Approach . . . 67
5.5.1 Overview . . . 67
5.5.2 Experimental Setup . . . 67
5.5.3 Methodology . . . 68

5.5.4 Observations and Analysis . . . 68
5.6 Summary . . . 74

6 Incorporating the Communities into Sentiment Analysis 77
6.1 Overview . . . 77
6.2 Sentiment Analysis: Supervised and Unsupervised Learning . . . 78
6.3 Sentiment Analysis: Proposed Methodology . . . 78
6.4 Bayesian Inference Model . . . 79
6.5 Case Study B: Members of the Parliament (MPs) Dataset, Inbound and Outbound 82
6.5.1 Objectives . . . 82
6.5.2 Dataset Collection and Statistics . . . 82
6.5.3 Community Detection . . . 84
6.5.4 Sentiment Analysis . . . 86
6.6 Building a General Community Classifier . . . 87
6.6.1 Objective . . . 88
6.6.2 Experimental Setup . . . 88
6.6.3 Experiment 1: The Effect of the Training Dataset Size . . . 89
6.6.4 Experiment 2: Changing the Temporal Distance Between the Testing and Training Datasets . . . 90
6.6.5 Experiment 3: Incorporating the Classified Communities into Sentiment Analysis . . . 92
6.7 Summary . . . 93
6.7.1 Sentiment Analysis Results . . . 93
6.7.2 Communities' Classifier . . . 94

7 Conclusion and Future Work 95
7.1 Project Discussion . . . 95
7.2 Contributions . . . 96
7.3 Achievements . . . 96
7.4 Future Work . . . 97
7.4.1 Using Overlapping Community Detection Algorithms . . . 98
7.4.2 Better Approaches Regarding Evolution and Dynamicity . . . 98
7.4.3 Semantic Analysis of the Detected Communities . . . 98

A *Case Study A Results* 99

List of Figures

2.1 A sample social network, showing closure and brokerage nodes. Source [21] 16
2.2 A sample network, showing possible Graph cuts. Source [17] 20

3.1 The system's SQL database schema 30
3.2 Network Analysis Tools comparison, source [3] 31

4.1 The dataflow of the community detection process 36
4.2 SQL database schema for the elements used at the user profiling process 37
4.3 The resulting Hashtags' communities of the example dataset Table 4.1. *Dendrogram* diagram (on the left), and the corresponding clustering diagram (on the right) 45
4.4 The resulting Retweets' communities of the example dataset Table 4.1. *Dendrogram* diagram (on the left), and the corresponding clustering diagram (on the right) 45
4.5 The overall community detection steps, example 47

5.1 A sample *Dendrogram* graph (left), and the resulting clustering graph (right). For the Hashtags dimension, case study A 54
5.2 The Purity metric charts associated with the edge-weight and clusters parameters of the Hashtags dimension. Case study A 56
5.3 The F-Score metric charts associated with the edge-weight and clusters parameters of the Hashtags dimension. Case study A 57
5.4 The B-Cubed metric charts associated with the edge-weight and clusters parameters of the Hashtags dimension. Case study A 58
5.5 The B-Cubed metric results charts associated with the Hashtags dimension, after aggregating similar clusters. Case study A 59
5.6 The B-Cubed metric results for the detected communities of the Links dimension, case study A 61
5.7 The B-Cubed results for the detected communities of the aggregation of Links and Hashtags dimensions. Case study A 61
5.8 The B-Cubed results for the detected communities of the Mentions similarity dimension. Case study A 62
5.9 The B-Cubed results for the detected communities of the Retweets interaction dimension. Case study A 62
5.10 The B-Cubed results for the detected communities of the aggregated interaction dimensions. Case study A 63
5.11 The B-Cubed results for the detected communities of the Replies interaction dimension. Case study A 63

5.12 The B-Cubed results for the detected communities of the aggregated interaction dimensions, without the Replies interaction dimension. Case study A 64
5.13 The B-Cubed results for the detected communities of aggregating all dimensions excluding the Replies dimension. Case study A 64
5.14 The B-Cubed results for the detected communities of the Hashtags similarity dimension, before removing the most common entities. Case study A 66
5.15 The B-Cubed results for the detected communities of the Hashtags similarity dimension, after removing the most common entities. Case study A 67
5.16 The effect of the edge-weight threshold parameter value over the number of users and Hashtags, *Fastgreedy* algorithm. Case study A 70
5.17 The edge-weight threshold parameter's metrics results for the detected communities of the Hashtags similarity dimension. Case study A 71
5.18 Users distribution plot over the detected communities, when adjusting the clusters count parameter. Hashtags dimension, *Fastgreedy* algorithm. Case study A 73
5.19 The clusters count parameter's metrics results for the detected communities of the Hashtags similarity dimension. Case study A 74

6.1 Diagram showing the application of the Naive Bayesian Classifier for sentiment analysis 82
6.2 Users' distribution plot over the detected communities, corresponding to edge-weight threshold value change. Case study B 85
6.3 Sentiment analysis results. Results of Sentistrength (blue) vs results after incorporating the communities' information through the Bayesian Inference Model (red). Case study B 87
6.4 Diagram showing the application of the Bayesian classifier to classify communities. Case study B 88
6.5 The effect of changing the training dataset size over the communities classifier performance. Case study B 90
6.6 The effect of changing the time span period between the testing and training datasets over the communities classifier performance. Time period 1 being the closest to the testing dataset. Case study B 91
6.7 Sentiment analysis results. Results of Sentistrength (blue) vs results after incorporating the classified communities' information through the Bayesian Inference Model (red). Case study B 93

A.1 Hashtags dimension results 99
A.2 Mentions similarity dimension results 100
A.3 Links dimension results 100
A.4 Mentions interaction dimension results 101
A.5 Replies dimension results 101
A.6 Retweets dimension results 102
A.7 Aggregated interaction dimensions dimension results 102
A.8 Aggregated interaction excluding Replies dimensions dimension results 103
A.9 Links+Hashtags aggregated dimension results 103
A.10 All aggregated dimension results 104

List of Tables

3.1 A sample similarity matrix, from section (4.3.1) 32
3.2 A sample similarity matrix represented as a sparse matrix 33

4.1 User profiling example, showing the Hashtags and Retweets affiliated with each user 38
4.2 The resulting Hashtags similarity matrix affiliated with the user profiling Table 4.1 39
4.3 The resulting Retweets interaction matrix affiliated with the user profiling table 4.1 42
4.4 Dataset properties affecting the community detection performance 46

5.1 Statistics related to the MPs Outbound Only dataset 50
5.2 The most common Hashtags removed from the Hashtags similarity dimension. Case study A 66
5.3 Hashtags clusters count input, and actual count output. *Fastgreedy* algorithm. Case study A 72

6.1 The likelihood probabilities of the Bayesian Model example, calculated from the dataset and community information. 80
6.2 Results of applying the Bayesian Inference formula, sentiment analysis example . 81
6.3 Statistics related to the MPs inbound and outbound dataset. Case study B . . . 83
6.4 Statistics related to the MPs Inbound and Outbound dataset, after removing the one-time-activity users. Case study B 83
6.5 Most common Hashtags and Mentions removed from the dataset. Case study B . 84
6.6 Actual clusters count for each input clusters count; setting the clusters count parameter. Case study B 84
6.7 The Likelihood probabilities for David Cameron dataset. Case study B 86

Chapter 1

Introduction and Motivation

This chapter is dedicated to outline the scope and overview of the project, the affiliated research area, and the project's goals.

1.1 Internet and Social Media: The Digital Communication Era

Internet and Social Media have become an important pillar of the modern day life, used on daily basis to transfer vast amount of raw data and knowledge. Social media has also transformed the traditional one-way media outlets into cooperative and multi-directional sources of information, where the user has become the news maker, rather than just the recipient. Such tools make the perfect venue for generating, shaping and exchanging individuals' opinions and sentiments. Studying and understanding this amount of raw and unstructured data should be such a valuable source for important insights and structured knowledge.

1.1.1 Social Media in Numbers

Based on some recent statistics, the number of internet users worldwide is about 2.4 billion users, or 35% of the entire population of the planet. This number jumped of about 361 millions in 2000, or a growth rate of about 566%[9]. For Social Media, the number of Facebook users for example reached about 1.15 billion users, or about 16% of the planet's population[14]. For Twitter, the number is about 500 million users[24]. Such staggering number indicates how reliant people have become to the world of internet and Social Media.

With the sheer magnitude of content generated and communicated on a daily basis, it is important to provide feasible and efficient techniques to transform this unstructured content, into usable and structured knowledge.

1.1.2 Detecting Virtual Communities

Humans are social creatures by nature, we always try to group ourselves instinctively within communities and societies. We also tend to group ourselves with people with somewhat similar ideologies and backgrounds, as this would facilitate easier understanding and communication. The communication patterns also tend to intensify for members in the same group, in comparison to members from the outside. All of these facts are the essence of the sociological principles of *Influence and Homophily;* where people eventually tend to develop similar views and opinions over different issues, based on the characteristics of these groups.

The Social Media, or the virtual social world in general, is a reflection of the real world. People in the virtual world still instinctively group themselves with people sharing similar backgrounds, forming a kind of virtual communities. Detecting these virtual groupings might be of special importance for the analysis and monitoring of Social Media, as it provides a disciplined view of the network in which people connect to each other.

Research in the field of community detection for Social Network Analysis is not new, however, applying the community detection procedures for Social Media (rather than the conventional Social Networks), is of relatively young age. This topic is one of the most important, and most addressed issues in this thesis. This thesis will also try to tackle some new issues

1.2 Sentiment Analysis

Sentiment Analysis is the automatic procedure in which the overall polarity of the textual content is measured. Or simply, it provides an estimation of how ”positive” or ”negative” a piece of text is. Sentiment analysis has many applications at different fields. Within the context of Social Media, sentiment analysis has important applications at the fields of brand monitoring, consumer products reviews, campaigning and financial markets.

Building up on the community detection arguments illustrated before, virtual communities are based upon the principle of sharing opinions and sentiments within same groups. In this capacity, a logical approach would be to try to join the the worlds of sentiment analysis and community detection, as the members of the same communities are supposed to share somewhat similar sentiments.

In this realm of sentiment analysis for the virtual communities, the relationship between the individual sentiments of the users and the sentiment of the containing communities is bidirectional. Finding the sentiment of the users will lead to the overall sentiment of the containing community, and knowing the overall sentiment of the community leads to an important prior information for the sentiment of new members joining this community.

This project will try to investigate this logic, and evaluate the accuracy of this approach.

1.3 Objectives and Contributions

This thesis details a project that examined and generated procedures to detect and analyse communities in Social Media content, Twitter in particular. The project investigates the different algorithms, parameters and factors that affect the overall performance and efficiency of detecting communities within a Twitter based datasets, and evaluates each and every one of them.

The project continues with an application to the detected communities by incorporating the communities' information into the sentiment analysis process. The sentiment analysis process tested here uses an unsupervised learning approach. The aim of this part would be to evaluate the accuracy enhancement that incorporating the structural information of the communities might bring about. Finally, the project presents a classifier that can be used to classify a user or a set of Tweets in the available data's space into a certain community, based on a supervised learning approach from the already available communities.

Despite the fact that implementing and applying these procedures and approaches entailed many challenges, the project's goals can be boiled down to these main objectives:

- Surveying the affiliated background materials and related work and researches to the project's fields, making sure to evaluate any of the used techniques and choosing the best performing ones, whenever possible.
- Investigating and implementing tools and procedures for detecting the communities within a Twitter content. In addition to studying the execution performance of each of these procedures, making any possible alterations to the implementation to enhance the overall performance.
- Studying and analysing the different parameters, algorithms, and factors that contribute to the community detection procedures, evaluating each of these factors individually or in the aggregate.
- Applying the community detection procedure at the sentiment analysis process, and evaluating the results.
- Building a general community classifier based on supervised learning, using the characteristics of the detected communities of the available dataset. In addition to studying the parameters affecting its performance, and evaluating its results.

All of these objectives have been achieved. These objectives will be referenced throughout the thesis to illustrate what objective each chapter or section is addressing. The section below explains the general layout of the thesis, and maps the chapters to the objectives.

The project also presents novel work, as of this writing, regarding the following contributions:

- Deep and novel analysis of the factors that contribute to the community detection problem in Twitter. With focus on community detection procedures based on Link similarity rather than content similarity.
- Utilising the sociological principles of *Homophily* and *Influence* for the sentiment analysis process, through the integration of the the community detection procedures with the sentiment analysis process for Twitter based content.
- Utilising the characteristics of the detected communities to build a classifier for classifying unknown users or Tweets within the current data space, into one of the detected communities. Approaching the issue of rapid evolution in Social Media.

1.4 Thesis Report Structure and Scope

This thesis report is divided into 6 different chapters:

- Chapter 1 provides an overview of the project and the research area, describing the motivation and objectives.

- Chapter 2 provides a review of the background literature affiliated with the project's research area. It highlights the important concepts that support the theory and implementation of the project's procedures. This chapter is concerned with fulfilling the first objective of the project.
- Chapter 3 describes the methodology and implementation details used for the community detection procedures, and the connectivity dimensions of the users. It describes the overall approach and data flow, provides pseudocodes for some of the implemented algorithms, and illustrates an example that outline the entire process. Chapter 3 is affiliated with the second objective of this project.
- Chapter 4 discusses the parameters and factors affecting the community detection procedure, and provides an evaluation for the different factors that affect the overall performance. This chapter is broken down to many experiments that explain and analyse the system's performance. This chapter is affiliated with the third objective.
- Chapter 5 addresses the issue of incorporating the detected communities to the sentiment analysis process, using a Bayesian Inference Model. The chapter then discusses the evaluation of this process. The chapter also addresses the community classifier's implementation, and how to obtain a general community classifier based on the available communities data. Chapter 5 fulfills the fourth and fifth objectives of this report.

Chapter 2

Literature Review

2.1 Social Network Analysis (SNA)

The Social Network Analysis field is not new, however, this field has been revived recently by the current outbreak of Social Media and Social Media tools.

Modeling communities of people (nodes) and interactions (connections) in a structured manner is of great importance for understanding how information gets propagated and spread in a system of people. The resulting structure builds up on two basic facts:

1. People tend to group into clusters as a result of communication opportunities for which people tend to meet, both physically and virtually.
2. Communication is more influential and frequent within these clusters, such that people within the same cluster tend to develop similar views. [21]

The main purpose for structure-based social network analysis is the detection of communities within a system of people, and to obtain the main opinion makers or opinion leaders within these communities. The resulting structure of the system can be viewed as a graph. Many graph-based techniques and algorithms can then be used to process the resulting network and extract subgroups or communities. These methods include cliques-based, degree, and matrix-perturbation methods.

2.1.1 Social Network

The Social Network: is a network structure composed of social entities (might be people, organisations or groups) as nodes. And connections in the form of edges, that represent the connection or relationship amongst these entities. Such connections might be friendships, acquaintanceship, or business connection etc.

Structural Holes in Social Networks

People, or nodes, within a single cluster tend to share similar views and knowledge. The process of making explicit data or knowledge into insider or local knowledge within the group is passed through certain channels, or relations, between two nodes of different clusters. This process of knowledge passing produces structural holes in the structure of the social network. People (nodes) in groups separated by structural holes focus more on their local knowledge and

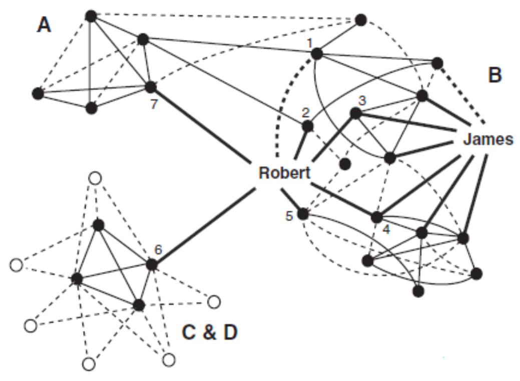


Figure 2.1: A sample social network, showing closure and brokerage nodes. Source [21]

interaction rather than the general network correspondence. One way to look at these holes is like a buffer, or like an insulator in an electric circuit isolating the different components of the network. It is also worth Mentioning here that holes represent non-redundant sources of information, amongst the different clusters. [21]

Nodes inside the network can be classified as either:

1. Closure: nodes within the cluster or group. Closure nodes are important to strengthen the connections, by constantly doing the same thing or having similar data, they get better at they do or know. Closure nodes also incur more trust within the networks, where reputation tends to propagate more sensitively for nodes within the clique, as the connections tend to transfer reputation status more efficiently.

2. Brokerage: nodes connecting two different clusters. Brokerage nodes are important to diversify the knowledge in the cluster by bringing in new data from other clusters. [21]

In the figure above it can be observed that James at clique B represents a closure node, a local node that strengthens the connections within the clique. Whereas Robert is a part of clique B, but at the same time he has connections with both cliques A , C and D. Robert here represents the Brokerage node at the network. Robert has a vital access to less redundant knowledge, and new knowledge and activities going on at other cliques.

Opinion Leaders

Robert in the Figure 2.1 above also has the capability of transferring new knowledge and data from other cliques to the local one. The structural holes within the network graph present Robert with the opportunity of brokering interactions and communication amongst several cliques, while displaying different beliefs to other contacts. Robert in the graph above is what researchers have identified as opinion leader.

Brokerage nodes, or opinion leaders, present a valuable opportunity of action amongst different cliques representing different communities. Brokerage nodes build bridges amongst the communities, bridges between markets and organisations for example. [21]

Rule of Agency and Personality

It's worth Mentioning here that the structure of the network and the presence of Brokerage nodes is not the main factor for the efficiency of the structure, after all it's the people that act, not networks. Taking advantage of the structural holes is mainly dependent upon the skills or the performance of the Brokerage node at transferring and decoding of potential useful knowledge.

Another point in this area is regarding the dynamics of the network structure. Modern technologies facilitate easy and continuous change in the connections and general structure of the network. However, researches have indicated that Opinion Leaders or brokerage nodes tend in previous structures tend to maintain their rule at consequent changes, and that structural holes tend not to change much, which provides some sense of stability and equilibrium for the observed structure over time. [21]

2.1.2 Structural Holes and Opinion Leaders Detection

The following parameters are used to identify and evaluate the potential Opinion Leaders. These parameters can be considered as metrics when it comes to the community detection and opinion leaders identification.

1. Network density: network density can be defined by the strength of the connections amongst the network. It is calculated by dividing the number of the available connections in a sample network, by the number of all possible connections.
2. Hierarchy: which is the degree at which the contacts are connected to a single contact within the network
3. Non-redundant contacts: The count of the unique contacts, without the contacts redundant with other contacts in the network. Which primarily refers to the count of different cliques or clusters that the contact is connected with.
4. Betweenness index: A measure of the structural holes that the agent or the contact has access to within the network.

2.1.3 Dimensions of Connection

The communication and interaction patterns within the context of social networks in general, and Social Media in particular, is of a multidimensional heterogeneous form[27]. The different nodes, or users, tend to use different forms of communication and knowledge sharing. A good example would be the Twitter content, where the interaction dimensions could be in the shape of Retweeting, Replying, Mentioning or Hashtags similarity.

2.1.4 Recent Challenges Facing SNA

Making the shift of traditional Social Network Analysis, to the Social Media Analysis incurs many challenges that didn't exist before, or weren't as challenging. Such issues include:

- Scalability: Social Network Analysis at the age of Social Media has to deal with millions of different nodes and even much more edges. This explosive outbreak of nodes and connections requires capable tools to carry out the required analysis.

- Heterogeneity: The communication and interaction patterns within the context of Social Media in general is of a multidimensional heterogeneous form. The different nodes, or users, tend to use different forms of communication and knowledge sharing

- Evolution: The Social Media network evolves constantly, which requires a dynamic and adaptive analysis techniques and modeling with snapshots of the general system.

- Evaluation: The main issue here is the lack of ground truth, or complete information due to many reasons, like privacy.

2.2 *Homophily* and *Influence*

Homophily can be defined as the tendency of people to associate and bond with other people who share similarities with them. This concept is directly correlated with the well know saying of "birds of a feather flock together". A more abstract definition has been provided in the literature as the "love of the same".

Homophily forms a very fundamental principle in social networks in general. Miller McPherson, et al, have explained the presence of the *Homophily* within social networks, and explained how *Homophily* has powerful effects and implications for the information people receive, the attitudes they form, and the interactions they experience. The Social Media's recent outbreak provides a fertile scene for the *Homophily* as a principle affecting the people's choices of other people to connect to.

The explanation above illustrates how *Homophily* affects the people's connections and interactions within a social network. *Influence Models,* on the other hand, study the information diffusion patterns and the spread of new ideas within the network as a result of the people's connections. *Influence* can be considered as a core principle affecting the knowledge propagation and new ideas generation. Some sample influence modeling approaches include:

- Linear Threshold Model (LTM).

- Independent Cascade Model (ICM).

2.3 Community Detection

The virtual communities within Social Media contents are the direct effect of the *Homophily* principle, and the effect these communities have over their members is the result of the *Influence* principle.

As we have illustrated before, contacts or nodes within a Social Media mean can be represented with graphs. Communities are then identified from the resulting graphs via utilising conventional graphs parsing algorithms. One way towards community detection is via identifying which nodes are more densely connected in comparison to the rest of the nodes.

In this context, the community structure can be defined as essentially a form of grouping or clustering of the nodes. The resulting clusters are characterised by having dense connections amongst the nodes within the community, and sparse connections for the inter-clusters nodes. These connections might have different semantics, as shall be illustrated at the next section, but they all contribute to the communities' structure.

Community types:

1. Explicit: based on conscious human decisions.
2. Implicit: stemming out of the interactions and communication amongst the users, considerably more difficult to observe and analyse.

Community attributes:

1. Overlapping: Networks not containing clear cuts in terms of possible cliques or communities, having a number of common nodes.
2. Weighted Participation: Nodes having different participation metrics' measures amongst the network or clique.
3. Roles: Where nodes can be assigned to roles based on their location/importance within the network.
4. Hierarchy: Hierarchical structure can be observed at the clique or network. [17]

2.3.1 Clustering

Clustering is the process of automatically aggregating or grouping similar items and records together, based upon sharing similar attributes' values. The clustering analysis process is usually aiming at exploring new insights at the given content, as the clustering process does not require the user to know the different clusters beforehand. Actually clustering sometimes aims at exploring the attributes and parameters of the clusters that the content might be representing.

There are a number of different algorithms for clustering. However, most of these algorithms are based at iteratively assigning items to clusters, calculating some measure, then reassigning items to clusters until no much of change is observed at the measure, i.e until the process converges. This measure can be based upon statistical variability, spatial distance, and others.

Some of the most used clustering algorithms include:

- K-means, an exclusive clustering algorithm. K predefined clusters, which might be a problem sometimes. In that sense, K-means can also be considered as a classification algorithm
- Fuzzy C-means, an overlapping clustering method.
- Hierarchical clustering.
- Mixture of Gaussians, a probabilistic clustering approach.

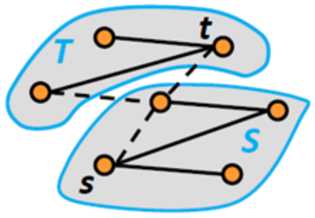


Figure 2.2: A sample network, showing possible Graph cuts. Source [17]

2.3.2 Graph Cuts and Graph Partitioning

The community in the graph theory context can be defined as a graph cut. The graph cut in this context can be defined as given the nodes u and v in graph $G = (V, E)$, a cut is a set of edges $C \subset E$, such that the two nodes are not connected on the graph $G' = (V, E - C)$. [17]

The problem of community detection in the context of graph theory can be defined as a Graph Partitioning problem, which can be formulated as given a graph $G = (V, E)$, find a partition of V in K disjoint subsets, such that the number of edges in E of which the endpoints belong to different subsets is minimized.[17]

2.3.3 Network-Centric Community Detection Algorithms

The most important and most widely used approach. The network-based community detection algorithms study the overall topology of the network, aiming at obtaining possible partitions from within the network. These algorithms usually include some kind of a criterion or metric defined upon all the partitions, rather than a certain group. The following are some algorithm types that follow the network-centric approach.

Vertex Similarity

Vertex similarity community detection algorithm is based upon obtaining some kind of a similarity metric between any pair of nodes. This similarity is a broad principle, it can be defined in many contexts and could attain many different features. A commonly used similarity is the *structural similarity*; where the similarity is defined by how much any pair of nodes connect to similar nodes. Once this similarity measures are obtained for all nodes in the system, any clustering algorithm can be used to obtain the different groups.

A sample similarity measures include *Jaccard* and *Cosine* similarity measures:

$$Jaccard(v_i, v_j) = \frac{|N_i \cap N_j|}{|N_i \cup N_j|}$$

$$Cosine(v_i, v_j) = \frac{|N_i \cap N_j|}{\sqrt{|N_i| \cdot |N_j|}}$$

Where $N_i$represents the neighbours of vertex v_i.

Modularity Maximisation

Modularity can be defined as a metric, or measure, of the quality of the network partitions. Modularity as a metric, and modularity maximisation as a community detection algorithm are of special importance. For any given network, with m different edges, the expected edges count between any given pair of nodes would be:

$$edges(v_i, v_j) = \frac{d_i d_j}{2m}$$

Where, d_i is the degree of the vertex i, or the number of edges incidents of vertex i. The strength of a community would then be:

$$strength(C) = \sum_{i \in C, j \in C} Aij - \frac{d_i d_j}{2m}$$

Where Aij is the number of edges between vertices i and j.

Assuming that the network is partitioned into k different communities, the overall communities network quality can be calculated as:

$$\sum_{l=1}^{k} \sum_{i \in C_l, j \in C_l} Aij - \frac{d_i d_j}{2m}$$

Modularity can then be defined as:

$$Q = \frac{1}{2m} \sum_{l=1}^{k} \sum_{i \in C_l, j \in C_l} Aij - \frac{d_i d_j}{2m}$$

Where the term $\frac{1}{2m}$ is used to normalise the modularity value to between -1 and 1.

A number of community detection algorithms uses the principle of modularity maximisation to find the most optimal structure. The universe of potential representations can get quite big for bigger problems, so, approximate and heuristic approaches are usually applied. [17]

A possible approach here is through a greedy algorithm. We begin off with all nodes belonging to their own separate communities, We then merge those two communities that make a better overall modularity score. The process continues until a local modularity maximum is found. The community detection of *Fastgreedy* is a direct application for this approach, which will be used intensively throughout this thesis. Another important algorithm that utilises modularity is Girvan - Newman algorithm.

2.3.4 Hierarchy-Centric Community Detection Algorithms

Another approach that uses the topology of the network to obtain the communities, by building a hierarchical structure of all possible communities based on the available topology. The hierarchy-centric community detection entails two different approaches.

Divisive Hierarchical Clustering

A Top-Down Approach. This approach starts with the entire network structure, then divides the nodes into several disjoint sets. The partitioning continues until a certain threshold is reached, or a metric score achieved. A good example about this approach is the Girvan-Newman's *Edgebetweenness* Algorithm community detection algorithm:

1. Compute betweenness centrality measure for all of the available edges.
2. Remove the edge with highest value.
3. Recompute all the measures or metrics.
4. Exit if threshold value achieved.
5. Go back to step 2.

Where the betweenness centrality is defined as:

$$g(v) = \sum_{s \neq v \neq t} \frac{\sigma_{st}(v)}{\sigma_{st}}$$

Where σ_{st} is the number of shortest paths going from node s to node t, and $\sigma_{st}(v)$ is the number of those paths that pass through v.

The end result of the algorithm guarantees optimal separation amongst the available communities, as removing the edge with highest betweenness centrality would lead to clusters' separation, as discussed at the opinion leaders section. However, the algorithm was found to be quite inefficient in terms of speed and coding complexity.

Agglomerative Hierarchical Clustering

In contrast to the divisive hierarchical approach, the agglomerative hierarchical approach is down-top. It begins from a number of base communities, might even be 1 node per community, and starts merging them based on a certain criterion. Modularity can be used here as the merging criterion.

2.3.5 Node-Centric Community Detection Algorithms

The node-centric approaches require each node in a possible group to satisfy certain characteristics and properties. An example of such algorithms are the Complete Mutuality algorithms, in which the groups, or communities, take the form of *cliques. A clique* is a maximum complete subgraph in which all nodes are adjacent to each other. [Tang Lei]

2.3.6 Group-Centric Community Detection Algorithms

As in the node-centric approaches grouping is based upon shared properties, however, the group-centric approaches require the group as a whole to satisfy certain characteristics rather than the individual nodes within. In this context, low connectivity nodes within the group are acceptable, as long as they satisfy the characteristics.

2.4 Textual Mining

Text mining can be defined as the process at which text is transferred into data that can be analysed. This incurs the procedures of creating an index for the individual terms, based on the location of the term within the original text, or based on other techniques or protocols. The words and indexes can then be used for a variety of analysis methods.

The idea behind indexing is not to store the abstract word's location, the indexing process in text mining is rather about storing the word's meaning, or concept, within the given context. The concepts are then stored in database, concepts database, and used for further analysis and processing. The concepts are extracted from the given text via a series of computational linguistic techniques for extraction processes.

Data mining, in contrast to text mining, is the process of discovering patterns and trends within large datasets. Data mining is an interdisciplinary field, involving constructs from the fields of machine learning, artificial intelligence, database systems, and statistical methods. The two most essential operations in the data mining process are clustering and classification.

2.4.1 Clustering and Classification

Clustering is the process of automatically aggregating or grouping similar items and records together, based upon sharing similar attributes' values. Clustering usually assumes no prior information regarding the number of available clusters. Classification in its essence is similar to clustering in the process of segmenting the available items into separate classes. However, unlike clustering, it is required that the user/analyst knows beforehand how each of these classes are defined, and what are the different attributes. It is also required that each of the available items should have values for the segmenting attributes. Classification is considered less exploratory that clustering.

As in clustering, many different algorithms can be used for classification purposes. One of the most important approaches is via decision trees, where new items can be classified by traversing the tree to the leaf which represents a class. Much effort is needed to properly obtain the decision tree and evaluate it.

K-means can also be used as a classification approach, as Mentioned above. Other algorithms include Support Vector Machines (SVM), and many others.

2.4.2 Reasons for text mining

1. Enriching the content, by improving the indexing of the text.
2. Systematic Review of Literature, by systematically reviewing a large content. This process can help reduce time and effort needed in general literature review.
3. Discovering new insights: as I have Mentioned earlier text mining on its own does not bring about new insights or knowledge. However, data mining the resulting indexed content of the text mining process might actually obtain new knowledge.
4. Computational Linguistics Research, where text mining is considered as a tool for bigger linguistic researches.

2.4.3 Text mining procedures

1. Information retrieval (IR): The process of selecting the most appropriate or relevant content or documents to be mined, by matching the desired query to the possible content. If the content was structured, as in scientific publications of a certain field for example, the information retrieval process would be relatively easier. However, if there weren't a structured content to refer to, the process will have to include as much content as possible to be able to derive the required results, and will be computationally more consuming to map content to locations.

2. Natural language processing (NLP): In a simple notation, NLP is about transferring or interpreting the human readable text into data that the computer can read and process. It utilises linguistic and grammatical principles and tools for the interpretation procedures. Such tools include Part Of Speech (POS), parsing tools, dictionaries, and others.

3. Information extraction (IE): By structuring the resulting data of the NLP step. Terms and named-entities are identified here, in addition to the possible connections or relationships between the entities and terms. The results of this steps are considered facts, or assertions and are usually collected in a database for future analysis. The starting point of extraction process is usually a set of templates for the kinds of information to be obtained. A simple example from chemistry is a pattern where two chemicals, A and B are found close to the phrase "reacts with". [2]

4. Data mining (DM): identifying trends and patterns in the dataset, to find new knowledge insights.

A special kind of text mining, although uses some different approaches, is opinion mining and sentiment analysis. A more comprehensive review follows.

2.5 Sentiment Analysis

Sentiment Analysis is the automatic procedure in which the overall polarity of the content is measured. It involved natural Language Processing (NLP), computational linguistic, and textual analytics to obtain the subjective and sentimental information from a given text. The sentiment analysis process entails a number of tasks or procedure after which the overall polarity or the subjectivity of the content can be specified.

Throughout the next sections I will be outlining the different tasks associated with sentiment analysis, and some of the approaches and algorithms that current researchers are using.

2.5.1 Polarity Classification

Classification here at a simple notation is the process of classifying the opinions in an opinionated piece of text, under two opposite sentiment polarities. The opinion in the text is assumed to be covering a single topic, field or context. The process may be utilized for summarisation processes, or simply to extract the overall sentiment, say negative.

The opinionated text may be addressing subjective or objective issues, where the author may be addressing an issue by stating his/her own opinions, or stating a fact or a piece of news that might be considered as polarised (e.g: good news or bad news) within a specific context. But it's worth Mentioning here that the distinction between subjective and objective information can be subtle.[16]

A number of related problems to the polarity classification include the Related Categories, which explores the reasons on why the users had an overall positive or negative sentiment. Identifying the pros and cons in this manner helps at improving the overall polarity, in addition to enriching the helpfulness of individual reviews, where judgments that are reinforced by reason are more trustworthy. Other problems include Rating Inference , by measuring the "degree of positiveness", and having a multi-point scale for the overall output.

An important issue in this context is Agreement; in the opposite manner of polarity classification, lies the issue of determining whether two different texts should receive similar or different sentiment label.

2.5.2 Subjectivity Detection

As illustrated above, polarity classification operates under the assumption that the content of the text being opinionated. However, real life texts might be objective or might include both objective and subjective content that are related to the given context.

Subjectivity detection can be thought of as an independent process. However, it was found that having subjectivity detection as a proceeding process for polarity classification increases the efficiency of the sentiment analysis. Subjectivity Detection is a harder problem than polarity classification, so improvements at the subjectivity detection promises better results for the sentiment analysis process as a whole.[19]

When we discussed polarity classification we have made the assumption that the given text to be processed covers a single and certain topic. However, this might not be the general case. An enhancement to the subjectivity detection can be done by first operating topic-based text filtering, then progressing with the subjectivity detection system. This procedure is called Joint Topic Sentiment Analysis.

Another similar class of problems is determining view points and perspectives. This problem is characterised by a collection of bundled attitudes and beliefs within the given text, which means that simple polarity classification techniques might not be suitable here, and techniques based on extraction will have to be utilised.

2.5.3 Features/Aspects extraction

The feature extraction processes are one of the fundamental steps for data driven analysis for textual content. Converting a plain text into a features vector enables more comprehensive and rigorous procedures of data mining and sentiment analysis.

2.5.4 General Approaches

- Term Presence vs. Frequency: Term frequency has been an important factor for regular information retrieval and topic-based analysis procedures. However, for sentiment and opinion analysis, it has been shown that just the presence of the term has a big impact on the sentiment analysis and subjectivity detection procedures, regardless of its frequency. (Bo Pang, 2002)

- Term-based Features: Corresponding to the position of the term within the text, which might have a big influence on the overall sentiment. The location can be encoded within the feature vector to be utilised in the analysis process. Another factor is the n-gram based analysis, where having a unigram, bigram and trigram based analysis will affect the analysis process and results. The actual effect of each is debatable though. (Pang & Lee, 2008)

- Parts of Speech: Which refers to the morphological behaviour of the lexical item within the text. It is important for both topic-based and sentiment analysis. Adjectives are commonly utilised for both sentiment analysis and subjectivity detection. (Pang & Lee, 2008). There were also many studies on the utilisation of certain verbs and nouns for sentiment analysis. (Pang & Lee, 2008)

- Syntax: The syntax is quite important for analysing short texts.

- Negation: Where despite that a negation term might be present only once at a certain sentence, it transforms it into the total opposite. An interesting research was conducted by combining the parts-of-speech approach with negation, to detect whether the presence of a negation term does indeed reverse the polarity, or to obtain which parts of the sentence the negation is affecting. (J.-C. Na, 2004)

- Topic-Oriented Features: Where similar sentences might refer to totally different sentiments if addressing different topics. In addition to the fact that the classifiers trained for certain domain will give erroneous results if used for different domains.

2.5.5 Supervised and Unsupervised approaches

One way to classify the algorithms used for the sentiment analysis process is to divide them into those that utilise machine learning constructs, and those that use some form of lexical inference based on specific terms and applying certain lexical rules. However, many approaches utilise both techniques together.[28]

The two major machine learning classes for the learning process are supervised and unsupervised learning methods. Supervised learning relies on pre-existing set of labeled data, or lexicons in our case, to be used to infer the overall sentiment of the text. Whereas unsupervised learning assumes no previous knowledge or training of the text at hand. Supevised learning based sentiment analysis requires an already labeled dataset to train the classifier. This training dataset is fed to the classifier to train upon what features' values produce which classifications. Probabilistic and non-probabilistic supervised machine learning algorithms are used for this learning process, like the Naive Bayesian Classifier, and Support Vector Machines (SVM). The main mathematical rule behind the Naive Bayesian Classifier is:

Unsupervised learning utilises machine learning methods used for classifying and extracting the sentiment lexicons and their degree of polarity. The advantage of these approaches is that it can be utilised to crude content without or with little background information or evidence. It can also be used to bigger content in comparison to other supervised approaches.

- Sample method includes Unsupervised Lexicon Induction, by first obtaining sentiment lexicon from the text using unsupervised algorithm. Then the degree of subjectivity for that lexicon can be measured by the available indicators at the text. The polarity can be measured by propagation from seed terms whose polarities are already known. The unsupervised algorithm used to extract the sentiment lexicons could utilise linguistic heuristics, like opposite connecters (e.g: ”but”), to obtain separate polarities and then separate them. [11] Or a number of other techniques.

- Another approach is by bootstrapping, where an initial classifier is used to obtain labelled data, which is then used as an input for supervised learning based algorithm.

2.5.6 Classification Based on Relationship Information

Another way to enhance the analysis process is by utilising Classification Based on Relationship Information. The main idea is to take advantage of the relationship connections between different documents, different classes, different features and discourse participants within the content at hand. Going into the details won’t be beneficial here as it’s a quite lengthy discussion for the size of this report.

A possible way to make use of relationship information within the context of Social Media is to investigate the usage of Social Network Analysis constructs to facilitate or enhance the sentiment analysis. Which this project will try research in more detail.

2.5.7 Challenges:

- The scarcity of the possible classes, as in ”positive or negative” or the star rating system, in movies rating for example. And the interleaving nature of the context, which makes it hard to distinguish a certain distinct classification for a certain context.

- The fact that sentiments and opinions are usually expressed in a more subtle way than regular fact based text. This makes sentiment analysis a harder problem than regular facts based analysis in terms of the used algorithms and techniques. The problem gets intensified when observing isolated chunks of text or small sentences. This makes applying sentiment analysis techniques for Social Media content a quite challenging process, due to the fact that the content in Social Media means is usually short.

- Sentiment analysis if context sensitive and domain dependent. The same phrase or expression can mean different things at different contexts.

- Unlike in topic analysis for documents, where the topic is based upon what the majority of the content is focusing on, regardless of the order, opinion analysis is quite affected by the overall order of the ideas. e.g: “A is better than B” represents the exact opposite opinion from “B is better than A.” [16]

2.5.8 Sentiment Analysis for Social Media Content

In addition to the previously Mentioned challenges for documents-based sentiment analysis, the application of sentiment analysis and opinion mining techniques for Social Media content, like Twitter and Facebook, imposes a set of new challenges. These include:

- The relatively short content (140 characters for Twitters). This imposes a big restriction for the application of many document or blog based sentiment analysis techniques.
- The unstructured nature of the content, and the necessity to work with unstructured information extraction methods.
- The issue of spam and fake reviews, posts, and comments.
- Sarcasm and irony detection. A very difficult problem to be addressed rigorously. Sarcasm can shift the entire polarity of the
- Grammatical mistakes, broken sentence boundaries, spelling problems, context specific terminology, slang terminology, etc.
- Lack of contextual data.

2.6 Discussion

Modeling communities of people (nodes) and interactions (connections) in a structured manner is of great importance for understanding how information gets propagated and spread in a system of people. And as in the real world, people in the virtual world also tend to aggregate themselves within communities and clusters of close interests and backgrounds. Many different algorithms and approaches can be utilised to capture and detect these virtual communities of a Social Media content.

The importance of the community for the context of Social Media, is the fact that communication is more influential and frequent within these clusters, such that people within the same cluster tend to develop similar views and sentiment polarities. The theoretical principles of *Homophily* and *Influence* support this fundamental argument.

Given the illustrated community definition, and it's importance within a Social Media context, raises the questions of potential practical applications. This thesis will investigate the potential advantage gained of utilising the community information at the sentiment analysis process. Sentiment analysis process in Social Media is quite prone to different factors affecting the overall accuracy as outlined at the sentiment analysis section. Sentiment analysis is Social Media also lacks the contextual information that might be vital for proper analysis. Integrating the community information with the sentiment analysis process might be able to provide this contextual background, and help improving the overall accuracy.

Chapter 3

Tools and Software Packages

This chapter is dedicated to discuss the implementation issues related to the project. The chapter begins by providing an overview of the used tools and software packages, it discusses the reasons why each tool has been used and provides a comparison of potential tools when possible. The chapter then progresses to provide a description of the different stages the analysis goes through, and what are the responsibilities of each stag

3.1 Twitter Data Crawling Tools

The data collection process for the project was operated through Trendak. Trendak is a Social Media analytics and monitoring website based in Egypt[30].

Accessing the Twitter content was operated through a special API that Trendak has provided me with access to. The API is a Restful service that collects the datasets in JSON format. The service enables collection of all Tweets, inbound and outbound, that are affiliated with a specific Twitter account, or list of accounts, in a given time period.

The datasets for the case studies included in this project will revolve about the UK Members of the Parliament (MPs). A more detailed for each of the corresponding datasets will be provided each case study's overview. The dataset was collected by crawling the official Twitter accounts related with each MP. The crawling process was grouped by the political affiliation; that is, crawling the Twitter accounts of each political party's MPs separately. Trendak's collection process progressed as follows:

1. Lists for all UK MPs, grouped by political affiliation, were obtained from the website Tweetminster.co.uk. Tweetminster.co.uk is a website that collects and analyses media content related to the UK politics.

2. A database was built for these members' details, including each member's political affiliation.

3. The Twitter crawling is now operated for each member of each political party.

The API provided from Trendak had the functionality to set all of the following features:

- Political party, which can be Conservative, Labour, Libdem, or others.

- Start and end dates of the dataset.
- Direction, inbound, outbound or both. Outbound means the Tweets issued by the crawled user, whereas outbound means the Tweets that targets the crawled user, by Mentions, Replies etc.
- Key words, which allows for the collection of the Tweets that contain these words only.
- Topic of the dataset.

3.2 Data Storage Tools

Throughout the implementation process, many split and join operations have to be used to properly prepare the data for further processing. To be able to efficiently analyse and process the data, relational-based database has to be used to generate some of the correlations and relationships amongst the different data segments. A MySQL database was used for this purpose, and the corresponding Python connecters were installed to be able to communicate with the database. In addition to the MySQL Workbench software for managing the database and the database schema.

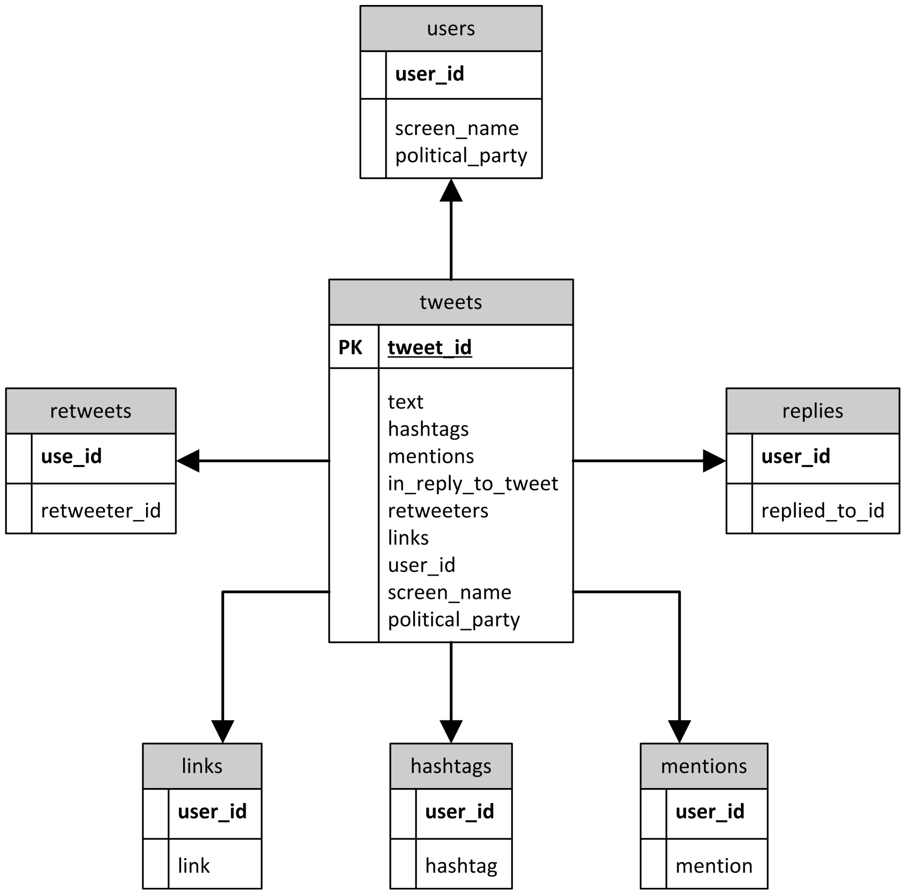


Figure 3.1: The system's SQL database schema

Figure 3.1 illustrates the SQL schema used to store the Tweets and other data items to be used in the analysis process.

3.3 Network Analysis Tools

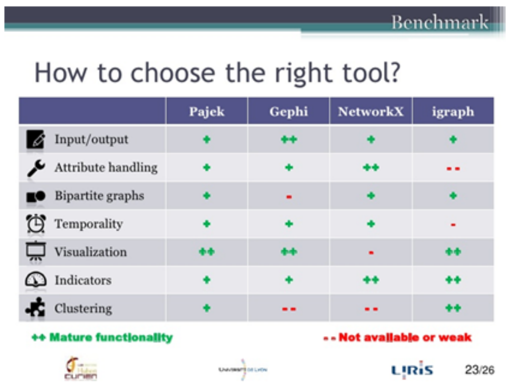
Social Network Analysis techniques have been under research for some time now, before the current outbreak of Social Media means of Twitter and others. The fact that Social Network Analysis is situated between several domains (sociology, computer science, mathematics and physics) has led to many different methodological approaches and to a lot of tools[3]. As illustrated before at the project's background paper, the structure of social network can be outlined by a variety of graph and matrix based approaches, which provided a number of different analysis tools. Tools generated so far cover different angles of networks processing, the tools below are amongst those of which I found to be most relevant:

- Pajek: a standalone windows-based platform for analysis and visualisation of large networks.

- igraph: an open source library for creating and manipulating graphs. igraph also implements some of the cutting edge algorithms for community detection and clustering.

- NetworkX: an open source library for creating, manipulating and studying the structure and dynamics of complex networks.

- Gephi: a standalone platform for visualising and exploring existing networks.



Benchmark

How to choose the right tool?

	Pajek	Gephi	NetworkX	igraph
Input/output	+	++	+	+
Attribute handling	+	+	++	--
Bipartite graphs	+	-	+	+
Temporality	+	+	+	-
Visualization	++	++	-	++
Indicators	+	+	++	++
Clustering	+	--	--	++

++ Mature functionality

-- Not available or weak

23/26

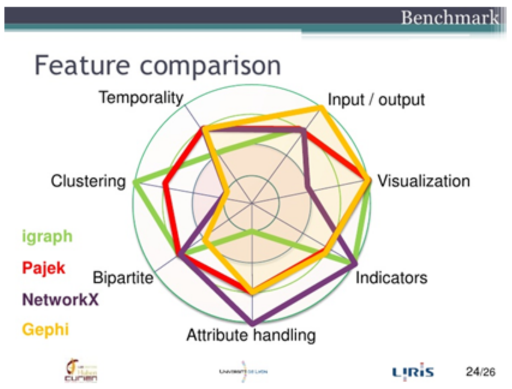


Figure 3.2: Network Analysis Tools comparison, source [3]

The diagrams at Figure 3.2 provide a comparison between all the different network analysis tools.

Tool to be used All of the previous tools can be utilised for the project's purpose. However, igraph matches the exact needs of the project better than the other three; primarily for its strong community detection and clustering algorithms and implementations, in addition to its capabilities to handle very large networks with efficient processing times.

3.4 Network Storage Tools

The analysis included in this project requires the production of many similarity and distance matrices, in which there will be frequent updates and processing. Conventional matrices handling tools will perform well up to a certain size limit, then the system will run out of memory. That is, consider the similarity matrices for the nodes included in the analysis, the system will require an $n \times n$ size matrix for each and every similarity dimension. Where n is the count of the available nodes.

The system and memory consumption will scale well for n size of up to an average scale of 10,000 - 30,000 depending on the available memory, and the type of data stored in the matrix. For bigger matrices, or when the memory is occupied by other running programs or other matrices perhaps, the system will crash.

Possible solutions for this issue include:

- Storing the matrices in the hard disk drive, using Pytables in Python language for example[18].
- Using sparse matrices.

One of the most common characteristics of similarity matrices in general is the sparsity. Similarity matrices represent the similarity between all possible pairs of nodes in the system. In reality, most of these similarities are zeros. Using a regular matrix to represent the similarity matrices would be a big waste of memory. The following table represents an example similarity matrix of 10 users, taken from the community detection methodology chapter.

Users	***0***	***1***	***2***	***3***	***4***	***5***	***6***	***7***	***8***	***9***
0	0	0	0	1	0	0	2	2	0	2
1	0	0	0	0	1	1	0	0	0	0
2	0	0	0	1	0	0	0	1	1	1
3	1	0	1	0	0	0	0	1	1	1
4	0	1	0	0	0	1	0	0	0	0
5	0	1	0	0	1	0	0	0	0	0
6	2	0	0	0	0	0	0	0	0	0
7	1	0	0	0	0	0	0	0	0	2
8	0	0	1	1	0	0	0	0	0	0
9	2	0	1	1	0	0	0	3	0	0

Table 3.1: A sample similarity matrix, from section (4.3.1)

There are 28 similarity values above 0, for 100 different possible values. And this is just a small space example, real life examples would be of a much bigger magnitude. The solution would be to use sparse matrices to represent the similarity, where only non-zero values are stored, and the remaining are assumed to be zeros.

3.4.1 Scientific Library for Python (SciPy)

The SciPy library of Python is an open-source software library that provides comprehensive tools for mathematics, science, and engineering. One of its important components is the sparse matrices support. SciPy supports 7 different types of sparse matrices types, differentiated mostly by the way they are indexed and how the data is fetched[22].

Users	***0***	***1***	***2***	***3***	***4***	***5***	***6***	***7***	***8***	***9***
0	0	0	0	1	0	0	2	2	0	2
1	0	0	0	0	1	1	0	0	0	0
2	0	0	0	1	0	0	0	1	1	1
3	1	0	1	0	0	0	0	1	1	1
4	0	1	0	0	0	1	0	0	0	0
5	0	1	0	0	1	0	0	0	0	0
6	2	0	0	0	0	0	0	0	0	0
7	1	0	0	0	0	0	0	0	0	2
8	0	0	1	1	0	0	0	0	0	0
9	2	0	1	1	0	0	0	3	0	0

Edge	***Weight***	***Edge***	***Weight***
(0-3)	1	(3-9)	1
(0-6)	2	(4-1)	1
(0-7)	2	(4-5)	1
(0-9)	2	(5-1)	1
(1-4)	1	(5-4)	1
(1-5)	1	(6-0)	2
(2-3)	1	(7-1)	1
(2-7)	1	(7-9)	2
(2-8)	1	(8-2)	1
(2-9)	1	(8-3)	1
(3-0)	1	(9-0)	2
(3-2)	1	(9-2)	1
(3-7)	1	(9-3)	1
(3-8)	1	(9-7)	3

Table 3.2: A sample similarity matrix represented as a sparse matrix

Using a sparse matrix to represent the matrix above would result in the following representation:

This representation restricts the elements to be saved to those that contain a non-zero values. In a case where the non-zero values outnumber the zero values this should be quite beneficial.

3.5 Natural Language Toolkit (NLTK)

NLTK is a platform for working with human language data, it is used for Python programs building. It provides easy-to-use interfaces to over 50 corpora and lexical resources such as WordNet, along with a suite of text processing libraries for classification, tokenisation, stemming, tagging, parsing, and semantic reasoning[15].

The NLTK package was used throughout the project for classification problems, the Naive Bayesian Classifier implemented that the NLTK provides was used at the sentiment analysis process, and communities classification. The equation below represents Bayes' rule, the fundamental law behind the Naive Bayesian Classifier.

$$p(C|F_1,\dots,F_n) = \frac{p(C)\ p(F_1,\dots,F_n|C)}{p(F_1,\dots,F_n)}$$

Or in plain English:

$$posterior\,probability = \frac{prior\,probability \times likelihood\,probability}{evidence\,probability}$$

Where the evidence probability is the normalising constant.

The classifier is called "Naive" because it assumes conditional independence for the different features in the system. The resulting posterior probability would then be the product of the probabilities of the individual features, times the prior probability. Which results in the following formula:

$$p(C|F_1, \ldots, F_n) = \frac{1}{N} p(C) \prod_{i=1}^{n} p(F_i|C)$$

Where N is the normalising constant for the features system.

3.6 Sentiment Analysis: Sentistrength

Sentistrength is an unsupervised learning based sentiment classifier. The classified text is given a score of [1,5] for the positivity in the text, and a score of [-1,-5] for the negativity within the classified text. Sentistrength's analysis is based upon:

- A lookup table of sentiment, having words with associated strengths on a scale of 2 to 5.
- A set of additional linguistic rules for spelling correction, negations, intensifying words (like very), emoticons and others.

The positive score for a text is assigned based on the highest scoring positive sentence, and the positive score for a sentence is assigned based on the highest scoring word, within the sentence. This analysis is operated after performing any linguistic modifications of the text at hand. The same procedure applies to negative sentiment scoring[28].

In addition to the sentiment lookup table for the words, Sentistrength also includes an emoticons lookup table, with a sentiment scoring corresponding to each available emoticon. In addition to other sentiment strength measuring procedures. The algorithm has a higher reported accuracy rate than other machine learning approaches for positive sentiment and a similar accuracy rate for negative sentiment strength[29].

Example:

"The text 'I love you but hate the current political climate.' has positive strength 3 and negative strength -4"

Approximate classification rationale: I love[3] you but hate[-4] the current political climate .[sentence: 3,-4] [result: max + and - of any sentence][overall result = -1 as pos<-neg] [23]

Chapter 4

Community Detection, Methodology

The field of Social Network Analysis, and community detection in particular, is of special importance within the context of Social Media content. Obtaining the structural shape of the communication and interaction patterns provides deep understanding of the models of influence and knowledge propagation for a given system. Throughout this chapter I will be illustrating the community detection theoretical background and implementation details used for the scope of this project.

4.1 Overview

4.1.1 Community Definition

The community structure can be defined as essentially a form of grouping or clustering of the nodes. The resulting clusters are characterised by having dense connections amongst the nodes within the community, and sparse connections for the inter-clusters nodes[27]. These connections might have different semantics, as shall be illustrated at the next section, but they all contribute to the communities' structure.

The importance of the community for the context of Social Media, is the fact that communication is more influential and frequent within these clusters, such that people within the same cluster tend to develop similar views and sentiment polarities. As explained at the theoretical background chapter.

The community in the context of social networks in general can be classified as overlapping or non-overlapping, depending on the possibility of the nodes to belong to different communities simultaneously. Throughout this project we will explore the community detection procedures as of non-overlapping nature. The overlapping communities' analysis incurs different techniques that are left for future work.

4.1.2 Multidimensional Heterogeneity

The communication and interaction patterns within the context of Social Media in general is of a multidimensional heterogeneous form[27]. The different nodes, or users, tend to use different forms of communication and knowledge sharing. A good example would be the Twitter content, where the interaction dimensions could be in the shape of Retweeting, Replying, Mentioning or Hashtags similarity. One way to observe these dimensions would be to classify them as either:

- Interaction dimensions, where the nodes engage in a direct interaction form that has two distinct users. In Twitter, such interaction forms include Replies, Retweets, or Mentions (when Mentioning actual users within the dataset).
- Similarity dimensions. This form of connection revolves around individual nodes, expressing individual behaviours. Similarity metrics are then utilised to measure the nodes' similarity given a certain topic or field of operation. Examples for this form within Twitter content would be Hashtags, Links and also Mentions similarity (the Mentions in general, for both users or entities).

Each of these dimensions contribute to the community structure with different proportions. Having these dimensions is generally good for the accuracy of the community detection process. However, this heterogeneity, and the different proportions at which these dimensions contribute to the community detection might be a bit challenging. In the upcoming sections I will be outlining the different experiments and methodologies used for handling these dimensions, and how they contributed to the overall accuracy of the detection process.

4.1.3 Community Detection Dataflow

The general dataflow for the community detection process for Twitter content is visualised in the figure below.

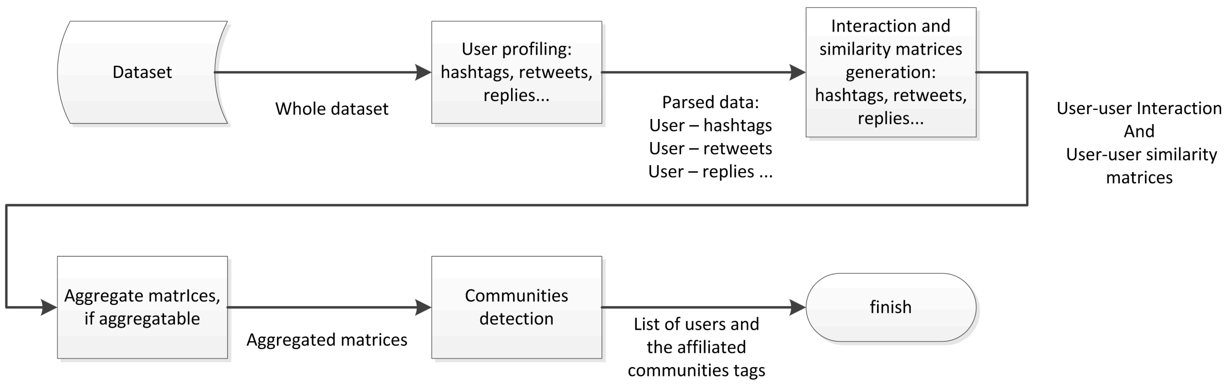

Dataset
Whole dataset
User profiling: hashtags, retweets, replies...
Parsed data: User – hashtags User – retweets User – replies ...
Interaction and similarity matrices generation: hashtags, retweets, replies...
User-user Interaction And User-user similarity matrices
Aggregate matrIces, if aggregatable
Aggregated matrices
Communities detection
List of users and the affiliated communities tags
finish

Figure 4.1: The dataflow of the community detection process

The process begins by the input dataset. The dataset should be in the format outlined at the previous chapter; it should have all the data items affiliated with each Tweet. These items primarily include the user information, Mentions, Hashtags, Retweeters, Reply_to_Tweet if the Tweet was a Reply, Links... etc. The dataset contents are then parsed to generate the user profiles. These profiles are defined as the data items affiliated with each user, in the form of (user, entity), as shall be illustrated at the consequent sections.

The user profiles are then used to generate the interaction and similarity matrices. These matrices are of size $n \times n$, where n is the number of users in the dataset. The intersection of the rows and columns represent the interaction, or similarity, between the different users. These matrices are initialised to zero, and are usually sparse. Some of these matrices can then be aggregated to form a single interaction matrix. A more detailed description of this procedure will be provided at the next sections.

Community detection algorithms are then operated over these matrices, the aggregated and single ones, and the corresponding communities for each matrix are then assigned to the related users. Each user can be assigned to several communities, each community corresponding to one of the interaction or similarity matrices.

4.2 User Profiling

The main concept of user profiling here is to change the dataset contents from Tweets-based into users-based. That is, to parse the contents of the Tweets, and group all the parsed items by the users' IDs. This will represent the user's behaviour in the dataset, based on his/her communication and interaction patterns.

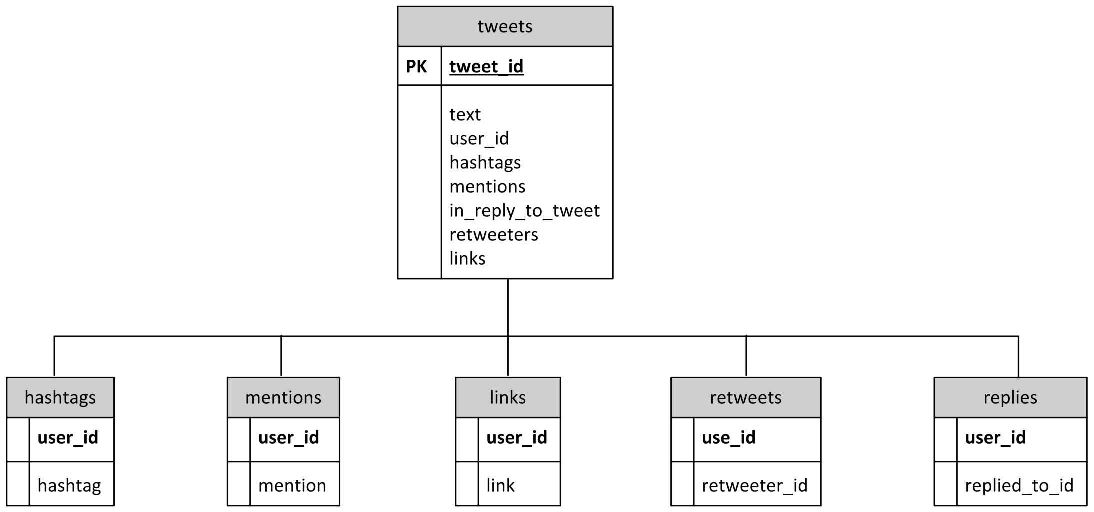


Figure 4.2: SQL database schema for the elements used at the user profiling process

The schema diagram at Figure 4.2 should clarify the procedure. The Tweets table represents the entire available dataset to be processed. The other tables represent the resulting users' profiles corresponding to the users' Hashtags, users' Mentions.... etc.

The table entries here take the form of (user_id, entity). For example, if a user X had 10 different Hashtags in all the Tweets in the dataset, there will be 10 entries in the Hashtags table with the same user_id of X, but with the 10 different Hashtags. An example of user profiling output is illustrated at the table below. Assume we have 10 different users for the example, each with the corresponding Hashtags (a similarity dimension), and Retweets (an interaction dimension), the Retweets represent the users who Retweeted the corresponding user. This example is limited to these dimensions, but the real dataset would constitute of Hashtags, Mentions, Links, Replies and Retweets.

User ID	***Hashtags***	***Retweeters' IDs***
0	#gibraltar, #prime_minister, #cameron	7, 9,2
1	#shale_gas, #fracking, #environment	4, 5
2	#cameron, #cost_of_living	8,3
3	#osborneMP, #cameron, #environment	None
4	#fracking, #polution	1, 5
5	#shale_gas, #polution	4,3
6	#prime_minister, #gibraltar	9
7	#gibraltar, #spain	0, 6, 9
8	#cost_of_living, #osborneMP	0, 2, 3
9	#gibraltar, #spain, #cameron	2

Table 4.1: User profiling example, showing the Hashtags and Retweets affiliated with each user

It is important here to stress on the importance of selecting the SQL server as the DBMS for the project, since that preparing the Tweets table, then the parsing and profiling processes required many join operations over the available tables. It would have been quite problematic to perform the profiling procedure without such operations. For example, a join operation was needed to fetch the user IDs of the users issuing a Reply Tweet. Since that the Reply_to_Tweet field at the original unparsed dataset has a Tweet ID, rather than the user's ID. The same issue applies for the Retweets as well.

The Links, or URLs, in Twitter are usually shortened, to fit in the Tweet's 140 characters limit. There are many URL shortening vendors, which would produce different shortened Links for the same ones. And so, as a part of the Links parsing process to assure accurate results, all the Links are tested to identify the shortened ones, and all shortened Links are then unshortened before storage.

4.3 Building the Network

After performing the profiling procedures, and after attaining a comprehensive information regarding the individual users' behaviour over the available dataset, it is now possible to actually obtain the connections within the different nodes, or users, and establish the network structure. The connections here represent the strength or weight of the interaction, or similarity, amongst two distinct users. The network as a whole represents all the available connections in the dataset, which then could go through a filtering process to discard any noisy or irrelevant edges.

As illustrated at the previous sections, the data in a Social Media mean like Twitter is of a heterogeneous multidimensional nature. And two different classes of connections were identified, the interaction and similarity types of connections. The process of obtaining the connections is different for these two types. The next two subsections are dedicated to discuss the theoretical and implementation details of these connection types.

4.3.1 Building the Similarity Matrix

The similarity dimension has been defined before as the type of connection that revolves around individual users, with individual behaviours. Similarity metrics are used to calculate the similarity measure for any two given users and obtain the connection weight.

For Twitter content, the following dimensions can be identified as similarity connection:

- Hashtags: a kind of metadata tag for the content of microblogging posts in general. Hashtags are identified by a '#' symbol preceding the tag itself. Like #imperial #Twitter.
- Mentions: In Twitter, a Mention is when a Tweet contains another user's Twitter username, preceded by the "@" symbol[25]
- Links, or URLs that are shared at the Tweet post. Unshortened, as discussed at the user profiling section.

The similarity matrix represents the weight of the dual relationship between any two users in the available dataset, with regards to the amount of similar items they share for a given similarity dimension. The matrix is $n \times n$, where n is the number of the users available in the dataset. The sample matrix below represents the Hashtags dimension of the example provided at the previous section.

Users	***0***	***1***	***2***	***3***	***4***	***5***	***6***	***7***	***8***	***9***
0	0	0	0	1	0	0	2	2	0	2
1	0	0	0	0	1	1	0	0	0	0
2	0	0	0	1	0	0	0	1	1	1
3	1	0	1	0	0	0	0	1	1	1
4	0	1	0	0	0	1	0	0	0	0
5	0	1	0	0	1	0	0	0	0	0
6	2	0	0	0	0	0	0	0	0	0
7	1	0	0	0	0	0	0	0	0	2
8	0	0	1	1	0	0	0	0	0	0
9	2	0	1	1	0	0	0	3	0	0

Table 4.2: The resulting Hashtags similarity matrix affiliated with the user profiling Table 4.1

Both the rows and columns represent the users available in the dataset, and the intersection between any user from rows with any users of the columns represents the count of similar Hashtags that they both share.

The similarity for each of these dimensions, for any two users in the dataset, is then calculated as follows.

Implementation, First Version:

The first implementation version takes the literal definition of the similarity matrix. An $n \times n$ sparse matrix is initialised with zeros, where n is the users count in the dataset. Most of the entries in the matrix will be zero after building the similarity, as only a small percentage of the n x n users range will have a similarity value. That's why the matrix was chosen to be sparse, to reduce memory footprint. This matrix is then iterated through to measure the similarity between all pairs of users. The pseudocode for this implementation is as follows:

Algorithm 1 Building the similarity matrix, first version

```
Input: SQL table entries in ($user\_id, entity$) form, entity like Hashtags for example
Output: similarity matrix $S$
 1: Map the user_id into sequential range of $[0, n)$
 2: Initialise the sparse matrix $S$ with the shape $(n, n)$
 3: for all $i$ in $users$ do
 4:     for all $j$ in $users$ do
 5:         if $i =! j$ then
 6:             compute (entity, count(entity)) for entities of user $i$
 7:             compute (entity, count(entity)) for entities of user $j$
 8:             $S(i, j)$ = count(entity) of $i$ + count(entity) of $j$ for similar entities
 9:         end if
10:     end for
11: end for
12: return $S$
```

You can notice in this implementation, however, that the algorithm will iterate through all the possible pairs of possible users, an order of $O(n^2)$. This will incur very bad performance for datasets with large users count.

Implementation, Enhanced Version:

The first implementation approach was focused upon the entities affiliated with the users. Another approach would be to consider the users affiliated with the entities instead. This can be implemented by iterating through the distinct entities in the table, instead of the users, and then check the users that used these entities. A possible pseudocode implementation would be like:

Algorithm 2 Building the similarity matrix, enhanced version

```
Input: SQL table entries in ($user\_id, entity$) form, entity like Hashtags for example
Output: similarity matrix $S$
1: Map the user_id into sequential range of $[0, n)$
2: Initialise the sparse matrix $S$ with the shape $(n, n)$
3: for all $entity$ in $entities$ do
4:     compute (user, count(entity)) for users that used this entity
5:     for all pair of users $i$ and $j$ that used the entity do
6:         $S(i, j)$ = count(entity) of $i$ + count(entity) of $j$
7:     end for
8: end for
9: return $S$
```

```
cursor.execute("SELECT count(*) FROM users")
n = int(cursor.fetchone()[0])
Hashtag_similarity = lil_matrix((n,n), dtype=float)# the sparse matrix
cursor.execute("SELECT DISTINCT Hashtags FROM Hashtags")
Hashtags = [item[0] for item in cursor.fetchall()]
for Hashtag in Hashtags:
        cursor.execute("SELECT id FROM Hashtags WHERE Hashtags = %s", Hashtag)
        users = cursor.fetchall()
        Hashtags_counter = Counter(users)
        users = set(Hashtags_counter)
                for pair in itertools.product(users, repeat=2):
                        Hashtag_similarity[pair[0], pair[1]] += Hashtags_counter[
                            pair[0]] + Hashtags_counter[pair[1]]

return Hashtag_similarity
```

In this implementation we are only iterating through the distinct entities available within the entities table, rather than all the available users. Then we iterate through pairs of users that used this entity. Instead of execution time of order$O(n^2)$**,** the execution time will be of order $O(h \times u)$ or $O(h \times u^2)$ depending on how the code iterates through the pairs of users affiliated with each Hashtag, where h is the number if distinct Hashtags in the dataset, and u is the number of users that used each distinct Hashtag. This should be much more efficient for large users count.

A sample Python implementation of the pseudocode above, for the Hashtags table:

4.3.2 Building the Interaction Matrix

The interaction dimension of the Social Media based connection involves two distinct users, engaged in direct communication. In Twitter, examples of the interaction dimensions include:

- Replies: A Reply is any update Tweet posted by clicking the Reply button on a Tweet[25].
- Retweets: A Retweet is a re-posting of someone else's Tweet. It is sometimes the habit of putting "RT" then the user name of the person of whom they are Retweeting.[26].
- Mentions, for Mentioned users within the available dataset.

The interaction matrix represents the weight of the dual relationship between any two users in the available dataset, with regards of their amount of direct interaction. The matrix is $n \times n$**,** where n is the number of the users available in the dataset. The sample matrix below represents the Retweets dimension of the example provided at the previous section.

Users	***0***	***1***	***2***	***3***	***4***	***5***	***6***	***7***	***8***	***9***
0	0	0	1	0	0	0	0	2	1	1
1	0	0	0	0	2	1	0	0	0	0
2	1	0	0	1	0	0	0	0	2	1
3	0	0	1	0	0	1	0	0	1	0
4	0	2	0	0	0	2	0	0	0	0
5	0	1	0	1	2	0	0	0	0	0
6	0	0	0	0	0	0	0	1	0	1
7	2	0	0	0	0	0	1	0	0	1
8	1	0	2	1	0	0	0	0	0	0
9	1	0	1	0	0	0	1	1	0	0

Table 4.3: The resulting Retweets interaction matrix affiliated with the user profiling table 4.1

Both the rows and columns represent the users available in the dataset, and the intersection between any user from rows with any users of the columns represents the Retweeting between these two users.

As illustrated at the user profiling schema, Figure 4.2, the tables associated with Replies and Retweets are structured in a (user_id, other_user_id) form. The "other_user_id" refers to the user ID of the Retweeter or Replier to the original user. The interaction matrix can be generated by iterating through all the entries at the interaction entity table (Retweets or Replies), and increment the matrix intersection of the user_id and the other_user_id by one. The pseudocode below explains the procedure.

Algorithm 3 Building the interaction matrix

```
Input: SQL table entries in $(user\_id, other\_user\_id)$ form
Output: interaction matrix $S$
1: Map the user_id into sequential range of $[0, n)$
2: Map the other_user_id into sequential range of $[0, n)$
3: Initialise the sparse matrix $I$ with the shape $(n, n)$
4: for all $(i, j)$ pair in table entries do
5:     $I(i, j)$ ++
6: end for
7: return $I$
```

A sample Python implementation of the previous pseudocode, for the Replies table, is available below.

```python
cursor.execute("SELECT count(*) FROM users")
n = int(cursor.fetchone()[0])
Reply_interaction = lil_matrix((n,n), dtype=float) #the sparse matrix

cursor.execute("SELECT user_id , Replier_id FROM Replies")
for item in cursor:
        Reply_interaction[item[0], item[1]] += 1

return Reply_interaction
```

4.3.3 Aggregation

Some connection dimensions might perform very well alone, with aggregating or adding up their network matrix with other dimensions. While other dimensions might perform much better if they were part of a group of aggregated dimensions. The aggregation process requires some form of normalisation of the individual matrices, before joining to make the aggregate matrix.

The evaluation chapter will be reviewing and analysing the aggregation process and the effect it has on the overall community detection performance.

4.4 Community Detection

The theoretical background chapter provided a good review about the community detection as a process, and samples of community detection algorithms. This section will be dedicated to the implementation issues related to community detection, in terms of the inputs, outputs, algorithms characteristics and parameters.

4.4.1 Input

The general community detection process takes a graph information as an input, and generates a clustered graph as the output. igraph's community detection algorithms in general takes a number of different graph types representing the network at hand.

The best input type, matching the output of the network matrix would be the *Edge List* format. As the output of the sparse matrix in Python's (or Scipy's) Coordinates sparse matrix (coo_matrix) would be in the format of list of matrix coordinates, which in essence represents the edge-list amongst the users in the network matrix. However, the edge list format doesn't allow for weighted edges to be considered. And so, the best input type for this project would be the labeled edge-list (NCOL), in the (vertex1, vertex2, weight) format[8].

4.4.2 Processing: Community Detection Algorithms

The theoretical background outlines a number of different community detection algorithms. The issue now resides on how to choose of these different types. The input dataset for the community detection algorithm in this project in general, and the analysis process that follows, have the following characteristics:

- Weighted edge list.
- Undirected edges.
- Adjustable parameters of number of clusters, and edge-weight threshold.

The parameter of number of clusters is important for the evaluation process, to analyse the effect of changing the clusters count upon the accuracy and quality of the analysis. The algorithms that support clusters count setting are those that output a *Dendrogram* graph, allowing for graph cuts at different levels. The clusters count parameter can then be used as an input for the algorithm, which makes a cut at the *Dendrogram* at a certain level, producing a specific clusters count.

The primary tool to be used for community detection is igraph, refer for igraph section under Tools and Software Packages for more details. There are 3 algorithms in igraph package that could handle these requirements: *Fastgreedy* (modularity optimisation), *Edgebetweenness* (hierarchical clustering) and *Walktrap* algorithms (random walk) [8]. The general description of each of these algorithms' types and sample implementation of some of the algorithms can be found at the theoretical background chapter.

4.4.3 Output

The output of all of the Mentioned algorithms at the previous section is a *Dendrogram* graph, more details at the next section. The *Dendrogram* graph can then be converted to regular graph clustering, which can be iterated through. The system then iterates through all the vertices within the resulting communities, and extracts the community name for each of the vertices, users in our case, and assigns it for the corresponding user in the users SQL table.

So the overall output of the community detection process would be the users' community tags. These tags would then be easily accessible for further processing from the users SQL table.

It is worth Mentioning here that some users can be unassigned to any communities. The reason can be due to the lack of any entities affiliated with this user, no Hashtags for example, which provides no information for the community detection algorithm. Or because there are no similar entities shared or no interaction happening between this user and any other users.

4.4.4 *Dendrogram* Graph

The *Dendrogram* is a tree based structure that represents the clusterings produced mostly by hierarchical clustering methods. The *Dendrogram* shows the most correlated nodes at each level grouped together[8]. To be able to generate a general clustering graph, as the graph on the right, the *Dendrogram* can be cut at certain levels to provide the required number of clusters.

The figures below show the community detection procedure's output for the example Mentioned earlier in the chapter. The figures show the *Dendrogram* diagram (on the left), and the corresponding clustering diagram (on the right).

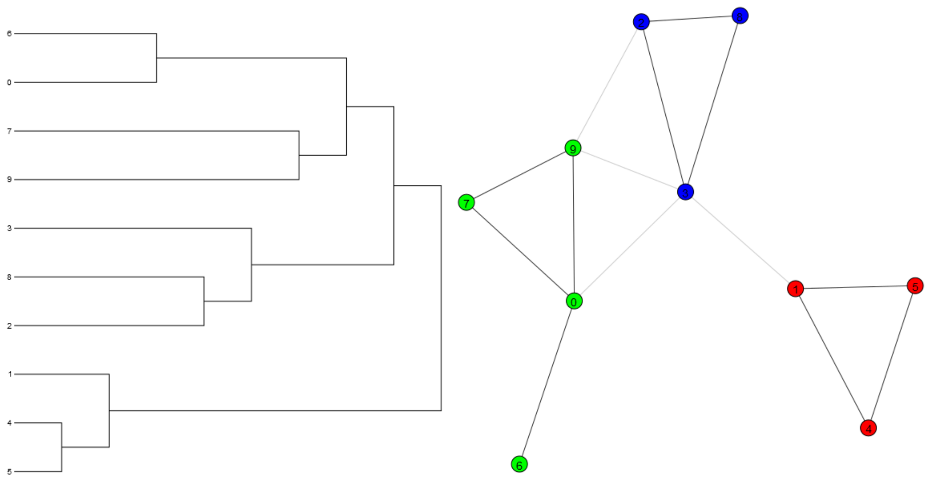


Figure 4.3: The resulting Hashtags' communities of the example dataset Table 4.1. Dendrogram diagram (on the left), and the corresponding clustering diagram (on the right)

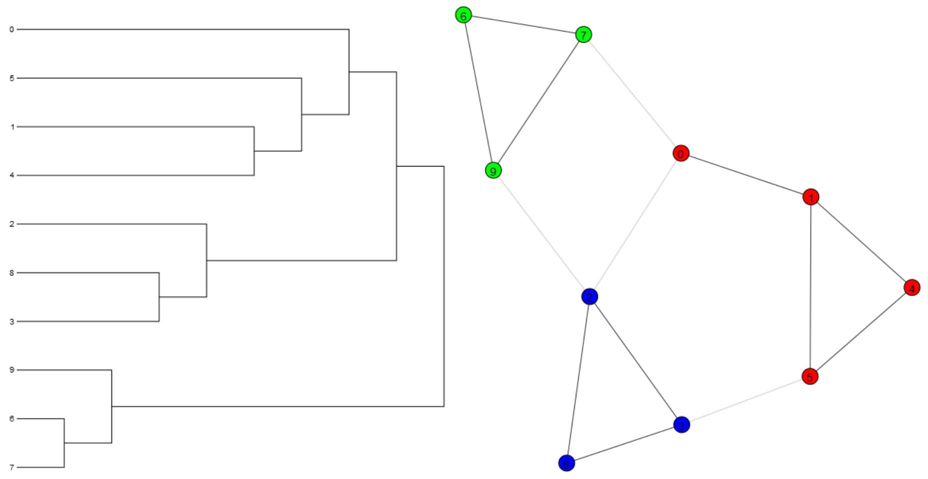


Figure 4.4: The resulting Retweets' communities of the example dataset Table 4.1. Dendrogram diagram (on the left), and the corresponding clustering diagram (on the right)

4.5 Performance Issues

The methodology proposed at the project is not limited to certain or specific datasets rather than others, the approach is a general concept that is applicable to all datasets as long as they have the required data elements for the analysis process. Such data elements for the problem of Twitter content include the Hashtags, Retweets, Replies, Links and Mentions affiliated with each Tweet. In addition to user information and timestamp of the Tweet itself.

However, the choice of the dataset might have a number of direct or indirect impacts on the analysis process itself, and for the environment at which the analysis process is taking place. These issues might include:

- Memory footprint.

- Processing time.
- Quality of results.

Some of the dataset's properties that affect the previous issues can be summarized at the following table. The analysis quality components of the table will be further discussed at the evaluation chapter.

	Processing time	***Memory***	***Analysis quality***	***Metric***
Large dataset size	Little negative impact	Medium negative impact	Positive impact	
Large number of different users	Big negative impact	Big negative impact	Positive impact	
Large Users/dataset-size ratio*	No impact	No impact	Big negative impact	Users/Tweets
Dataset Topic**	No impact	No impact	Big impact	

Table 4.4: Dataset properties affecting the community detection performance

* One of the main issues or problems affecting the analysis here is the users to dataset size ratio. A big ratio indicates that the communication patterns among the users within the available dataset is sparse, which means that a big percentage of the available Tweets are individual isolated Tweets and not in Reply or Retweet for other users or Tweets. This singleton nature of the data poses some problems for the community detection step of the analysis, where a big portion of the data utilized for the community detection part is derived from the interaction patterns amongst the users.

It's worth Mentioning here though that a small ratio does not indicate better quality; as the small users count can be the result of spammers and spam content, or due to extensive interaction within a few users only of the entire users group.

** Proper selection of the topic of the dataset or the topic for which the sentiment analysis is run upon is of great importance for the applicability of Social Network Analytics (SNA). The SNA constructs that are being used at the project are mainly about community and centrality analysis. Community detection analysis provides good results when the dataset is expected to actually contain communities of different views or sentiments.

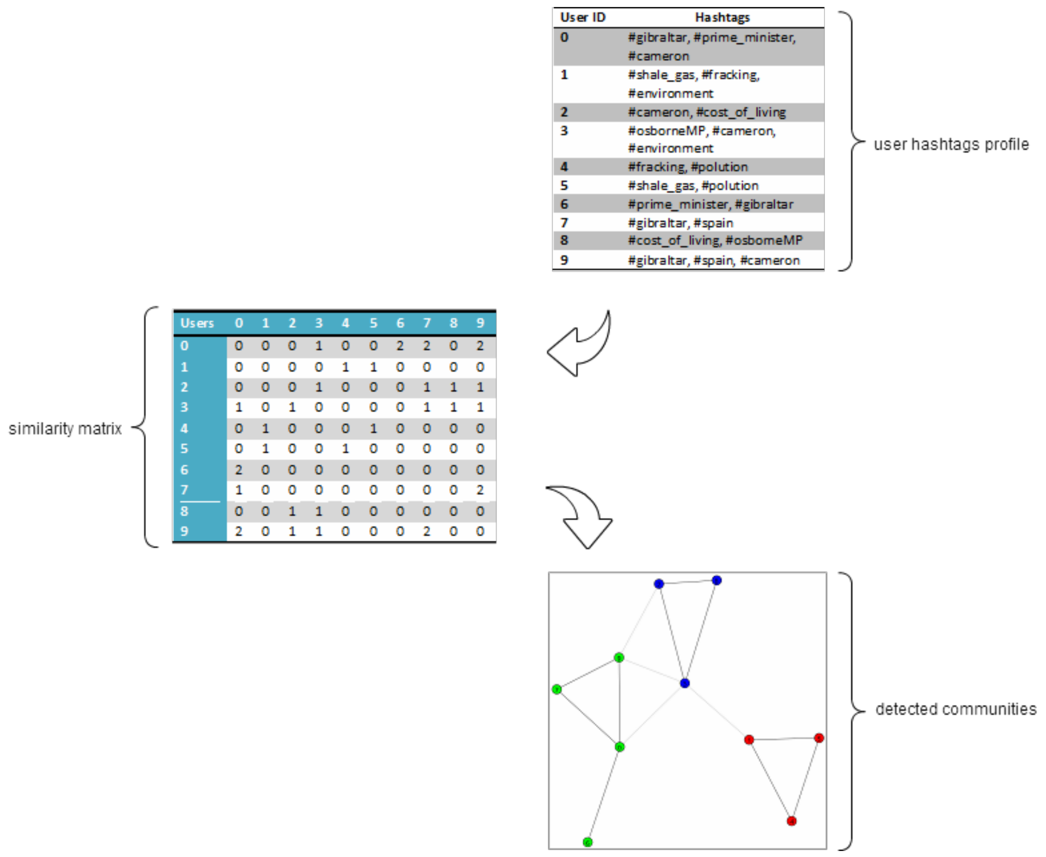
4.6 Summary

This chapter presented a methodology and provided the supporting algorithms and implementations for carrying out Link-based community detection procedures for Twitter content. The main process relies primarily on the interactions users tend to have with other users on the network, along with the individual behaviours, forming some sort of virtual communities characterised with dense connections within the community, in comparison to inter-cmmunities.

The chapter discusses two different types of connection patterns or dimensions found in Twitter. The first type being the interaction dimensions, which involve two or more users engaged directly in a form of interaction. Examples for this type include Retweeting, Replying and Mentions. The other connection type is based upon similarity. This kind of connection relies more on the individual behaviours of users, and how similar these behaviours might be to other users' behaviours over the network, Examples of this type of connection in Twitter include Hashtags, and URLs.

The process of detecting the communities passes through the steps of first user profiling; in which the general behaviours and communication patterns of individual users are extracted of the dataset. These profiles are then used to generate interaction and similarity matrices amongst all the available users. These matrices might be aggregated, to form an overall distance matrix, or treated individually resulting in dimension-based communities. These matrices, aggregate or individual, form the basis for the community detection process that follows.

The diagram below outlines the main steps that the example provided in this chapter has went through throughout the community detection process.



User ID	Hashtags
0	#gibraltar, #prime_minister, #cameron
1	#shale_gas, #fracking, #environment
2	#cameron, #cost_of_living
3	#osborneMP, #cameron, #environment
4	#fracking, #polution
5	#shale_gas, #polution
6	#prime_minister, #gibraltar
7	#gibraltar, #spain
8	#cost_of_living, #osbomeMP
9	#gibraltar, #spain, #cameron

Users	0	1	2	3	4	5	6	7	8	9
0	0	0	0	1	0	0	2	2	0	2
1	0	0	0	0	1	1	0	0	0	0
2	0	0	0	1	0	0	0	1	1	1
3	1	0	1	0	0	0	0	1	1	1
4	0	1	0	0	0	1	0	0	0	0
5	0	1	0	0	1	0	0	0	0	0
6	2	0	0	0	0	0	0	0	0	0
7	1	0	0	0	0	0	0	0	0	2
8	0	0	1	1	0	0	0	0	0	0
9	2	0	1	1	0	0	0	2	0	0

Figure 4.5: The overall community detection steps, example

Chapter 5

Community Detection, Analysis and Evaluation

5.1 Overview

The community detection process is based on latent analysis. In order to properly understand the factors contributing to the community structure, and to evaluate the proposed methodology and software package, a comprehensive evaluation process is extremely vital. The evaluation process was designed with the main project's objectives affiliated with community detection taken as an end goal. The main evaluation objectives discussed at this chapter are to measure how efficiently the proposed procedures capture the potential communities in a dataset. And to analyse and understand what factors and parameters affect the overall detection process. The chapter also aims at identifying the execution time complexity affiliated with the implementation.

The evaluation process has been divided into two categories; supervised and unsupervised evaluation approaches. The supervised approach assumes the availability of labeled data or ground truth information regarding the dataset. This approach will be used primarily to evaluate the methodology and software package. The unsupervised approach reflects real life problems, where there usually isn't, or not easily accessible, ground truth information. This approach is primarily used to study the parameters affecting the system performance.

5.2 Case study A: Members of the Parliament (MPs) Dataset, Outbound Only

This case study revolves around the current UK members of the parliament. The dataset covers all the Tweets that involved any of the members in the outbound direction; that is, Tweets that were issued by the members of the parliament themselves, rather than Tweets from other people involving the members by Mentioning or Retweeting.

5.2.1 Dataset Collection

The dataset was collected by crawling the official Twitter accounts related with each member. The crawling process was grouped by the political affiliation. The process progressed as follows:

1. Lists for all UK MPs, grouped by political affiliation, were obtained from the website Tweetminster.co.uk. Tweetminster.co.uk is a website that collects and analyses media content related to the UK politics[31].

2. A database was built for these members' details, including each member's political affiliation.

3. The crawler should be configured in a way to collect outbound Tweets only, that is, Tweets issued by the MPs themselves. This should guarantee that we know the political affiliation of each user in the dataset, where including inbound Tweets doesn't guarantee such thing.

4. The Twitter crawling is now operated for each member of each political party. The collected Tweets are filtered through the collecting API, Trendak, to include the Tweets with political content only. This enhances our analysis as we will evaluate the results later on relative to the political affiliations of the MPs.

The resulting dataset covers all the UK MPs, each member and affiliated Tweets tagged by the political affiliation.

5.2.2 Dataset Statistics

The following numbers represent the data items collected for the MPs dataset:

Item	***Count***
dates	29/7 - 05/8 2013
users	300
Tweets	9266
unique Hashtags	1380
total Hashtags	4836
unique Mentions	5380
total Mentions	23630
unique Links	1302
total Links	1589
Replies	119
Retweets	244

Table 5.1: Statistics related to the MPs Outbound Only dataset

5.3 Evaluation Metrics

The main goal of the evaluation process to measure by what accuracy the system was able to capture the community information. The ground truth information are compared with the detected communities by utilising clustering evaluation metrics, like $F-score$, $Purity$ and $B-Cubed$.

5.3.1 Evaluation by Set-Matching: $F-score$ and $Purity$

Evaluation metrics provide a systematic and unified way into evaluating the performance of the system for different algorithms and parameter values. One of the possible types of evaluation is the evaluation by set matching. The set of the available political parties affiliation will be the ground truth information to compare against. The set of captured community information will be the subject of the comparison, to be compared against the ground truth[4].

Purity

Purity for a clustering problem is basically the weighted average of the precision values:

$$Purity = \sum_i \frac{|C_i|}{N} max_j Precision(C_i, L_j)$$

Where:

C_i is the cluster i from the detected communities.

L_j is a given ground truth category j

and the precision of a cluster C_i for a given category L_j is denoted as:

$$Precision(C_i, L_j) = \frac{|C_i \cap L_j|}{|C_i|}$$

The main idea of Purity is to assign the cluster to the most frequent class within the cluster, the accuracy is then measured by counting the number of correct assignments for each cluster. Weighted average is then computed for all the cluster to find the over all metric.

Purity is a good measure to detect and scale down with noisy elements within the clusters. However, there are several issues regarding Purity as a sole evaluation metric that renders it inefficient:

- Purity tends to be biased to small clusters, that is, small clusters are more likely to attain better Purity measures.
- Purity is associated with individual clusters alone, so changes with other clusters won't have an effect on the individual score. That is, Purity does not reward the grouping of items of different clusters from the same category together.[4]

Purity is, however, still important and provides some important information about the individual clusters. But it is better used along with other metrics.

F-Score

F-Score metric considers the precision, and recall into the evaluation process. The recall of a cluster C_i for a given category L_j is denoted as:

$$Recall(C_i, L_j) = \frac{|C_i \cap L_j|}{|L_i|}$$

The recall and precision can then be combined using Van Rijsbergen's formula[20]:

$$F(R, P) = \frac{1}{\alpha(\frac{1}{P}) + (1-\alpha)(\frac{1}{R})}$$

Where R is the Recall, P is the precision, and α is the relative weight of the metric. If we assigned α a value of 0.5, the resulting metric would be the harmonic average of R and P:

$$F(Recall, Precision) = \frac{2 \times Recall \times Precision}{Recall + Precision}$$

F-Score is good balanced metric for evaluating clustering problems. F-Score prefers coarse clustering rather than the small clusters Purity opts for. F-Score, however, shares the same problem as Purity. F-Score is associated with individual clusters alone, so changes with other clusters won't have an effect on the individual score.

Increasing the number of clusters would severely reduce the F-Score result, without rewarding or the possibility of grouping items of different clusters, from the same category, together[4].

5.3.2 Evaluation by Set-Matching, based on Counting-Pairs Approach: $B-Cubed$

It can be inferred, from the previous discussion, that set matching evaluation metrics alone are not the best possible candidates for clustering evaluation. Another approach for evaluating clusters can be the Counting Pairs; by considering statistical results over pairs of items at a time[4]. The counting pairs approach is node, or item, based rather than cluster based, where the analysis is operated over any pair of items in the distribution. Mixing both counting-pairs with set-matching approaches would provide a clustering evaluation metric that cover all of the issues Mentioned before. An approach that is node based, but evaluates clusters.

An important example of such methodology is the B-Cubed metric. B-Cubed metrics breaks down the evaluation process for estimating the B-Cubed precision and B-Cubed recall affiliated to each item, rather than cluster, in the distribution. The item B-Cubed precision reflects the amount of items in the same cluster, that belong to the item's category. Similarly, The B-Cubed recall affiliated with an item reflects the amount of items from its category that appear in its cluster[4].

A correctness measure can be defined to represent the correctness between two edges e and $e\prime$:

$$Correctness = \begin{cases} 1 & iffL(e) = L(e\prime) \leftrightarrow C(e) = C(e\prime) \\ 0 & otherwise \end{cases}$$

The B-Cubed precision of an item can be defined as the proportion of items in the item's cluster, that share the item's category. The overall B-Cubed precision can be calculated by average the precision of all items in the distribution. And because the average is calculated over items, there's no need to use weighted mean of the clusters or categories. The same approach is used for calculating the B-Cubed recall, by Replacing "cluster" with "recall" in the reasoning above[4].

$$Precision\,BCubed = Avg_e[Avg_{e\prime.C(e)=C(e\prime)}[Correctness(e, e\prime)]]$$

$$Recall\,BCubed = Avg_e[Avg_{e\prime.L(e)=L(e\prime)}[Correctness(e, e\prime)]]$$

5.4 Supervised Evaluation Approach

5.4.1 Overview

The main idea of the supervised evaluation, is the evaluation of the detected communities relative to some ground truth information. The ground truth information here are the available political affiliation tags for each MP. The resulting community tags of the community detection procedure are compared against the already available political tags, by means of evaluation metrics illustrated before. This approach should provide viable results as the Tweets collected for the dataset are filtered first, to make sure to keep Tweets with political content only.

5.4.2 Experimental Setup

Objectives

The main goals of the supervised evaluation approach is to measure the accuracy of the overall system performance. Sub goals of the experiment might include:

- Analysing and investigating the effect of changing the parameters of number of clusters and edge-weight threshold upon the overall system performance.
- Studying the efficiency of community detection for each individual dimension, in comparison to aggregated dimensions.
- Investigating and comparing the different evaluation metrics.
- Studying the efficiency by which the system could accurately perform the community detection procedures.

Dataset

The dataset for this study will be the Members of the Parliament dataset described previously. Refer to the dataset section for more comprehensive information.

Parameters

1. Number of clusters.
2. Edge weight threshold.

Algorithms

- *Fastgreedy.*
- *Walktrap* (results in the appendix).
- *Edgebetweenness* (results in the appendix).

All used algorithms are used as part of the igraph software package.

5.4.3 Methodology

The evaluation has been performed over three different community detection algorithms; *Fastgreedy*, *Walktrap*, and *Edgebetweenness* algorithms. The *Fastgreedy* algorithm has performed the best among the available three algorithms. Hence the observations and analysis of the *Fastgreedy* algorithm will be presented in this section, while the results of the remaining algorithms are to be found at the appendix. The evaluation process for the parameters is performed by adjusting one of the parameters while fixing the other.

The edge-weight threshold is adjusted by taking 15 steps of variable sizes depending on the data's nature. If the data is big and provides ranges of over 15 different values, then step size of 1 is taken starting from the minimum value of the available edge-weights. If the range of the data is not large enough, the step size is calculated as below, with the first analysed value being zero.

$$step = \frac{maxValue - minValue}{15}$$

For each threshold value, the entire community detection procedure is performed. The evaluation metrics are then calculated over the resulting communities, and the overall metric value for all the communities is plotted at the corresponding chart. The resulting overall chart would then represent the evaluation metric for all the detected communities, for each threshold value.

The number of communities is adjusted at a fixed range of 2-20 clusters. The same procedure as in the edge-weight threshold parameter is then performed; by operating the community detection over each cluster count, calculating the metrics values and plotting the results. It is important to note here that many of igraph's community detection algorithms accept an input specifying the number of returned communities. However, this count is considered as the maximum value for the clusters count, rather than a fixed value. The algorithm will then decide the optimal clusters count given this maximum. For example, if the input clusters count was 10, but the algorithm found the optimal clusters at a count of 7, the resulting number of communities will be 7.

The community detection algorithm might return an error that no optimal clusters structure can be found at a given clusters count input. The system in this case is designed to skip this clusters count value and move to the other, that's why the following clusters charts sometimes don't show the full range of 2-20 clusters input. The community detection algorithms chosen for this analysis all provide a *Dendrogram* graph. The *Dendrogram* graph, shown at Figure 5.1, can be converted to a regular clustering graph by cutting at a certain level. The input clusters count is used to determine this level and provide the desired cluster count, based on the optimal structure.

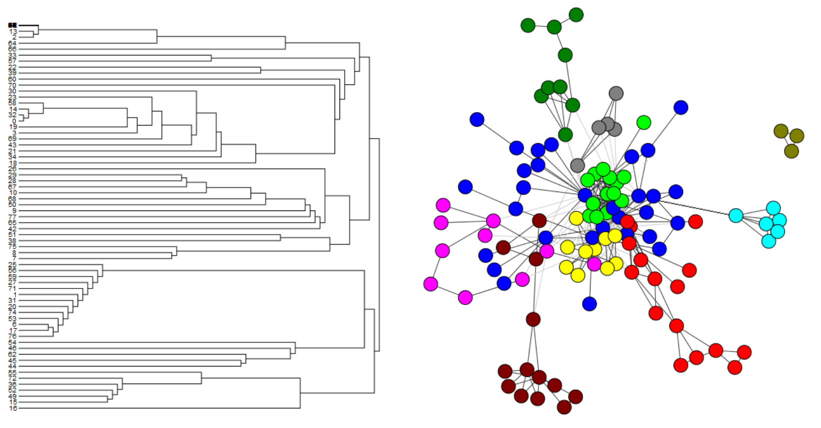


Figure 5.1: A sample Dendrogram graph (left), and the resulting clustering graph (right). For the Hashtags dimension, case study A

Charts types

Different chart types are used based on the nature of the data to be plotted.

- Plot box chart: used here when each point in the data for the chart is an average of a number of values. Like in the Purity metric.

- Line chart: used for general data plotting. Data examples include the Purity (the mean value), B-Cubed and F-score metrics results.

- Bar chart: used to show data distributions, as in the distribution of users and Hashtags for each detected community.

5.4.4 Experiment 1: Analysing the Different Metrics and Parameters

Objectives

- To evaluate and compare the available clustering evaluation metrics given the project's context.

- Investigate the effect of the parameters change upon the overall performance, measured by the metrics.

Observations and Analysis

The charts below show the resulting Purity values as a result of the edge-weight threshold and the number of clusters parameters for the ”Hashtags” dimension, detected communities:

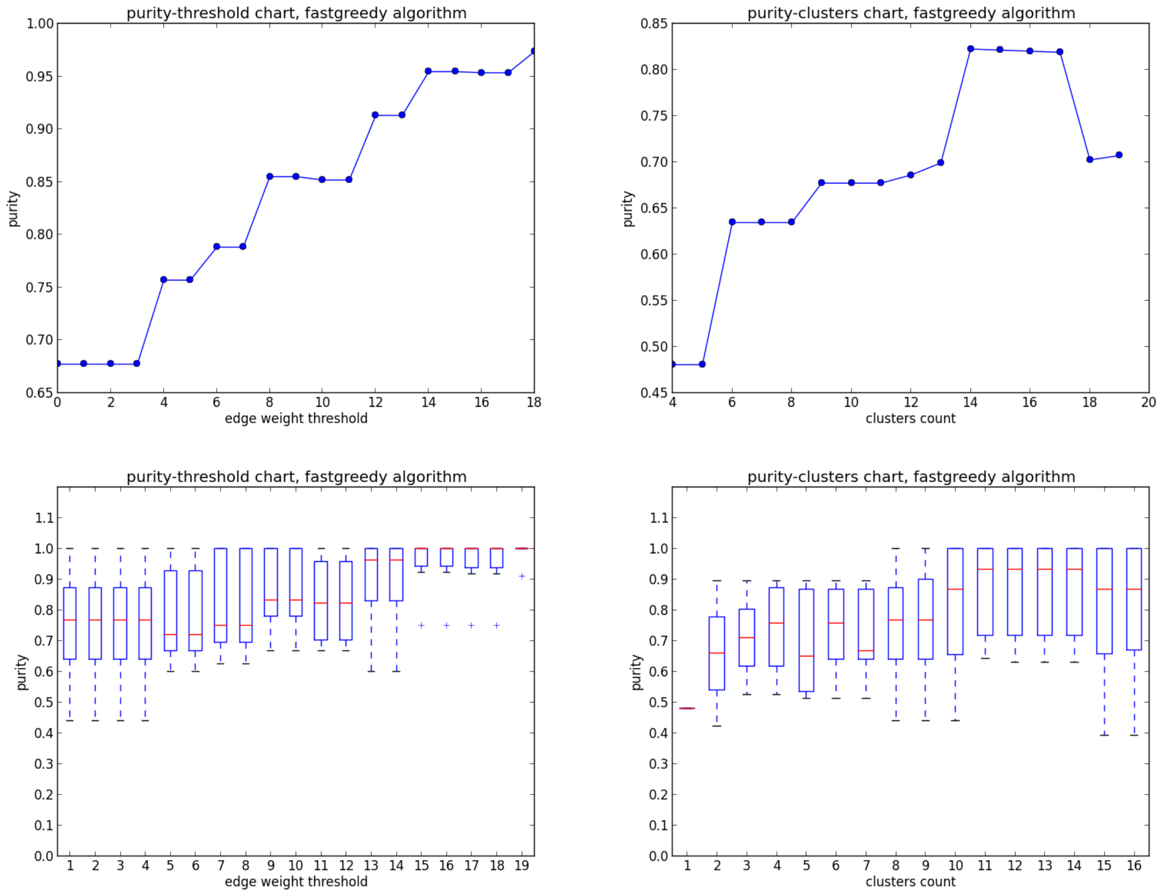


Figure 5.2: The Purity metric charts associated with the edge-weight and clusters parameters of the Hashtags dimension. Case study A

We could infer from the charts that the overall Purity of the ”Mentions” dimension is above 50% for most of the cases, and it achieves much better results for higher threshold and clusters values. We could also infer that the Purity in general tends to increase with increasing the number of clusters, as the number of users per cluster drops and only ”pure” members remain. The Purity also increases with increasing edge-weight threshold, as the noisy edges get eliminated. You can also observe that the deviation in the Purity values, illustrated through the Box Plot chart, is relatively high. This makes the Purity alone a not efficient measure as grounds for comparison.

The following chart shows the F-Score values, again with the parameters of edge-weight threshold and number of clusters, for the ”Hashtags” dimension.

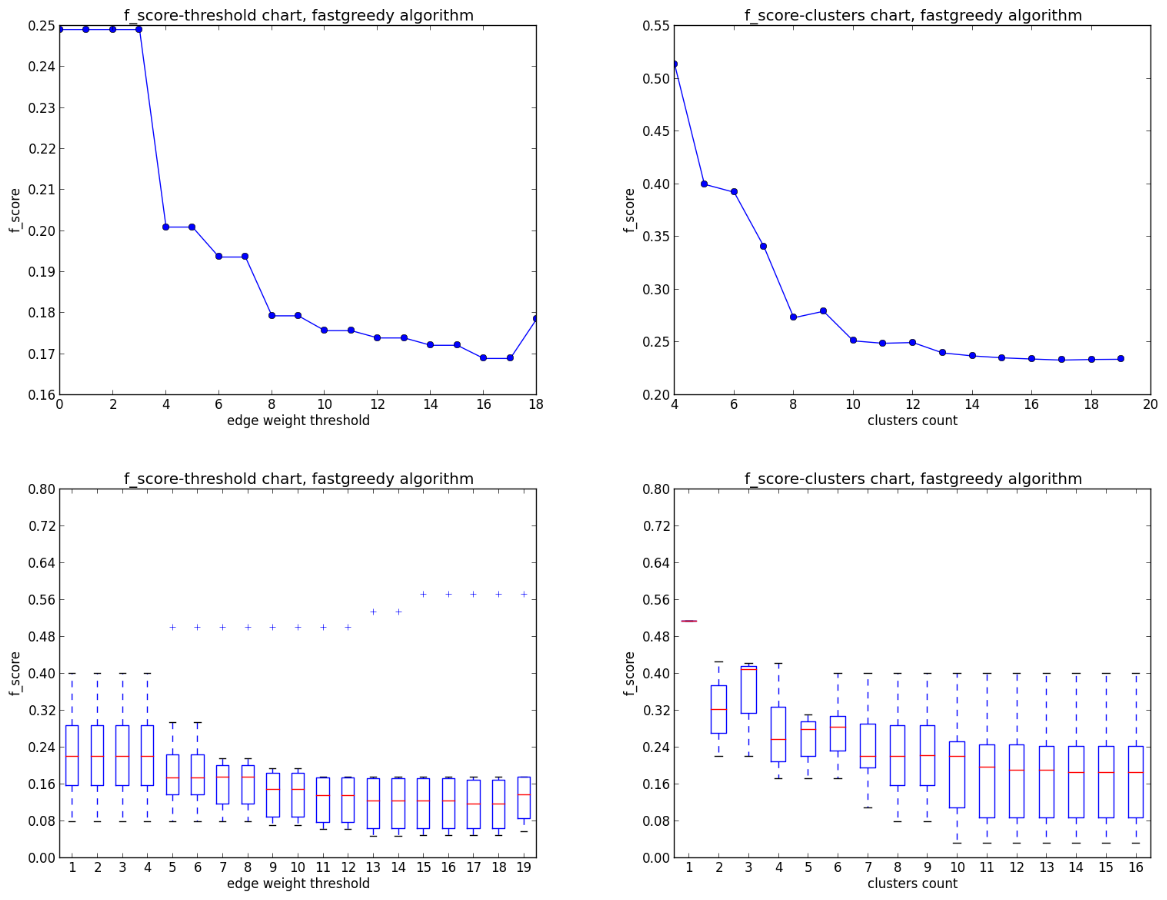


Figure 5.3: The F-Score metric charts associated with the edge-weight and clusters parameters of the Hashtags dimension. Case study A

The first note is regarding the F-Scores low values. The main reason for the low values was because of the large cluster count the the community detection algorithm has assigned to the system by default (12 clusters in this case). As the clusters count hasn't been specified as an input. You could observe the better F-Score ratings on the clusters count diagram. When the clusters count was specified as an input, with small reasonable range, it produced much better results.

Again as in Purity, the deviation of the mean tends to be a bit high as we can observe from the box plots, so again F-Score alone wouldn't be a good metric on its own. We can also infer from the charts that the F-Score value tends to drop as we increase the number of clusters and edge-weight threshold. The reason for that being the big drop of users count per cluster in both cases, even if the accuracy, or precision, of each cluster increases.

The F-Score values in general tend to be a bit small in our system as presented above. The reason being the relatively big number of clusters in the detected communities (around 8 in general) in comparison to the ground truth clusters (4 clusters). That's why we observe the biggest F-Scores around clusters count of 4, mainly because the clusters count is close to the ground truth clusters count.

The charts below represent the B-Cubed metric results of adjusting the parameters. As illustrated before, the B-Cubed metric is element-based and the metric calculation is operate on the overall precision and recall. Hence, there is no averaging for individual B-Cubed scores and there's no need to show the box plot charts.

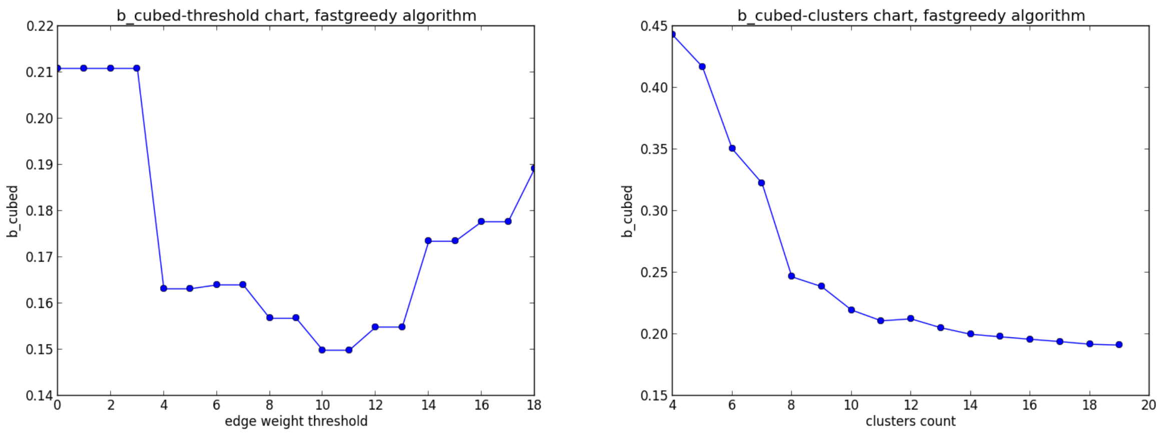


Figure 5.4: The B-Cubed metric charts associated with the edge-weight and clusters parameters of the Hashtags dimension. Case study A

As in F-Score the B-Cubed values tend to drop with increasing clusters count or edge-weight threshold. The reason being the same; less users per cluster, which would decrease the overall performance.

Conclusions

- Other evaluation metrics provide good information about the system behaviour, however, B-Cubed metric is the best metric for comparisons in general, as it's an item-based and doesn't average individual values.
- Purity, or precision, of the system tends to increase as we increase the number of clusters and the edge-weight threshold. As the number of users per cluster drops, and the more "pure" users remain.
- The F-Score and B-Cubed values tend to drop as we increase the number of clusters and the edge-weight threshold. The reason being the increase in clusters count and the fewer users per cluster decreases the system recall.
- The optimal parameters setting should take into consideration both the system Purity and B-Cubed values. For the dimension above, an edge-weight threshold setting of around 6, and clusters count of around 8 should provide good results; Purity of 79%, and B-Cubed value of around 34%.

5.4.5 Experiment 2: Aggregating Similar Clusters

Objectives

- To study the effects of aggregating similar clusters in the detected communities, which reduces the overall clusters count, over the overall system performance.

Approach

As it has been Mentioned at the previous analysis, increasing the clusters count or the edge-weight threshold would enhance the Purity of the resulting communities, but would decrease the F-Score and B-Cubed. The reason being the fact that there are less users per community, and more communities for the clusters count parameter. An interesting experiment here would be to combine all the detected communities that share similar ground truth information (political parties affiliations in our case), and re calculate the B-Cubed metric. According to our previous analysis, doing this should increase the overall B-Cubed for all parameter values.

The procedure for this experiment would be as follows:

1. Apply the community detection procedure as normal.
2. Calculate the Purity of ground truth category (political party affiliation in our case) for each detected community.
3. For each detected community, assign a tag that is associated for the maximum Purity category. For example, if the category (political affiliation) "Conservative" gave the highest Purity for a given detected community, assign the tag of this community to "Con".
4. Aggregate all the detected communities that share the same tag together.
5. Calculate the B-Cubed metric for the aggregated and compare the results with the B-Cubed values without aggregation.

Observations and Analysis

The resulting charts of the B-Cubed metric were as follows:

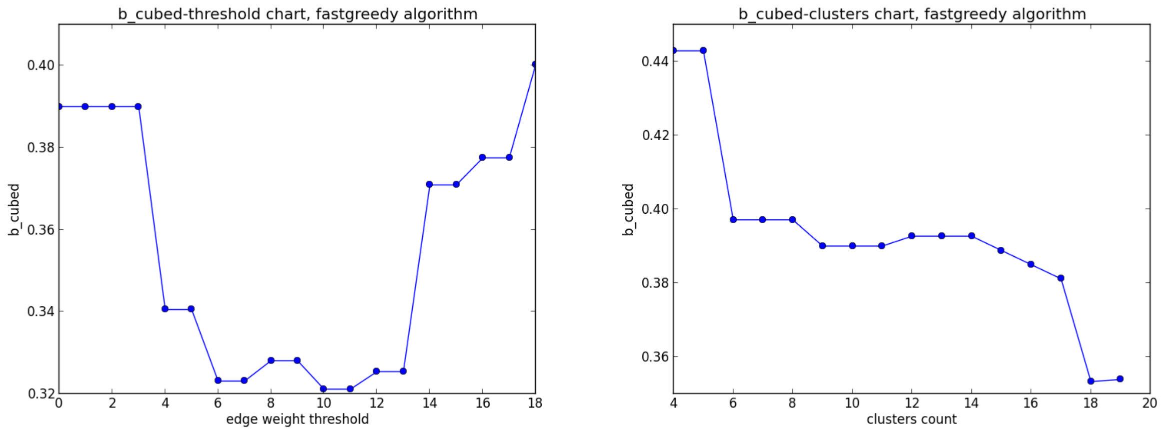


Figure 5.5: The B-Cubed metric results charts associated with the Hashtags dimension, after aggregating similar clusters. Case study A

It can be observed clearly from Figure 5.5 that the overall B-Cubed values are much better for the aggregated clusters than without aggregation as in Figure 5.4. It is also observable that the B-Cubed values are relatively stable and when increasing the clusters count parameter. This fact is expected as no matter how more clusters there are, they will eventually be aggregated.

The variations after clusters count of around 16 are primarily due to the fact that the users distribution within the clusters became different, which affected the clusters' precision and purity.

Conclusions

- Aggregating similar clusters will greatly affect the system performance, as the reduced clusters count after aggregation will increase the B-Cubed metric. The effect over the analysis would be much more accurate results for the detected communities, to be utilised at any consequent studies.
- It was easy to aggregate the clusters here as we already have ground truth information to base our analysis upon. It is important to also be able to do such aggregation without having ground truth information, but it would be much more challenging.

5.4.6 Experiment 3: The Effect of the Different Dimensions Over Performance

Objectives

- To study the effect of the individual dimensions over the overall system performance.
- To investigate the effects of aggregating different dimensions over the system performance.

Approach

The previous chapter about the methodology of community detection, discussed the multi-dimensional heterogeneity of Social Media in general. In Twitter, two distinct classes of dimensions were identified. These dimension classes include interaction dimensions; which are represented by Retweeting, Replying and Mentioning. And similarity dimensions; which include Hashtags, Links and Mentions. It is essential at this stage to measure how these different dimensions measure on the metrics individually, and what combinations of the dimensions could enhance the overall performance. The main approach would be to calculate the metrics for the detected communities resulting of each individual dimension, then calculate the metrics for some combinations of these dimensions.

This experiment and the affiliating results' charts are quite big in quantity, and so, only the highlights and important results will be presented here, and the remaining details are to be found at the affiliated appendix.

Observations and Analysis

As illustrated before, the dimensions of connection within the Twitter content have been divided into similarity and interaction dimensions.

Similarity dimensions:

The results for the Hashtags dimension alone can be found at Figure 5.4.

The Links similarity dimension's results are below:

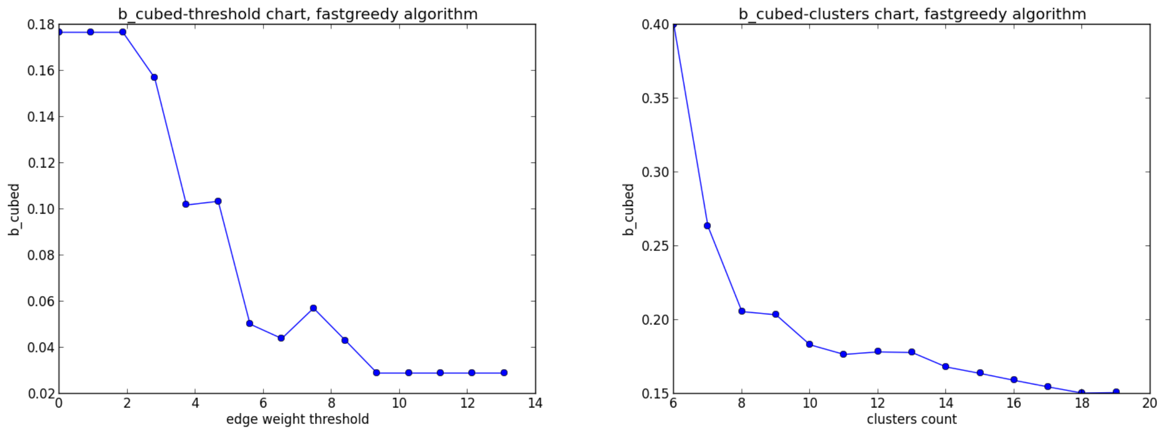


Figure 5.6: The B-Cubed metric results for the detected communities of the Links dimension, case study A

It can be observed that both the Hashtags and Links dimensions didn't perform very well on the edge-weight threshold chart. A normalised aggregation of both dimensions can be found below. The resulting chart shows a noticeable improvement over the individual Links or Hashtags dimensions. An important note here is that the edge-weight threshold here is in fractions rather than integers as before, the reason being for the normalisation process lowering the overall range of dimension to below 1.

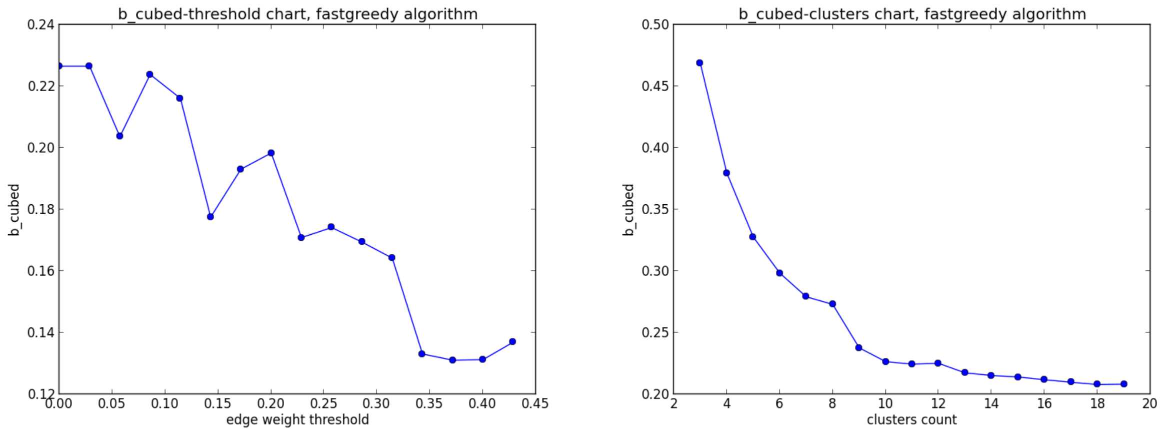


Figure 5.7: The B-Cubed results for the detected communities of the aggregation of Links and Hashtags dimensions. Case study A

The results for the Mentions similarity dimension can be find below:

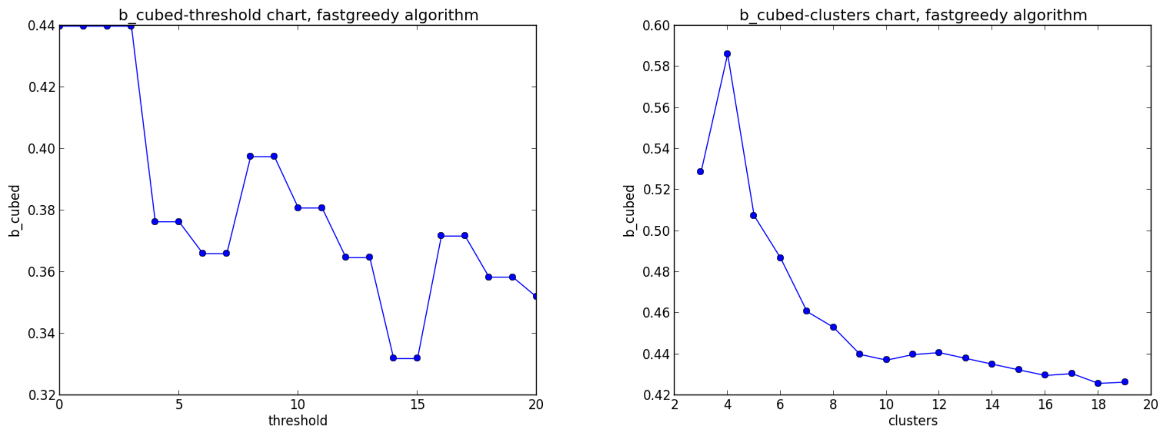


Figure 5.8: The B-Cubed results for the detected communities of the Mentions similarity dimension. Case study A

The Mentions similarity dimension performed well on its own. Aggregating it with the other similarity dimensions didn't perform well, as the number of Mentions in the MP dataset was big, so the normalisation process degraded the over all performance. More details and charts explaining the issue at the appendix.

Interaction dimensions:

The interaction dimensions didn't perform well individually, as the number of edges per dimension wasn't big enough in comparison to other dimensions. A good example regarding this issue is for the Retweets interaction dimension. The charts below show the low B-Cubed values for the edge-weight threshold parameter. The number of clusters parameter is also performing very bad, as the community detection algorithm only accepted clusters count of above 16, which is an indication of very bad network structure.

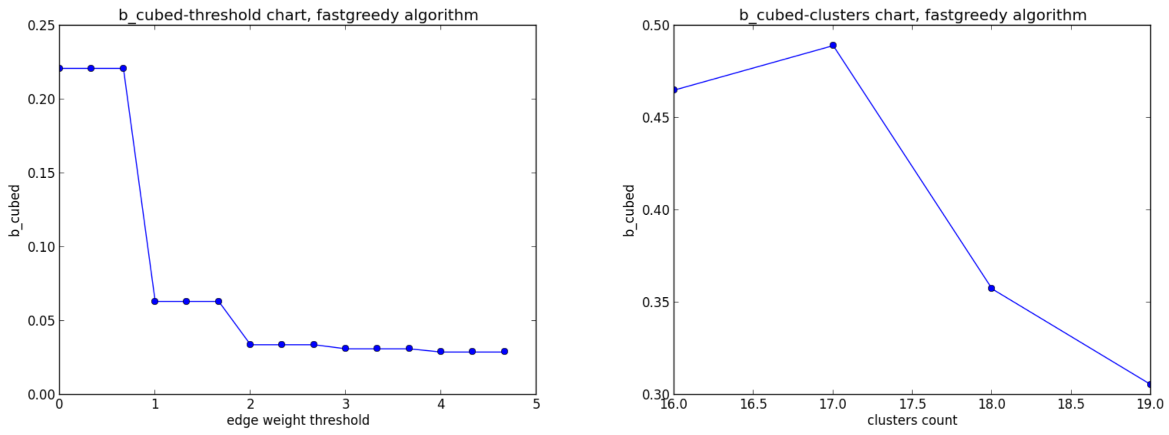


Figure 5.9: The B-Cubed results for the detected communities of the Retweets interaction dimension. Case study A

The normalised aggregated dimension for all three interaction dimensions (Retweets, Replies and Mentions interaction) showed better results, as seen below:

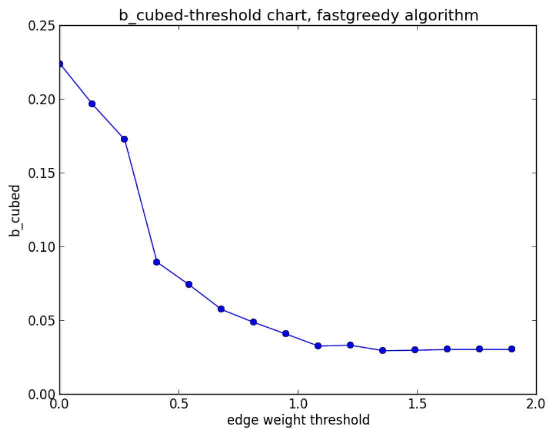
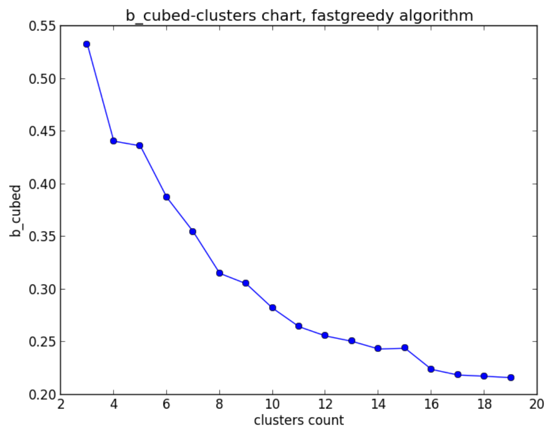


Figure 5.10: The B-Cubed results for the detected communities of the aggregated interaction dimensions. Case study A

A very important discovery for the interaction dimensions was to observe that the Replies dimension individually provided very poor results, as seen at Figure 5.11 below. Probably for the nature of when people use the "Reply" option on Social Media. So another experiment for the aggregated interaction dimensions, without the Replies dimension, was operated with the results shown on Figure 5.12 below. The resulting charts show better results than in the aggregated interaction dimension without removing the Replies dimension. The clusters count parameter charts clearly shows some better results, while the edge-weight threshold parameter shows a slight improvement.

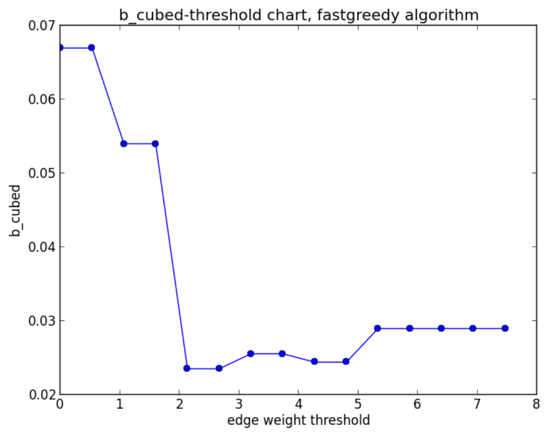
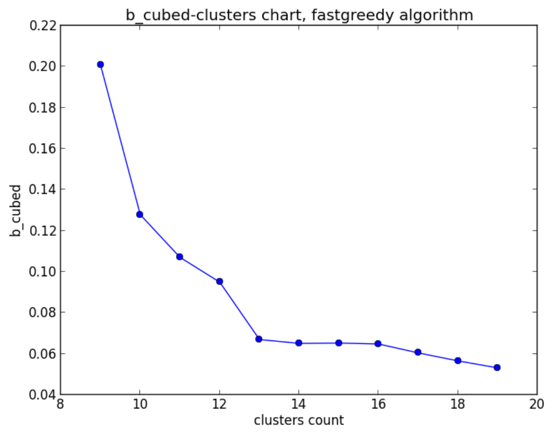


Figure 5.11: The B-Cubed results for the detected communities of the Replies interaction dimension. Case study A

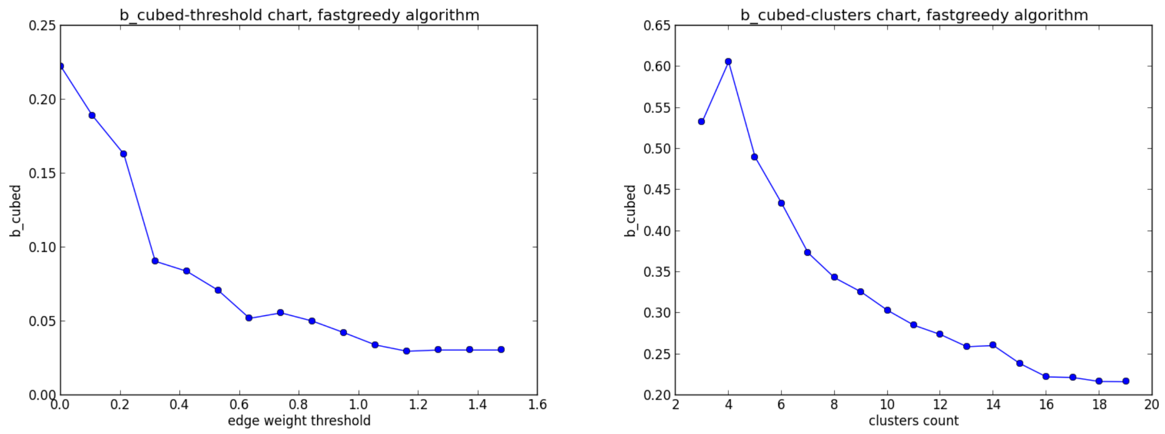


Figure 5.12: The B-Cubed results for the detected communities of the aggregated interaction dimensions, without the Replies interaction dimension. Case study A

Similarity and interaction dimensions

Finally, aggregating all the dimensions from both similarity and dimensions, excluding the Replies dimension, performed moderately in comparison to the individual or aggregated dimensions explained before. The results are shown at the Figure below.

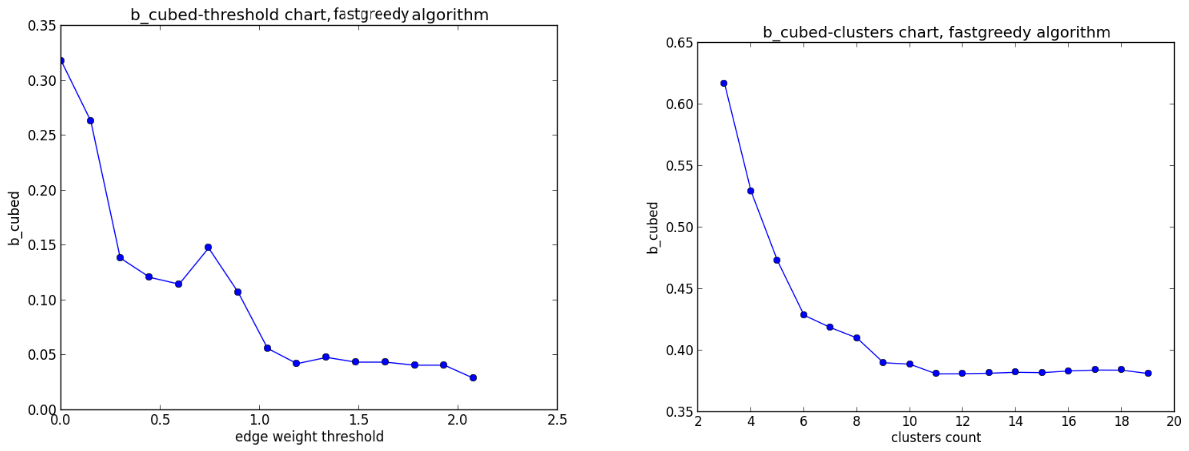


Figure 5.13: The B-Cubed results for the detected communities of aggregating all dimensions excluding the Replies dimension. Case study A

It is observed that the clusters count chart shows better results than most of the previous charts, the edge-weight threshold chart shows some medium performance in comparison to all the previous charts. Choosing the right aggregation of charts depends on the depth that the researcher would like to pursue for the problem. The all aggregated dimensions give moderate performance, while going into separate dimensions individual or aggregated might provide some better results.

Conclusions

- Some dimensions, that share some similar characteristics, might perform better if aggregated.
- For similarity dimensions, the Mentions dimension performs better alone, as the number of Mentions in the system was big in comparison to the others. The Hashtags and Links dimensions performs better when aggregated.
- For the interaction dimensions, the Replies dimension performs very poor in general, and it was decided to eliminate it from the analysis process. The reason might be the fact that users usually Reply to other users when they have disagreement regarding a certain issue or topic (people from other communities), rather than Replying to people who share the user's points of view (people of the same communities).
- The normalised aggregation of all interaction dimensions performs much better than the individual dimensions. Probably because the number of edges per dimension was low, so aggregating the efforts gave better results. The normalised aggregation while eliminating the Replies dimension gives even better results.
- The aggregation of all dimensions, excluding the Replies dimension, will give acceptable results in terms of performance. The normalisation procedure for the dimenions' matrices for aggregation might be problematic for some dimensions based on the individual dimension's characteristics. The decision to go for the all aggregated dimensions versus individual or some aggregated dimensions will be based on the depth of analysis the researcher might want to pursue. But in general, the aggregation should provide acceptable results.

5.4.7 Experiment 4: Removing the Most-Common-Entities

Objectives

- To study the effect of removing the most common entities from the similarity dimensions, Hashtags or Mentions for example, over the overall system performance

Approach

Like the "stop words" effect in the NLP, some Hashtags or Mentions might be common and general amongst the different detected clusters. Examples of these entities include #fb, #BBC, #ff ... etc. The inclusion of these entities would affect the performance of the similarity dimensions, as these words are of general relevance to all different communities that might be available in the dataset.

The approach to to study the effect of these entities over the performance would be by removing them, operating the community detection procedures again, and comparing the evaluation metrics with the metrics obtained of the detected communities without removing the entities.

To detect these entities:

1. Order the available entities (Hashtags or Mentions) by the frequency of which they were used in the dataset, in descending order.

2. From the top entities in the ordering, identify the entities that would be of general relevance to everybody. Such entities includes #fb, #Twitter, #BBC...

It is important to note that only entities of general relevance of the ordered entities are removed. That is, if for example the Hashtag #partyofwales was the most common Hashtag, it shouldn't be removed as it is not of general relevance to all the MPs.

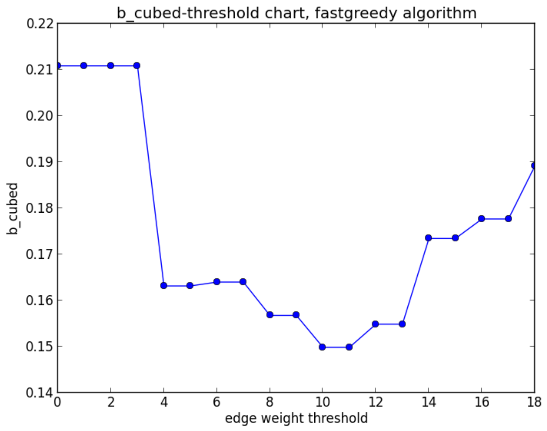
Observations and Analysis

The charts below represent the Hashtags similarity dimension before removing the most common entities, at Figure 5.14. And the Hashtags dimension after removing the most common entities, at Figure 5.15. The most common entities that were removed for the Hashtags dimension are shown at the table below. Again, these are the Hashtags of high frequency, and of general relevance. The Hashtag #feile25 was one of the most common in the MPs dataset, but since that it isn't of general relevance relativity (compared to #fb for example), it wasn't removed.

Hashtag	***Frequency of usage***
#ff	71
#nhs	36
#fb	22
#newsnight	16

Table 5.2: The most common Hashtags removed from the Hashtags similarity dimension. Case study A

The charts of the dimension after removing the entities show an improvement on the overall performance of the system, indicating that the removal of such entities should have a positive impact on the system's performance.



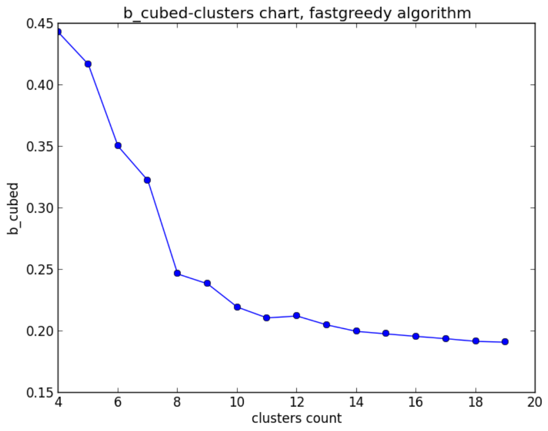


Figure 5.14: The B-Cubed results for the detected communities of the Hashtags similarity dimension, before removing the most common entities. Case study A

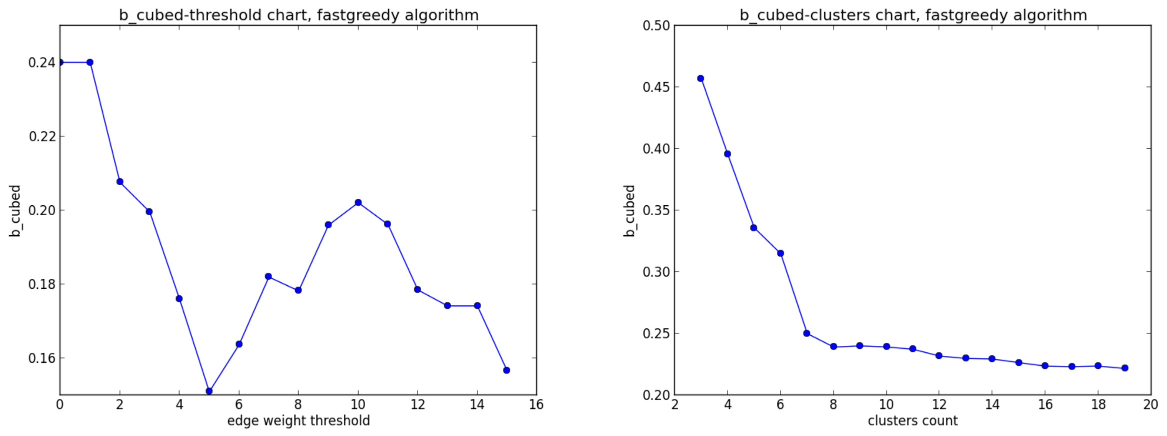


Figure 5.15: The B-Cubed results for the detected communities of the Hashtags similarity dimension, after removing the most common entities. Case study A

Conclusions

- Removing the most common entities of the similarity dimensions enhances the overall system performance.

5.5 Unsupervised Evaluation Approach

5.5.1 Overview

The previous section provided a deep and thorough analysis and evaluation of the community detection problem, given the ground truth information to compare the evaluation results against. However, real life problems generally don't offer the luxury of ground truth information or labeled data to work with. Having the ground truth information in this context contradicts with the main motivation of the community detection procedures in the first place. In this section, I will try to illustrate some analysis procedures for the community detection problem without having any ground truth data.

Such unsupervised evaluation would be scarce by nature, as there aren't much information to evaluate upon. This section will try to provide some insights for the problem at hand, with more detailed and elaborate analysis left for future work.

5.5.2 Experimental Setup

Objectives

- Trying to obtain heuristics for the available parameters, affecting the accuracy of the overall system performance.

- Obtaining a general approach to find the best values for the parameters.

- Reflecting the resulting analysis upon the results of the supervised evaluation performed before.

Dataset

The dataset for this study will be the Members of the Parliament dataset described previously. Refer to the dataset section for more comprehensive information.

Parameters

1. Number of clusters.
2. Edge weight threshold.

Algorithms

Fastgreedy community detection algorithm, from the igraph package.

5.5.3 Methodology

The main methodology for the unsupervised evaluation will be based upon heuristics. The process of obtaining the proper evaluation heuristics entails careful analysis and study of the parameters that alter the overall performance or quality of the problem at hand. The ”good” parameter values are then inferred by judgment over these heuristics.

The heuristics that will be used here are based upon the users and entities (Hashtags here) distribution. The effects of adjusting the parameters over this distribution are then studied to obtain an approximate judgment.

5.5.4 Observations and Analysis

The Threshold Parameter:

The threshold parameter is directly connected with the noise levels within the detected communities. Having a threshold, or increasing the threshold, would help eliminate or reduce such noise. However, increasing the threshold above acceptable margins might also result in losing non-noisy important edges and connections within the the system's structure. Obtaining this threshold value that establishes the balance between reducing the noise and not losing information is such an important task for a proper analysis.

Experimenting with the community detection algorithms, while adjusting the threshold, provided vital observations regarding the system behaviour. The charts at the Figure (5.16) below, from the Hashtags community detection, describe the effect of changing the threshold parameter over the number of users and Hashtags for each detected community. The number of clusters for all the charts was unassigned as input and left for the community detection algorithm (*Fastgreedy*) to obtain the optimal structure, to isolate the parameters and study the edge-weight threshold parameter individually.

Each of the charts represents the number of different users and Hashtags available at each community partition. The community labeled ”unassigned” indicates the users and Hashtags that haven't been assigned to a particular community. This might be the result of:

- Having no Hashtags affiliated with these users. In the Hashtags similarity case, 86 users have no Hashtags.

- Having no shared Hashtags with any of the other users in the dataset, and this is the reasons for the unassigned Hashtags.

- Or having a small number of shared Hashtags that get removed when increasing the edge-weight threshold.

The goal of the charts is to observe what happens to this distribution when we adjust the edge-weight threshold parameter on a range of values. The first chart represents the resulting distribution without applying any threshold filter. The consequent charts represent the new distribution after increasing the threshold by a step value of 1 in this case (steps value depends on the nature of the data, here chosen to be 1).

Analysing the charts, the number of unassigned users continues at the same levels until the threshold value 2, which happens to be the minimum value within the dimension's matrix, where we notice an increase of about 35 unassigned users. Looking at the next few charts we observe some movements for the users amongst the communities, especially the communities 4,5 and 6. These minor shifts are the result of reducing the noise levels while assigning the clusters, which might result in users having totally different clustering.

So the first round of increasing the unassigned users began at threshold value 2, which is the minimum value in the matrix as illustrated. This could mean that the first set of "noisy" edges was removed here. The next few charts show that the unassigned users kept on increasing at a slow rate. For all the charts after threshold value of 2, the number of unassigned users and Hashtags keeps on increasing. This could mean eliminating both noisy and correct edges! The best approach here is to try to find a balance, a threshold value that would eliminate noisy edges, but not lose to many correct edges. Such "balance" threshold value might be around 2, 3, or 4. As the system starts losing lots of users and Hashtags afterwards, which is more likely to be correct edges rather than noisy ones, at this stage.

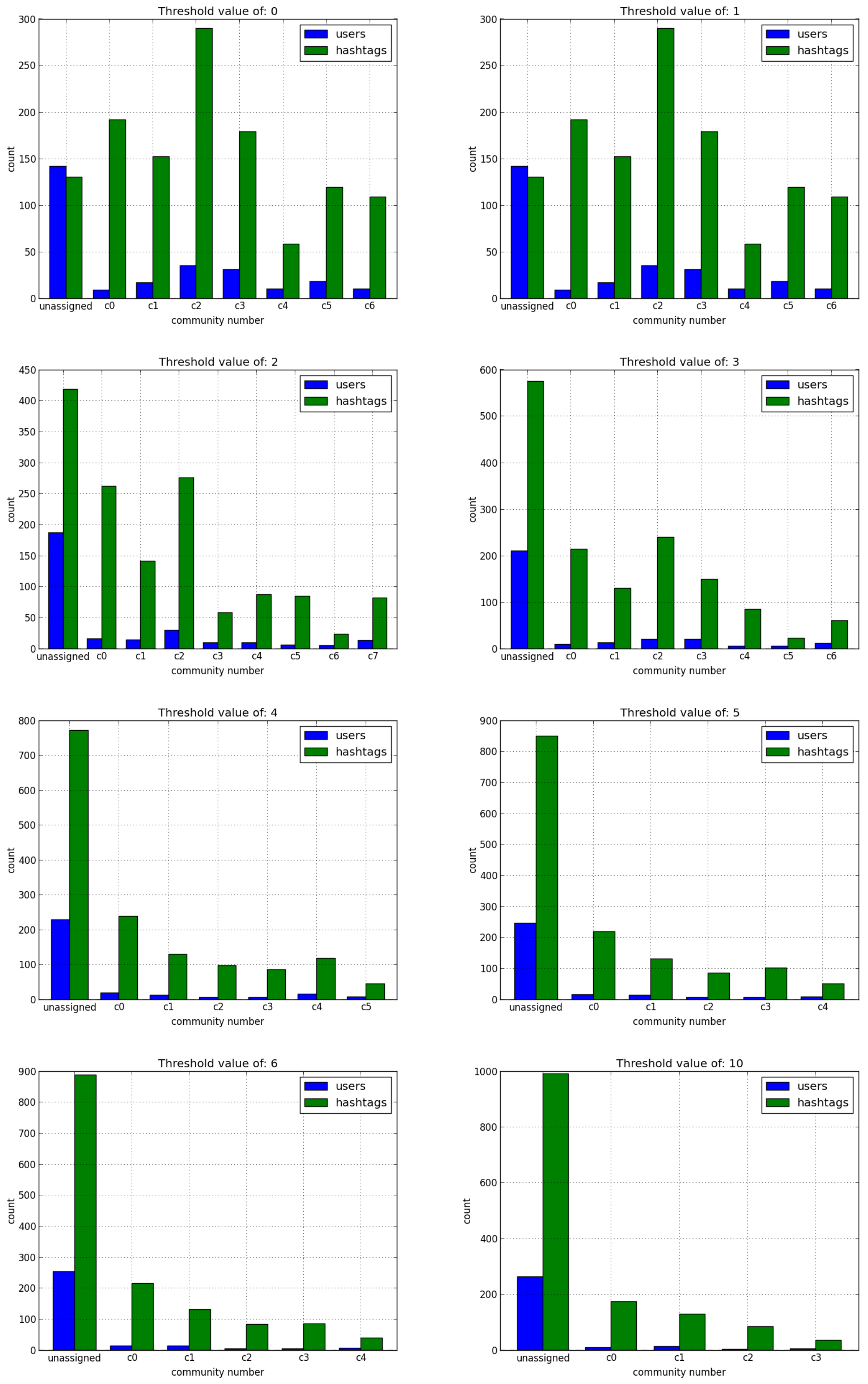


Figure 5.16: The effect of the edge-weight threshold parameter value over the number of users and Hashtags, Fastgreedy algorithm. Case study A

Comparison with the Ground Truth

The procedure Mentioned above to obtain the best parameter values were based on unsupervised learning and analysis, which is prone to errors, lack of information, bias and bad judgment! A good approach in our case is to compare the results of our analysis against the results of the supervised evaluation process, with the already available political parties affiliation for each user. The following charts outline the corresponding Purity (and Purity box plots to check the deviations) and B-Cubed measurements for the edge-weight threshold parameter.

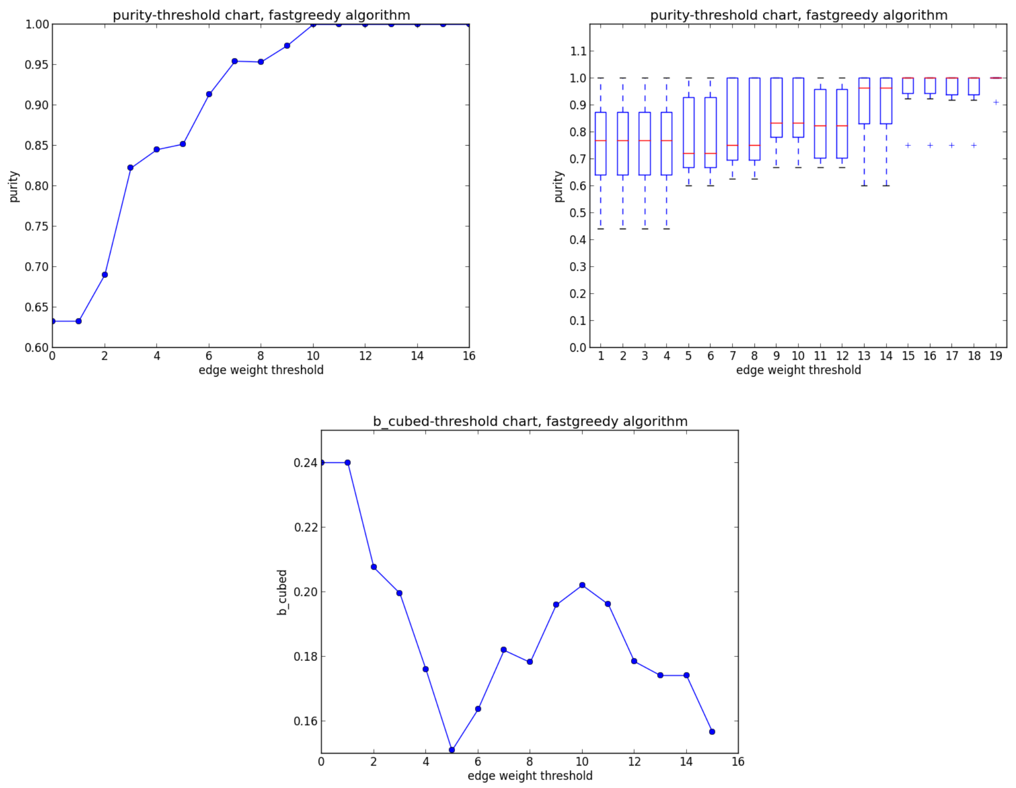


Figure 5.17: The edge-weight threshold parameter's metrics results for the detected communities of the Hashtags similarity dimension. Case study A

The graph clearly indicate that the best edge-weight threshold value should be 3 or 4; as such threshold value would provide relatively good Purity, with good B-Cubed value. This threshold value matches the previous approximation based on the unsupervised analysis.

The Number of Clusters Parameter:

Before diving into analysing the clusters count, it is vital to outline again the way community detection algorithms generally function with regards to this issue. Many of igraph's community detection algorithms accept an input specifying the number of returned communities. This is performed by cutting the resulting *Dendrogram* graph at certain levels, as illustrated at the supervised evaluation section's methodology, Section 5.4.3. However, this count is considered as the maximum value for the clusters count, rather than a fixed value. The algorithm will then decide the optimal clusters count given this maximum. For example, if the input clusters count was 10, but the algorithm found the optimal clusters at a count of 7, the resulting number of communities will be 7.

However, if we didn't specify any value for the maximum clusters count, the algorithm will return the overall optimal number of clusters. This is dangerous as the overall optimal clusters count within the scope of social content might be large, which is bad for our analysis. To illustrate the proposed methodology, consider the table below with the input clusters count to the community detection algorithm, with increasingly adjusting the clusters count. And the actual returned clusters given each input.

Input count	***Actual returned count***
3	1
4	2
5	3
6	4
7	5
8	6
9	6
10	6
11	6
12	6
13	6
14	6
15	6
16	6
17	8
18	8

Table 5.3: Hashtags clusters count input, and actual count output. Fastgreedy algorithm. Case study A

It can be observed from the Table 5.3 that increasing the input clusters count, would eventually increase the actual output count. The reason being that the community detection algorithm would find a more solid structure representation at that level to cut the *Dendrogram* at. However, the output clusters count starts to stabilise momentarily at around 6, for the input clusters count of 8-16, then it continues to rise again. We could perceive this issue as a solid structure for the network at hand given the available edges.

This observation provides a methodology to set the clusters count when we don't have ground truth information that enables us to calculate the B-Cubed and Purity values. The approach is to experiment the community detection algorithm with a range of input clusters count, and observe when the actual resulting communities count becomes stable for multiple inputs. This actual communities count would represent the best value of the clusters count in the system. The users distribution chart below for the clusters count parameter also supports this approach. The users distribution stabilizes at the stable cluster count of 8-16. However, it can be observed that the number of unassigned users tends to increase a bit as we increase the input clusters count away of 8. This suggests that the best clusters count should be around 8.

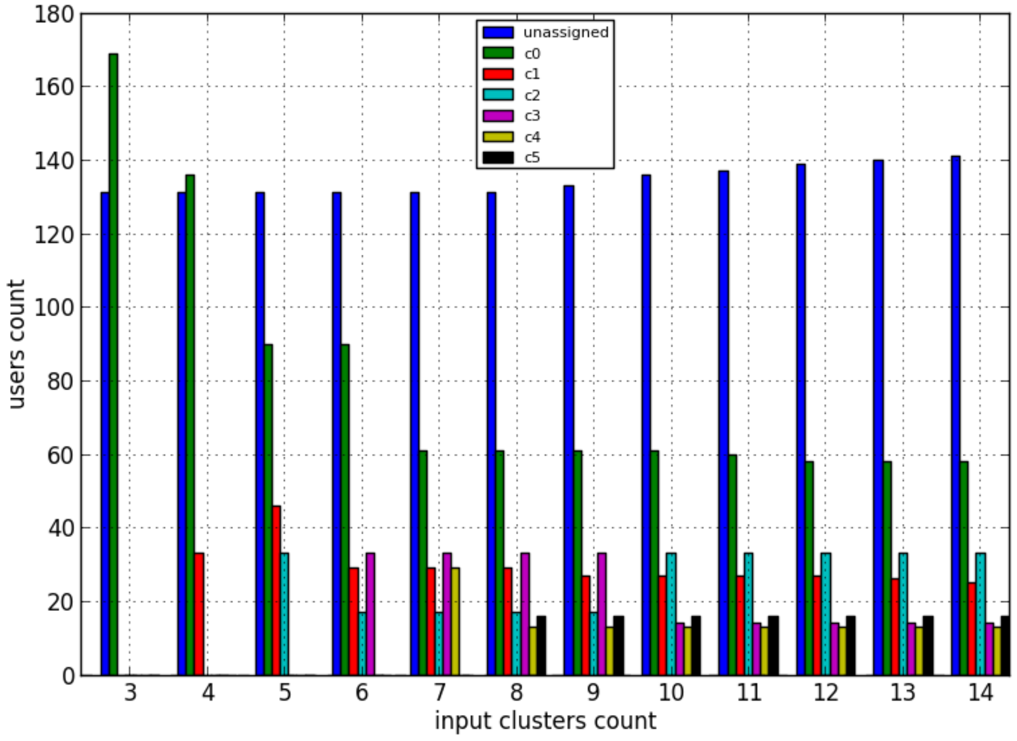


Figure 5.18: Users distribution plot over the detected communities, when adjusting the clusters count parameter. Hashtags dimension, Fastgreedy algorithm. Case study A

Comparison with the Ground Truth

Again as in the threshold parameter, the supervised evaluation results are shown at Figure 5.19 below. The charts represent the Purity (and Purity box plots to check the deviations) and B-Cubed measurements for clusters count parameter. We could detect from the charts that the best clusters value for the Hashtags dimension should be around 6 or 7, which would provide relatively good Purity and B-Cubed values. The corresponding actual clusters count for both 6 or 7 is 6.

The best resulting clusters count for the unsupervised approach was around 8, which also provided 6 actual clusters. The result doesn't exactly match the supervised results, but it's close enough. As the entire unsupervised evaluation approach is approximate and error prone as explained before.

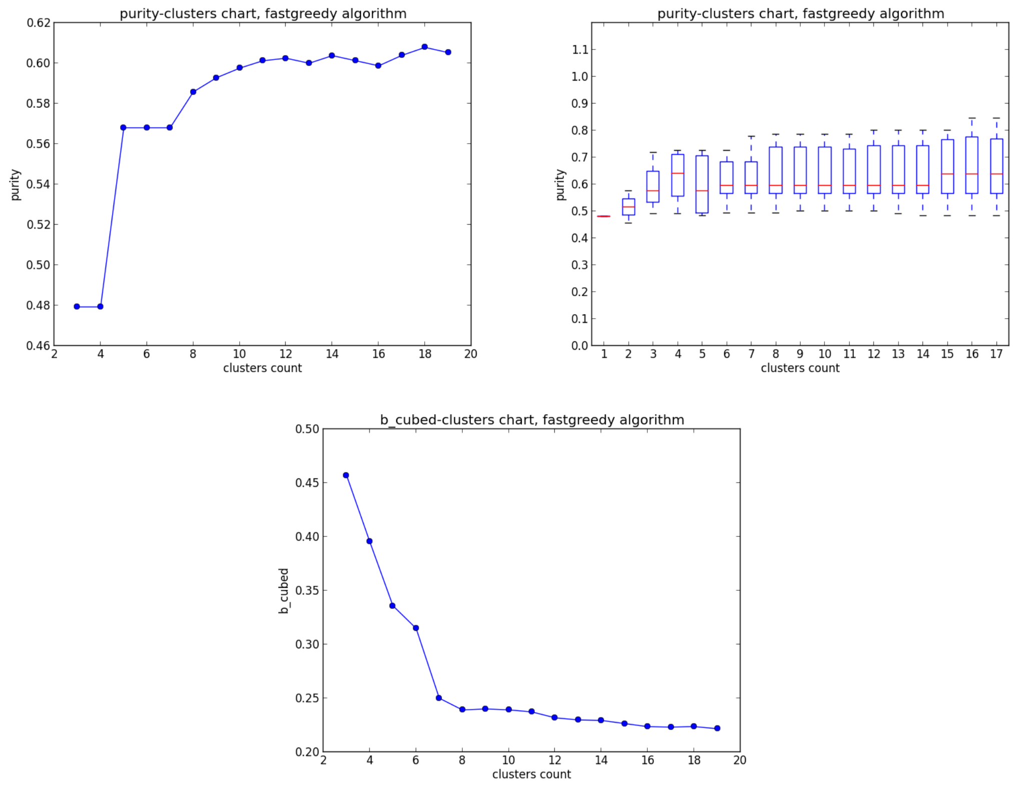


Figure 5.19: The clusters count parameter's metrics results for the detected communities of the Hashtags similarity dimension. Case study A

The previous arguments and experiments support the methodology followed to infer the appropriate parameter values without having ground truth information. The same procedure was applied to the "Mentions dimension" and other dimensions and showed similar results.

5.6 Summary

The goal of this chapter was to provide a thorough analysis and evaluation of the factors that affect the overall performance of the community detection procedure. The analysis was divided into separate experiments, each with specific goals. Each of these experiments provided a comprehensive description of its goals and approaches, then the conclusions drawn of the results were provided.

This section is dedicated to provide a quick recap of the conclusions of these experiments, while stressing that a much more comprehensive analysis is provided at the corresponding parts of the chapter.

Parameters and Metrics Analysis

The parameters of edge-weight threshold, and clusters count were analysed here. And the metrics of Purity, F-Score and B-Cubed were used for this analysis. The drawn conclusions were:

- Purity, or precision, of the system tends to increase as we increase the number of clusters and the edge-weight threshold. As the number of users per cluster drops, and the more "pure" users remain.
- The F-Score and B-Cubed values tend to drop as we increase the number of clusters and the edge-weight threshold. The reason being the increase in clusters count and the fewer users per cluster decreases the recall value.
- Purity and F-Score evaluation metrics provide good information about the system behaviour, however, B-Cubed metric is the best metric for comparisons in general, as it's an item-based evaluation approach, and it doesn't average individual values. B-Cubed was used for most of the consequent analysis.

Similar Communities Aggregation

Aggregating similar communities will greatly affect the system performance, as the reduced clusters count after aggregation will increase the B-Cubed metric, by increasing the recall value. And the fact that the communities to be aggregated are similar, the precision remains equal.

It must be noted here, however, that this aggregation is possible only in the cases where ground truth information (political affiliations in our case) is accessible, aggregating clusters using unsupervised approaches will be left for future work.

Dimensions Analysis

Each of the available connection dimensions were studied individually. The conclusions were as follows:

- For similarity dimensions, the Mentions dimension performs better individually, as the number of Mentions in the system was big in comparison to the others. The Hashtags and Links dimensions performs better when aggregated.
- For the interaction dimensions, the Replies dimension performs very poor in general, and it was decided to eliminate it from the analysis process. The reasons being the nature of interaction by Replies, as it's limited to the users involved only, which might result in outlier conversations. Also the sociological nature of Replies in general, where it's most used to counter an opposite opinion rather than discussing similar ones.
- The normalised aggregation of all interaction dimensions performs much better than the individual dimensions. Probably because the number of edges per dimension was low, so aggregating the efforts gave better results. The normalised aggregation while eliminating the Replies dimension gives even better results.
- The aggregation of all dimensions, excluding the Replies dimension, will give acceptable results in terms of performance.
- The normalisation procedure for the dimenions' matrices for aggregation might be problematic for some dimensions based on the individual dimension's characteristics. The decision to go for the all aggregated dimensions versus individual or some aggregated dimensions will be based on the depth of analysis the researcher might want to pursue. But in general, the aggregation should provide acceptable results.

Most Common Entities Removal

Removing the most common entities of the similarity dimensions enhances the overall system performance. The reasoning behind this is that the inclusion of these entities would affect the performance of the similarity dimensions, as these words are of general relevance to all different communities that might be available in the dataset, rather than to certain ones. So, instead of acting as a similarity measure of individual communities, they act as noisy edges for all communities.

Unsupervised Evaluation

The unsupervised evaluation experiment showed a general approach that can be applied at the community detection procedure to obtain good parameters settings. This approach doesn't require ground truth information, as in most real life problems. The experiment then provided an evaluation for this approach using the ground truth analysis results, and the evaluation showed good performance.

Chapter 6

Incorporating the Communities into Sentiment Analysis

6.1 Overview

As illustrated thoroughly at the literature review of this thesis, the sentiment analysis problem in Social Media suffers of many serious challenges. Some of the main challenges includes the lack of context regarding the analysed text. Lack of context poses many problems for proper analysis, or in cases of sarcasm where most traditional sentiment analysis tools fail miserably. Incorporating the communities with the sentiment analysis should provide this context and background for the sentiment analysis, and should enhance the overall performance, as shall we investigate here.

The community information detected after operating the community detection procedures represent the structural characteristics of the network at hand. This chapter is concerned with how to make use of such information to enhance the accuracy of the sentiment analysis. The main challenge would be the way to incorporate the detected communities' information with the sentiment analysis process, aiming at enhancing the overall sentiment accuracy.

The problem of sentiment analysis within the context of Social Media, and Twitter particularly, can be viewed as a model that is affected by the feature-sets of:

- Textual features of the Tweet itself, obtained by means of lexical analysis and natural language processing.
- Structural features, extracted of the position or location of the user that issued the Tweet within the social network. Community detection algorithms, with overlapping communities, should provide the evidence for such features.

The main issue here would be the approach of which these two different classes of feature-sets can be combined.

6.2 Sentiment Analysis: Supervised and Unsupervised Learning

As discussed at the theoretical background of the project, the sentiment analysis problem can generally be approached through supervised and unsupervised learning methods. The supervised learning methods require big enough set of labeled data to train the classifier. Such labeled data at real life problems might be quite tedious to obtain and analyse. The other approach would be the unsupervised learning methods, analysing the lexical and linguistic features of the text at hand. Unsupervised methods generally tend to have lower accuracy than supervised approaches, as the analysis considers that linguistic features only rather than the linguistic features given the overall context. The same sentence could have totally different sentiments in different contexts or for different users.

A sample of good unsupervised learning tools for sentiment analysis is the Sentistrength package. As illustrated at the tools section before, Sentistrength measures the strength of the positive and negative sentiments available in the text, based purely upon linguistic and lexical features, like the POS and verbs/adjectives characteristics. In addition to having a dictionary of positive and negative adjectives and their respective sentiment value. Sentistrength also can translate abbreviations and emoticons widely available in Social Media means into proper English, to incorporate them in the analysis process[23].

But again as expected, Sentistrength performs very badly at some situations; like sarcasm for example. Sentistrength lacks the context and the users information affiliated with the text. Having a prior information about the writer of the text, or the overall environment, helps making a better and more informed analysis for the text at hand. Here comes the importance of incorporating the detected communities' information at the sentiment analysis process. The communities add the context or prior knowledge required to make a more informed analysis.

6.3 Sentiment Analysis: Proposed Methodology

So the problem of unsupervised learning methods for sentiment analysis is the lack of users and context information. A possible approach is to simply feed such missing information into the unsupervised process! If we managed to provide the context information for each individual text, Tweets in Twitter dataset for example, the unsupervised process becomes more aware about the text and its background, increasing the judgment accuracy.

A good source for obtaining the users' and context information is their social affiliation. Based on the sociological principles of *Influence* and *Homophily* discussed before, people who are part of a community tend to share the same sentiments regarding the topics of matter for the communities. So the fact that a user belongs to a certain community involves a big indication of what his/her sentiment regarding a certain issue might be like.

There are now two different feature-sets affecting the polarity of the user's sentiment; the linguistic and structural features. One way to look at such system would be to perceive both of these feature-sets as probabilities. The user's sentiment depends upon the probability of the sentiment as of its linguistic features, and the probability of the sentiment as of the community membership. The overall aggregate probability distribution of each text can then be inferred using a form of Bayesian Inference Model. The probabilities extracted of both of these feature-sets would represent the Prior and Likelihood information of this Bayesian model.

- Prior probability: The probability of the sentiment's value (positive or negative), as extracted from the linguistic sentiment extraction process. Sentistrength will provide this piece of information in our case.
- Likelihood probability: Which is the overall sentiment's value probability (positive or negative) of the community that contains the user of which the Tweet is issued, given the topic.

The Bayesian Inference Model can then be used to obtain the overall probability, given the Prior and Likelihood.

6.4 Bayesian Inference Model

Let S denote the sentiment of the user, and$C$ be the overall sentiment of the community**,** Bayes Inference formula would be:

$$P(S|C) = \frac{P(S)P(C|S)}{P(C)}$$

Where:

$P(S|C)$ is the probability of the Tweet to have sentiment S, given that the user issuing the Tweet belongs to a community having an overall sentiment C.

$P(S)$ is the prior probability of the Tweet to have sentiment value S, which is the result of the Sentistrength analysis.

$P(C|S)$ is the likelihood probability, the probability of the community to have sentiment C, given the sentiments of its belonging users.

$P(C)$ is the normalizing constant. It is calculated by summing the values of $P(C|S)$ over all the possible values of S.[13]

Calculating the likelihood probability $P(C|S)$:

According to the Bayesian Inference Model, the likelihood of a set of parameter values, ϑ, given outcomes x, is equal to the probability of those observed outcomes given those parameter values, that is $L(x|\theta) = P(\theta|x)$. In our case this model can be viewed as the users tendency to convey a certain sentiment as a result of belonging to a certain community, based on the sociology principles of *Influence* and *Homophily.* So the parameter values will be the community labels, and the outcomes are the sentiments (positive or negative).

To calculate the likelihood function for a given dataset with discrete probability distribution, let X be a random variable with a discrete probability distribution p depending on a parameter ϑ. Then the function

$$L(\theta|x) = p_\theta(x) = P_\theta(X = x)$$

To calculate the conditional probability $P(C|S)$ the following formula can be used:

$$P(c_i|s_i) = \frac{occurrences\,of\,[c_{i,}s_i]}{occurrences\,of\,[s_i]}$$

So for example, to calculate $P(c_1|negative)$**,** the formula would be:

$$P(c_1|negative) = \frac{occurrences\,of\,[c_{1,}negative]}{occurrences\,of\,[negativve]}$$

Example:

	probability
$P(c1\|neg)$	0.8
$P(c1\|pos)$	0.4
$P(c2\|neg)$	0.3
$P(c2\|pos)$	0.7

Table 6.1: The likelihood probabilities of the Bayesian Model example, calculated from the dataset and community information.

Assuming we have the communities information and the sentiment polarities for each community's list of users, we can obtain the table below outlining the likelihood probabilities distribution amongst the possible community values (assuming we have 2 communities only, c1 and c2) and sentiment values (neg and pos).

The normalizing constants for the table above:

Normalizing constant for c1, P(c1)= P(c1|neg) + P(c1|pos) = 0.8 + 0.4 = 1.2 (notice that the normalizing constant value doesn't have to sum to 1)

Normalizing constant for c2, P(c2)= P(c1|neg) + P(c1|pos) = 0.3 + 0.7 = 1

The Bayes formula can then be applied to calculate the P(S|C) values, e.g the P(neg|c1) value can be calculated as follows:

$P(neg|c1)=\frac{P(neg)P(c1|neg)}{P(c1)}= \frac{0.5*0.8}{1.2}= 0.33$

Applying the formula for the other three items:

text	*community*	$P(neg)$	$P(pos)$	$P(neg\|C)$	$P(pos\|C)$	*sentiment*
David Cameron's policies are just brilliant!	c1	0.5	0.7	0.33	0.23	negative
David Cameron's policies are just brilliant!	c2	0.5	0.7	0.15	0.49	positive

Table 6.2: Results of applying the Bayesian Inference formula, sentiment analysis example

From the table above you could observe that the resulting negative probability was higher than the positive probability for the first row ($P(neg|c1) > P(pos|c1)$), so the overall sentiment will be negative (even though the overall probability was positive before applying the formula). For the second row the overall probability remained positive, which coincides with the sentiment of the community.

It can be observed from the example above that the overall sentiment got modified relative to the sentiment and sentiment percentage of the containing community. So even if the output of the lexical analysis of the text didn't reflect the actual semantics of the text, the inclusion of the community information was able to alter the sentiment polarity. And in this case, it was actually able to toggle the overall sentiment value.

The Bayesian Inference Model is the core concept of the the Naive Bayesian Classifier, which is a statistical classifier that predicts class membership, with certain probability. The Naive Bayesian Classifier works on the basis of features that contribute to the posterior probability of class membership. It assumes feature-independence; that is, each feature or attribute contributes to the class membership's probability independently.

In the project's context, the Bayesian Classifier is used to predict the sentiment labeling of the Tweets, as belonging to the positive or negative classes. The textual content of the Tweet can be considered as the first features-set for the model, and the community information as an additional structural feature that contributes to the overall membership probability. The classifier is first trained over an existing labeled dataset, using which the classifier calculates the corresponding membership probability of each of the textual and structural (community information) features.

The Bayesian Classifier requires a training dataset to calculate the features-set class membership probabilities, as explained before, and a testing dataset to measure the accuracy of the classification upon. The diagram below outlines the general flow of the classification process.

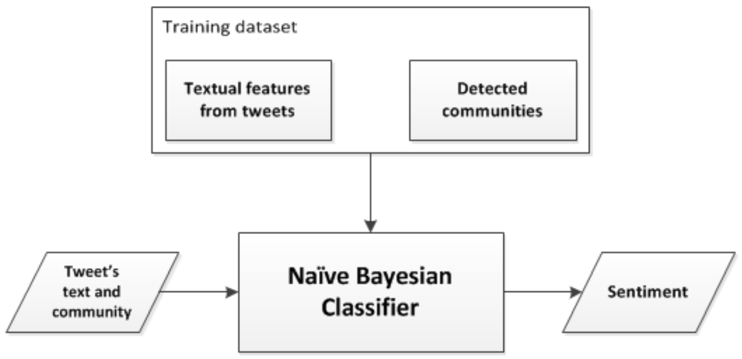

Training dataset
Textual features from tweets
Detected communities
Tweet's text and community
Naïve Bayesian Classifier
Sentiment

Figure 6.1: Diagram showing the application of the Naive Bayesian Classifier for sentiment analysis

6.5 Case Study B: Members of the Parliament (MPs) Dataset, Inbound and Outbound

6.5.1 Objectives

- Making use and applying the analysis obtained at the community detection evaluation chapter in the community detection problem of bigger datasets.
- Applying the Bayesian Inference Model to incorporate the detected communities' information to the sentiment analysis process.
- Measuring the enhancement in the sentiment analysis accuracy resulting of incorporating the detected communities' information.
- Introducing a general classifier for community detection.
- Measuring the sentiment analysis accuracy as a result of incorporating the classified communities from the classifier above.

6.5.2 Dataset Collection and Statistics

The dataset was collected by crawling the official Twitter accounts related with each member. The process progressed as follows:

1. Lists for all UK MPs, grouped by political affiliation, were obtained from the website Tweetminster.co.uk. Tweetminster.co.uk is a website that collects and analyses media content related to the UK politics.
2. The crawler should be configured in a way to collect both inbound and outbound Tweets, that is, Tweets issued by the MPs themselves, or Tweets issued by the general public that involves one of the MPs (via Mentions or Retweets for example).
3. The Twitter crawling is now operated for each member of each political party. The collected Tweets are filtered through the collecting API, Trendak, to include the Tweets with political content only. This enhances our analysis as we will evaluate the results later on relative to the political affiliations of the MPs.

The following numbers represent the statistics of the data items collected for the MPs, inbound and outbound, dataset:

Item	***Count***
dates	29/7 - 26/8 2013
users	27422
Tweets	68182
unique Hashtags	4527
total Hashtags	20850
unique Mentions	12075
total Mentions	131021
unique Links	5125
total Links	7488
Replies	15725
Retweets	37664

Table 6.3: Statistics related to the MPs inbound and outbound dataset. Case study B

Eliminating the one-time-activity Tweets

The dataset as is contains many Tweets belonging to users with a one-time-activity, that is, users who only participate or appear once in the dataset. Such users aren't adding up anything to the community detection procedures, but useless overhead. A suggested action would be to remove such Tweets and users. The table below shows the statistics of the dataset after removing these Tweets. A noticeable drop in the users count, of about 18,000 users! Including all of these users in the analysis process would have caused useless system overhead, and potential erroneous results.

Item	***Count***
dates	29/7 - 05/8 2013
users	9134
Tweets	49894
unique Hashtags	3720
total Hashtags	15190
unique Mentions	10201
total Mentions	97921
unique Links	3497
total Links	5704
Replies	12280
Retweets	23482

Table 6.4: Statistics related to the MPs Inbound and Outbound dataset, after removing the one-time-activity users. Case study B

6.5.3 Community Detection

Algorithm

The community detection evaluation chapter concluded that the *Fastgreedy* algorithm performed the best throughout the different tests, so the main community detection algorithm for this case study will be the *Fastgreedy.*

Eliminating the most common entities

The experiments in the community detection evaluation chapter, indicated noticeable performance enhancement after removing the most-common-entities from the dataset. Following the results of Mentioned experiment, the same approach has been applied for the current dataset for the Hashtags and Mentions dimensions. The removed most common Hashtags and Mentions include:

Hashtag	***Frequency***
ff	205
bbc	79
newsnight	93
uk	85
bbcnews	64

Mention	***Frequency***
bbcr4today	714
BBCNews	436
guardian	405
Telegraph	208
–	–

Table 6.5: Most common Hashtags and Mentions removed from the dataset. Case study B

Setting the community detection parameter values

First for the clusters count parameter, the Table 6.6 below outlines the resulting actual communities of the community detection process, for gradual clusters count input. The same procedure as illustrated at the community detection evaluation chapter is applied here.

Input clusters count	***Actual clusters count***
2	1
3	2
4	3
5	3
6	4
7	5
8	6
9	6
10	6
11	6
12	6
13	11

Table 6.6: Actual clusters count for each input clusters count; setting the clusters count parameter. Case study B

It is observable from the table that the actual clusters count of 6 is relatively stable for the system. So the clusters count of 6 is chosen as the preferable value.

For the edge-weight threshold parameter. The Figure 6.2 outlines the users' distribution for each threshold value for individual dimensions.

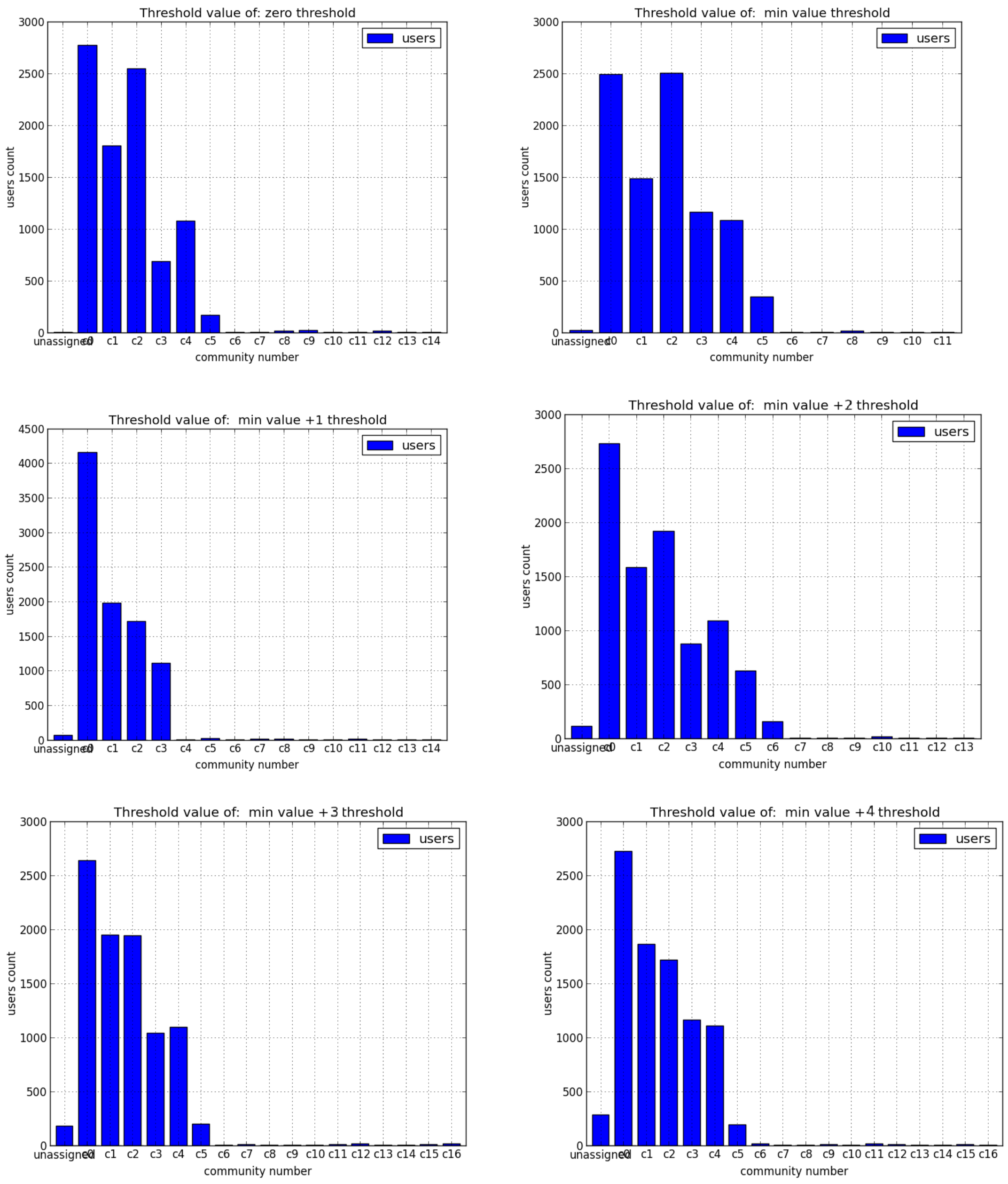


Figure 6.2: Users' distribution plot over the detected communities, corresponding to edge-weight threshold value change. Case study B

Again as illustrated at the community detection evaluation chapter, we could observe that the best performance boost occurs at a threshold value around the minimum value available at each dimension's matrix. That is, setting the threshold to around the minimum edge-weight values in the matrix associated with each dimension. For this dataset, the edge-weight threshold values were set to the minimum edge-weight value for each dimension, plus two. The Figure 6.2 shows that the minimum value +2 threshold shows good users distribution amongst the communities. Going further after this threshold value increases the unassigned users count largely.

6.5.4 Sentiment Analysis

The community detection procedure outlined before was operated upon the bulk of the main dataset of the MPs inbound and outbound. To work with sentiment analysis however, a topic based dataset has to be derived of the main dataset, as a sub-dataset, to perform the basis of the sentiment analysis procedure. The dataset chosen for this purpose was with the topic of "David Cameron", as the Prime Minister.

All Tweets of this dataset are part of the bigger dataset of the MPs, so each Tweet and user in this dataset should have community information from the MPs dataset.

"David Cameron" dataset

The dataset contains 1400 Tweets in total, and 700 different users, all share the topic of "David Cameron". All 1400 Tweets were manually labeled, with sentiment labels identified within the context of "David Cameron". The dataset was composed of 136 positive Tweets, 732 negative Tweets, and 532 neutral Tweets. The neutral Tweets will not be included at any subsequent analysis. The dataset was divided into testing dataset of 300 Tweets, and training sets of 1100 Tweets.

Bayesian Inference Model

The conditional probability of $P(C|S)$, which represents the likelihood probability, was calculated for the training dataset using the corresponding formula in the Bayesian Inference Model section.

The resulting likelihood probabilities were as follows:

P(C|S)	***Negative***	***Positive***	***Neutral***
Community 0	0.521	0.242	0.081
Community 1	0.077	0.095	0.108
Community 2	0.359	0.570	0.761
Community 3	0.009	0.030	0.018
Community 4	0.017	0.046	0.021
Community 5	0.017	0.016	0.010

Table 6.7: The Likelihood probabilities for David Cameron dataset. Case study B

The linguistic sentiment resulting from the unsupervised analysis of Sentistrength was then multiplied by the corresponding likelihood probability from the table above, then normalised. As explained at the example at the Naive Bayesian Model section.

Results

The sentiment analysis results are illustrated at the charts below. The first bar represents the results without incorporating the community information, with the sentiment results of the unsupervised learning approach tool Sentistrength. and the second bar is the sentiment result after incorporating the detected communities information. The metrics used are the Positive Precision, Negative Precision, and Accuracy. Where:

$$Positive\,Precision = \frac{True\,Positive}{True\,Positive + False\,Positive}$$

$$Negative\,Precision = \frac{True\,Negative}{True\,Negative + False\,Negative}$$

$$Accuracy = \frac{True\,Positive + True\,Negative}{total\,labels}$$

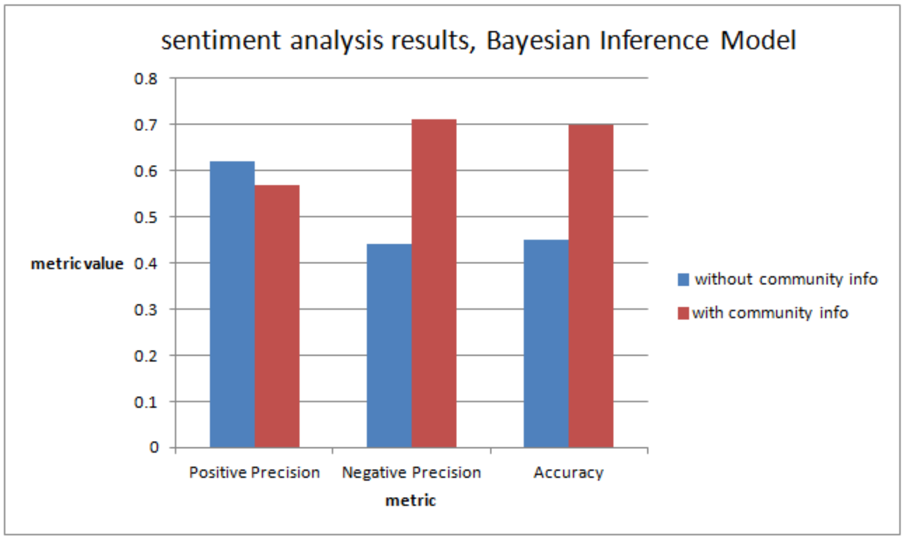


Figure 6.3: Sentiment analysis results. Results of Sentistrength (blue) vs results after incorporating the communities' information through the Bayesian Inference Model (red). Case study B

As the chart illustrate, the accuracy of the sentiment analysis process was enhanced by over 25% as a result of incorporating the communities' information within the analysis.

Conclusion

Incorporating the communities' information increases the accuracy of the sentiment analysis process using the unsupervised learning approach.

6.6 Building a General Community Classifier

The approach illustrated before operates upon the sentiment analysis process on the main condition of the users of both the training and testing data being part of the detected communities. This renders the whole process useless for the general case or general population.

Another approach would be to be able to characterise the available communities and build a general classifier, trained by the available communities' data. This classifier can then be operated upon any other users or Tweets that aren't part of the detected communities, but can be considered as a part of the general space of the available dataset.

6.6.1 Objective

- To introduce a communities classifier to be able to detect the community information for a new user to the dataset space. That is, a user that doesn't have community information available already.
- To apply this user's new community information for the sentiment analysis process.

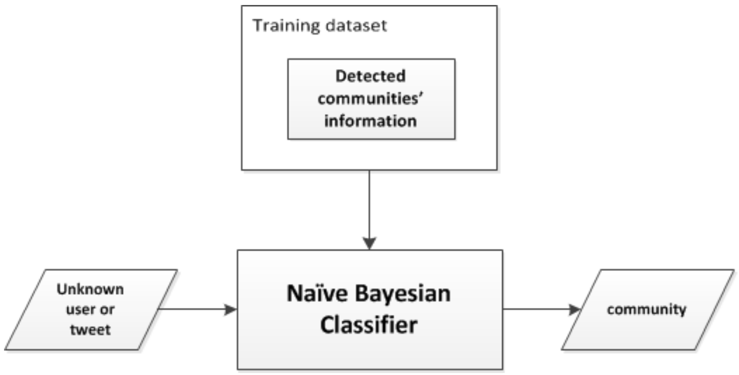


Figure 6.4: Diagram showing the application of the Bayesian classifier to classify communities. Case study B

6.6.2 Experimental Setup

Objectives

1. To train a Naive Bayesian Classifier for the users in the detected communities (training dataset), with the Hashtags, Mentions and Links as features, and the corresponding community as label.
2. To investigate the effect of the size of the dataset over the performance of the classification.
3. To investigate the effect of temporal difference between the training and testing datasets.
4. To apply this classifier to users with unidentified communities (testing dataset), within the general space of the available dataset.

Dataset and algorithms

The dataset to be used for this analysis, along with the corresponding algorithms, is the MPs inbound and outbound dataset explained at the case study section.

Methodology

1. Divide the original dataset, that has the community information, into training and testing datasets. The ratio for the datasets o be divided as 90:10, as more training data is needed to capture the big communities.

2. Train the Naive Bayesian Classifier using the training dataset. The Hashtags, Mentions and Links are considered as features for the classification, and the corresponding community information as labels.

3. Apply the generated Bayesian Classifier over the testing dataset. Again with the Hashtags, Mentions and Links as potential features.

4. Evaluate the resulting labels. The newly generated labels as the predicted_labels, and the already available community labels as the true_labels.

5. Repeat the previous steps for different combinations of the features set (Hashtags, Mentions and Links), and analyse which features provide better results.

Parameters

- The size of the dataset.

- The features of the classifier.

6.6.3 Experiment 1: The Effect of the Training Dataset Size

This experiment is directly related to the 2nd objective of this analysis. The size of the training dataset that includes the detected communities information might be small, or large, depending on the dataset space. This experiment studies the effect of the data size over the classifiers performance.

Approach

The size of the overall dataset is increased in a given range, then the testing and training datasets were derived of the new overall dataset. The steps of the general approach Mentioned before for building the classifier are then operated for each dataset size.

The tests were operated for the dimensions of Hashtags, Mentions, and an aggregate dimension of both Hashtags and Mentions.

Results and analysis

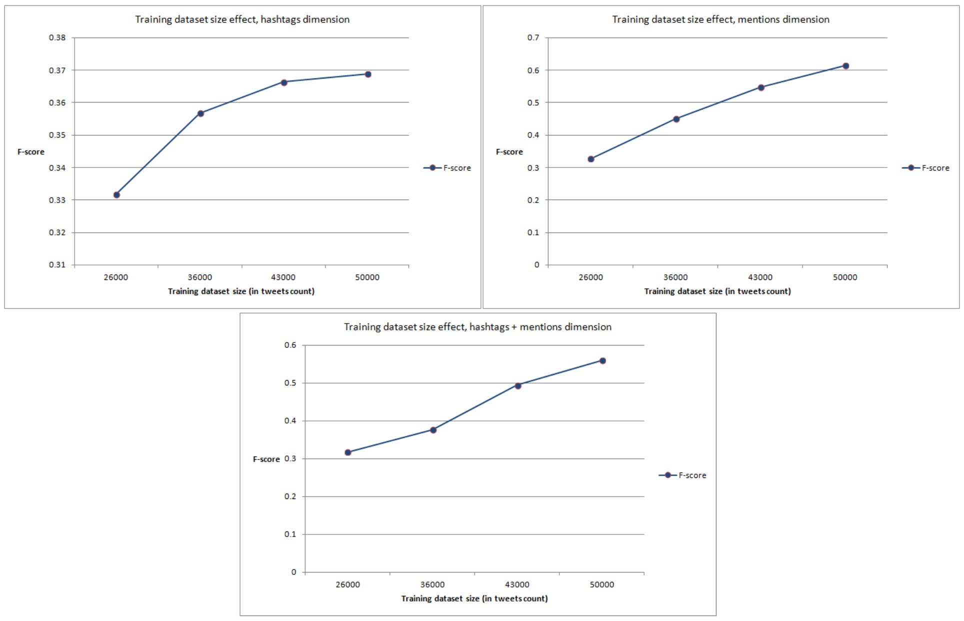


Figure 6.5: The effect of changing the training dataset size over the communities classifier performance. Case study B

The charts show the resulting F-score metrics of the changing the data size of the corresponding dimension.

The charts indicate that the overall performance of the classification increases slightly with increasing the dataset size, but the generally the results are quite close. The reason being the fact that the training and testing datasets were derived of the same overall dataset, even if the overall size was changing. This leads us to the second experiment, where the effect of fixing the testing dataset, and deriving the training dataset of different time periods from the overall dataset.

Conclusion

Increasing the training dataset size for the community Bayesian classifier enhances the accuracy of the classification slightly.

6.6.4 Experiment 2: Changing the Temporal Distance Between the Testing and Training Datasets

This experiment is directly related to the 3rd objective of this analysis. The first experiment studied the effect of changing the dataset size over the performance, while deriving both the testing and training datasets from the same overall time period. In this experiment, the effect of increasing the time span between the training and testing datasets is investigated.

The reasoning behind the experiment is to investigate whether the community information obtained for a general dataset represent general characteristics of the entire space of the data, or represent the characteristics of the data given a certain time period.

Approach

Derive the testing dataset from a fixed time period within the overall dataset, here this time period was at the end of the dataset time span. Then derive the training dataset of fixed size, but different time span periods.

Results and analysis

The charts below illustrate the F-score results of the different time span periods, for the Hashtags, Mentions, and the aggregation of Hashtags and Mentions dimensions. The X-axis in the chart represents the different time periods, with 1 being the closest time period to the testing dataset, and 4 is the farthest away.

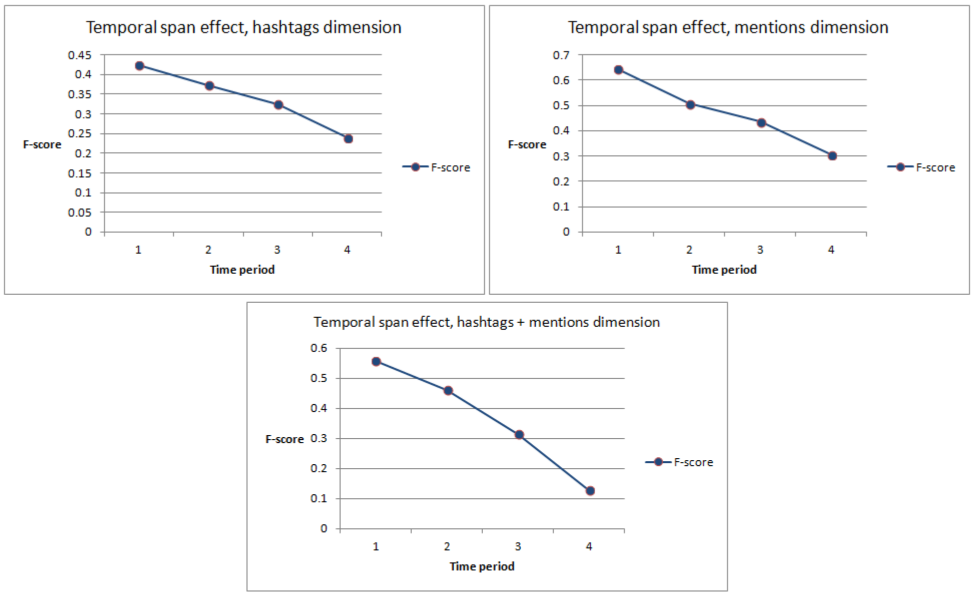


Figure 6.6: The effect of changing the time span period between the testing and training datasets over the communities classifier performance. Time period 1 being the closest to the testing dataset. Case study B

You can observe that increasing the time difference between the testing and training datasets decreases the accuracy of the analysis. This means that the community information detected now represent the characteristics of the data given the current or close time span. The community information still characterises the data for further away time spans, but with less accuracy.

Conclusion

- Increasing the time span period difference between the testing and training datasets decreases the accuracy of the analysis.

- The detected communities best describe the characteristics of the data space at the given time period.

- The detected communities can describe the characteristics of the data space at a different time period, but with reduced accuracy.

6.6.5 Experiment 3: Incorporating the Classified Communities into Sentiment Analysis

The trained classifier illustrated before provides the capability to classify a Tweet or a user from the currant data space to one of the available communities, if the community is not known beforehand. This experiment will investigate the results of incorporating these new classified communities into the sentiment analysis process.

Approach

1. Replace the communities' information at the testing dataset of the sentiment analysis process illustrated before, with the communities resulting of the community classifier.

2. Run the Bayesian Inference Model over these new communities, to calculate the new likelihood probabilities.

3. Then continue with the sentiment analysis procedure explained before.

Results and analysis

The accuracy of the communities classification of the testing dataset was 72%, which is high enough to continue the analysis process with.

For the sentiment analysis process. The bar chart below illustrates the results of the sentiment analysis process without incorporating the community information, with sentiments obtained through the unsupervised approach tool Sentistrength. The second bar (red) represents the sentiment result after incorporating the resulting classified communities information. The metrics used are the Positive Precision, Negative Precision, and Accuracy

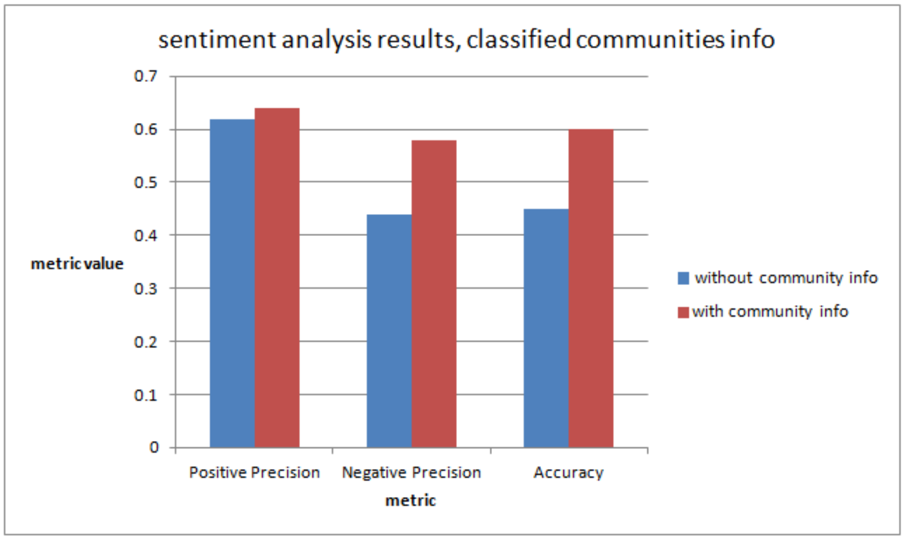


Figure 6.7: Sentiment analysis results. Results of Sentistrength (blue) vs results after incorporating the classified communities' information through the Bayesian Inference Model (red). Case study B

The resulting sentiment accuracy is obviously better than the Sentistrength's results. However, the results are still not as accurate as the results of the original detected communities illustrated at Figure 6.3. The reason being due to the relative accuracy of the communities classification; 72%.

6.7 Summary

This main goals of this chapter were to provide a practical application of the detected communities at the sentiment analysis process. The chapter began by providing a brief overview of the sentiment analysis process and some of the challenges it faces in Social Media, the main of which is the lack of context information. Then it progressed to discuss the proposed methodology to integrate the detected communities with the sentiment analysis, which should help providing the contextual background, needed for more proper analysis, like handling sarcasm for example.

The main approach to achieve this integration with the sentiment analysis was via the Bayesian Inference Model; where the sentiments of the users belonging to the detected communities provides the likelihood probability for any other, or testing, users. This chapter also addresses the issue of Dynamicity and Evolution in Social Media, where we constantly have new users and Tweets. A method to generalise the detected communities' information was needed to handle this constant change. A Naive Bayesian Classifier was used, with the characteristics of the detected communities as features, to build a classifier for classifying communities.

6.7.1 Sentiment Analysis Results

The provided results in this chapter regarding sentiment analysis show a quite noticeable improvement over the accuracy of the sentiments, after integrating the community information. The accuracy of the unsupervised sentiment analysis tool, Sentistrength, was about 47%, and it jumped to around 72% after the communities integration.

6.7.2 Communities' Classifier

Many experiments were conducted to investigate the accuracy and performance of this classifier, and the factors that contribute to this performance. The results are discussed briefly below. A much more comprehensive discussion can be found at the corresponding parts of the chapter.

Classifier's Accuracy

The results of the classifier showed about 70% accuracy at capturing the communities of a training dataset. The experiments' results below further discuss the analysis of the classifier and its results.

The Effect of Changing the Temporal Distance Between the Testing and Training Datasets

- Increasing the time span period difference between the testing and training datasets decreases the accuracy of the analysis.
- The detected communities best describe the characteristics of the data space at the given time period.
- The detected communities can describe the characteristics of the data space at a different time period, but with reduced accuracy.

The Effect of the Training Dataset Size

Increasing the training dataset size for the community Bayesian classifier enhances the accuracy of the classification slightly.

Incorporating the Classified Communities into Sentiment Analysis

The results of the sentiment analysis show an improvement over the accuracy of the sentiments, after integrating the newly classified communities. The accuracy of the unsupervised sentiment analysis tool, Sentistrength, was about 47%, and it jumped to around 60% after the communities integration.

Chapter 7

Conclusion and Future Work

This chapter provides a brief summary of the project's achievements and research results. The first section provides a general overview and discussion of the research field and project's goals. The second section discusses the main contributions of the project. And the last section is dedicated for future work and possible extensions to this project.

7.1 Project Discussion

This project discussed the vibrant field of Social Media analysis. This thesis presents a structured methodology to detect the communities and groups that the people tend to group themselves into implicitly. The detection procedures are targeting the implicit groupings rather than explicit ones made publicly by the users.

The project leverages the Link-based similarity and interaction as the core of the analysis process, rather than content-based analysis as many previous papers have discussed. The project then continues to investigate the different factors and parameters that contribute to the overall performance of the community detection process. Many experiments were conducted to understand the dynamics of the problem at hand, and to provide proper tuning for these factors to achieve better results.

The sociological fields of *Homophily* and *Influence* indicate that people within similar communities tend to develop similar ideas and perspectives regarding shared issues. The project investigates the applicability of this principle, along with the detected communities, for the well known problem of sentiment analysis. The project proposes an approach to integrate the information derived of the community detection procedure to enhance the accuracy of the sentiment analysis. This enhancement targets the sentiments of the content belonging to the community's members.

Social Media means in general are quite dynamic. Continuous changes and updates make the community and sentiment analysis restricted to the content of the current data snapshot. The project tackles this issue by building a community classifier based on the available communities' characteristics. This classifier can obtain the community affiliated with a user or Tweets belonging to the same space as the dataset used at the community detection procedure.

7.2 Contributions

As of this writing, and to my knowledge, this project has provided novel approaches and analysis in these areas:

- Thorough analysis and investigation of the factors and parameters that affect the Link-based community detection process. The project conducts thorough experiments and analysis to understand these factors, how they affect the community detection performance, and how to tune them to obtain better results.

- Utilising the sociological principles of *Homophily* and *Influence* to form a methodology to enhance the accuracy of the sentiment analysis process in Twitter. The project integrates the detected community information with the sentiment analysis procedure, to obtain more accurate results.

- Approaching the issue of evolution and dynamicity in Social Media content by building a classifier for the communities, based on the characteristics of the detected communities. This classifier can provide the most suitable community classification for a user or Tweets with unknown community information, as long as they belong to the same space of the dataset used at the community detection procedure.

7.3 Achievements

To better evaluate the degree of which the objectives of this project have been met, the points below outline each designated objective and some discussion regarding the level of which the project has achieved it.

- *Surveying the affiliated background materials, related work and researches to the project's fields, making sure to evaluate any of the used techniques and choosing the best performing ones, whenever possible.*

Chapter 2 of this thesis provides suitable illustration of all the concepts and researches required for the project's scope. The chapter provides some insights regarding Social Network Analytics in general, then goes more focused upon the problem of community detection in particular. The chapter provides a review of the possible algorithms and their differences. Then provides description about textual mining and sentiment analysis for Social Media content.

The consequent chapters all use concepts derived of this chapter. However, whenever any of these concepts or tools is set to be used, a thorough analysis and evaluation is done to illustrate its suitability for the problem at hand.

- *Investigating and implementing tools and procedures for detecting the communities within a Twitter content. In addition to studying the execution performance of each of these procedures, making any possible alterations to the implementation to enhance the overall performance.*

Chapter 3 details the implementation procedures for the community detection approach used here. It provides pseudocodes for the algorithms, and examples to illustrate the steps. The chapter also illustrates bad performing implementations, and provides better approaches when possible. And finally the chapter provides some explanation regarding the execution complexities, however, not enough analysis was done regarding the execution complexity, due to time limitations.

- *Studying and analysing the different parameters, algorithms, and factors that contribute to the community detection procedures, evaluating each of these factors individually or in the aggregate.*

This objective has been fully achieved. Chapter 4 provides comprehensive experiments and trials regarding the factors contributing to the community detection process. Many metrics and evaluation methods were used, and many chart types were utilised to present the results in the most efficient way. Each experiment finally presents the drawn conclusions. All the analysis and evaluation processes were based upon a case study outlining the general approach and results.

- *Applying the community detection procedure at the sentiment analysis process, and evaluating the results.*

Chapter 5 discusses this objective. The chapter provides a recap for the theoretical background for integrating the communities with the sentiment analysis. It also discusses the approach used for the integration and its mathematical background, with an example to clarify the procedure. A case study is then presented to illustrate the entire procedure, starting with the community detection along with the integration with the sentiment analysis, in addition to the evaluation approach. It was proven that this integration does enhance the accuracy of the sentiment analysis indeed.

- *Building a general community classifier based on supervised learning, using the characteristics of the detected communities of the available dataset. In addition to studying the parameters affecting its performance, and evaluating its results.*

Chapter 5 also discusses this objective. The last section of the chapter discusses the approach to build the communities classifier and the evaluation affiliated to it. The results of the evaluation show good accuracy in capturing the communities, around 70%. The section also shows some experiments regarding the factors that contribute to the classifier's performance. The section finally experiments the integrating the classified communities with the sentiment analysis process, and the evaluation shows some enhancement. However, there are still some room for further work, as shall be explained at the future work section.

7.4 Future Work

After performing the designated objectives of the project, and having better understanding of this research area, some possible extensions and additions to the current project were observed. The following items represent possible extensions to the topics addressed within this thesis, providing the basis for any future build-up upon this project's achievements.

7.4.1 Using Overlapping Community Detection Algorithms

The nature of the communities in Social Media is more of overlapping rather than isolated. People in real life lean to sometimes be part of different groups rather than totally isolated ones. This reflects to the Social Media content as the social behaviour tends to be the same, regardless of the different means.

The field of overlapping community detection is still relatively new. Modern researches have tackled the topic of overlapping community detection, however, all of these new algorithms suffer of bad execution time complexities, which is problematic for big content as in Social Media.

7.4.2 Better Approaches Regarding Evolution and Dynamicity

As illustrated at the background review chapter, the Social Media network evolves and gets updated constantly, which requires a dynamic and adaptive analysis techniques and modeling. A good approach in this regards would be to have many snapshots and analysis of the data space at different times or settings. The snapshots continue until the analysis results reach some kind of stability.

7.4.3 Semantic Analysis of the Detected Communities

The community detection provided at this thesis operated clustering techniques to obtain different groups and communities within the dataset. Ground truth information, which are the political affiliations in this case, provided the basis of the semantic meaning for each of the detected communities. Obtaining ground truth information for real life problems in Social Media is usually not applicable. That is why it might be a good addition to introduce an approach that provides the semantic meaning of the detected communities.

Appendix A

Case Study A Results

Hashtags dimension results

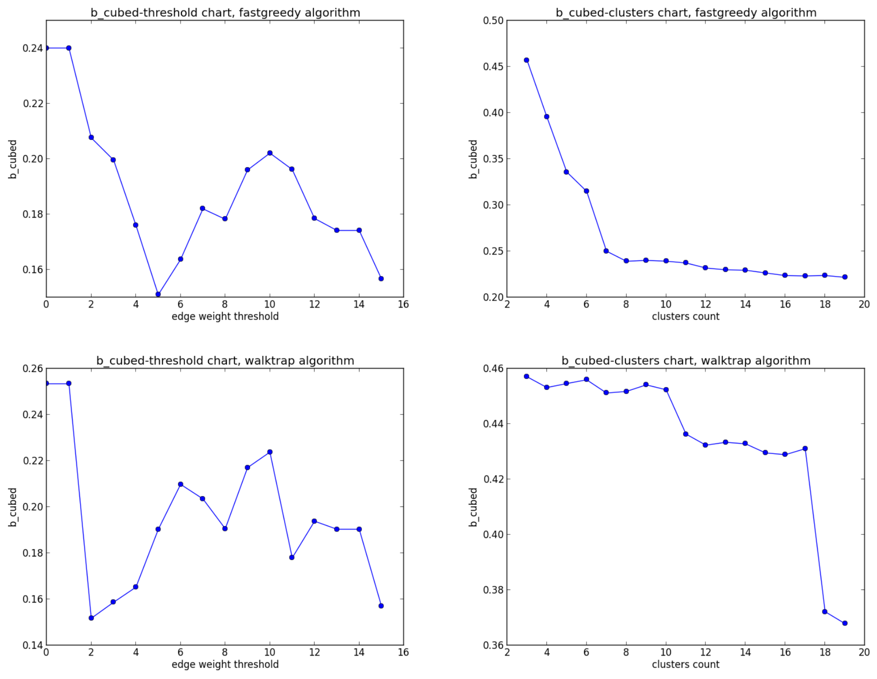


Figure A.1: Hashtags dimension results

Mentions similarity dimension results

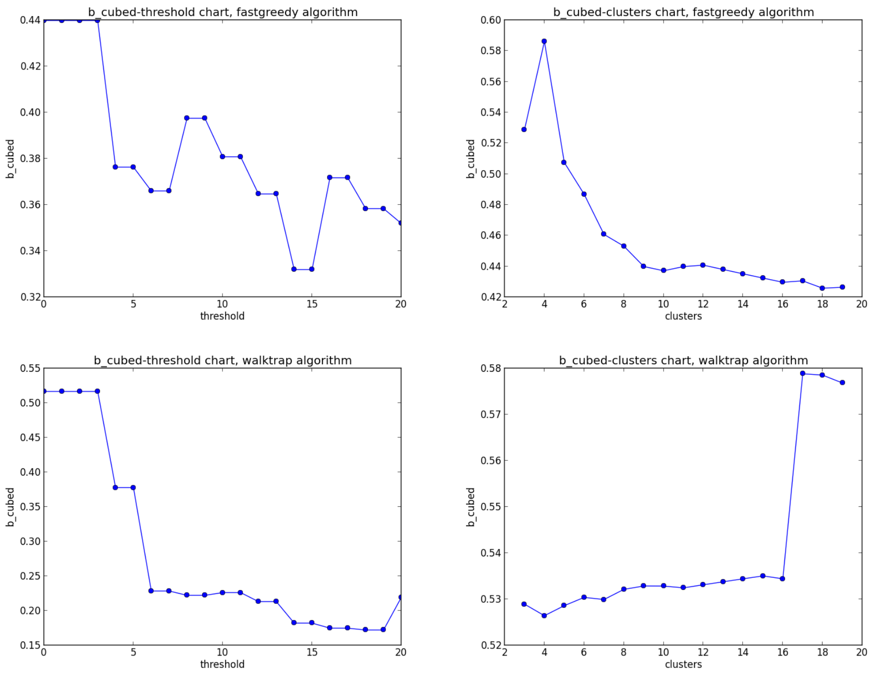


Figure A.2: Mentions similarity dimension results

Links dimension results

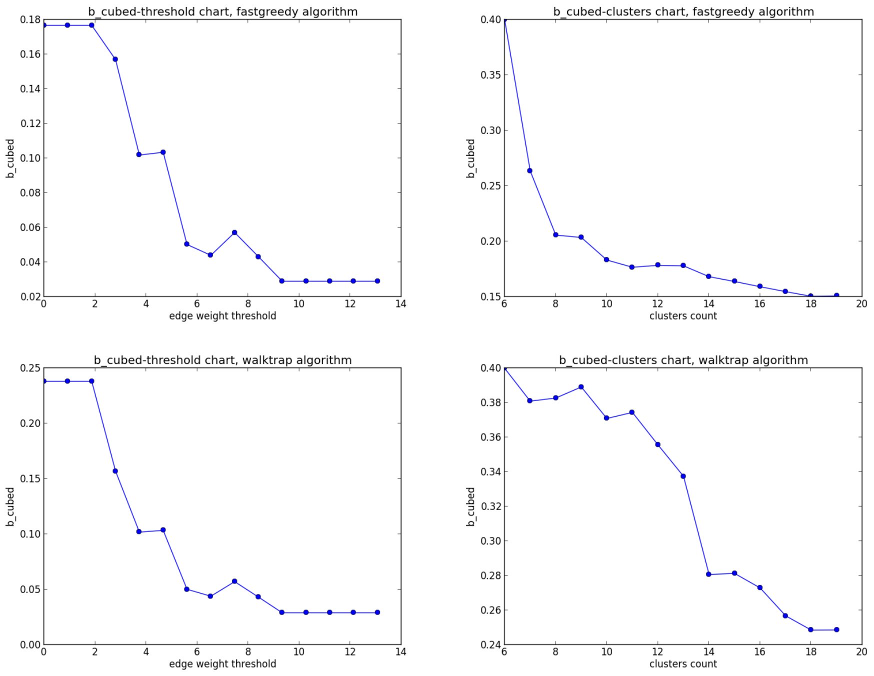


Figure A.3: Links dimension results

Mentions interaction dimension results

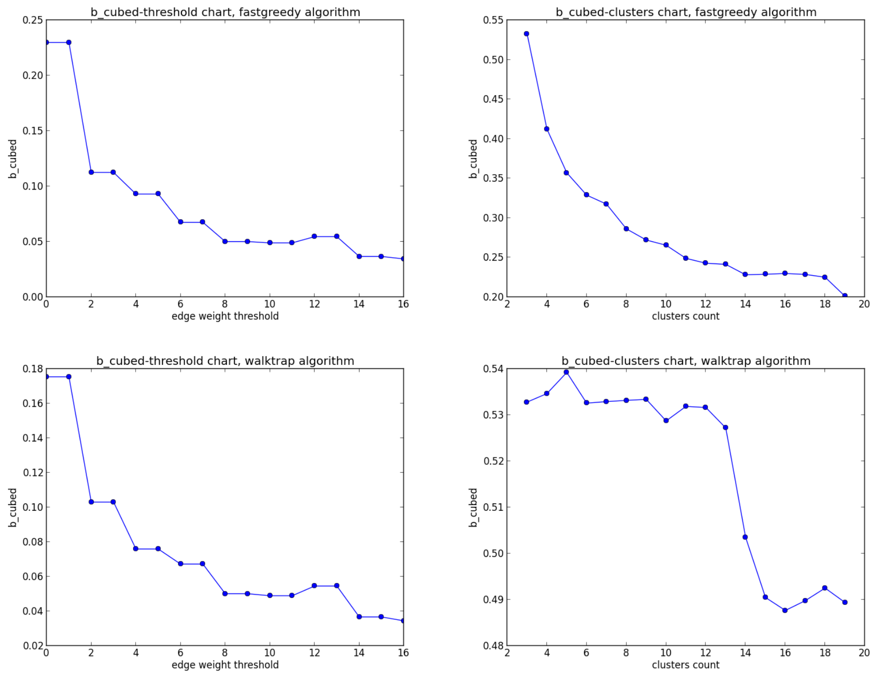


Figure A.4: Mentions interaction dimension results

Replies dimension results

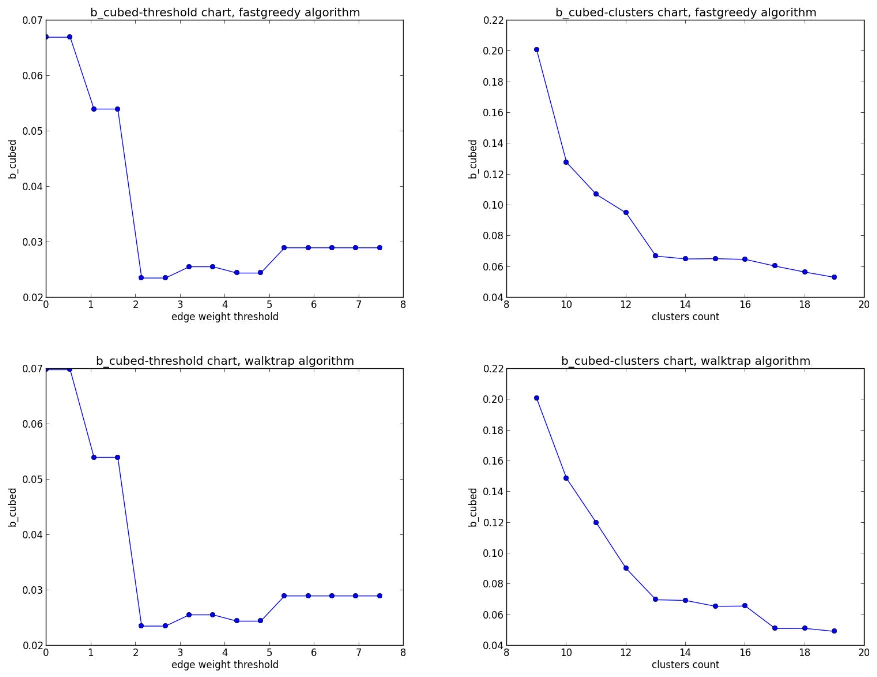


Figure A.5: Replies dimension results

Retweets dimension results

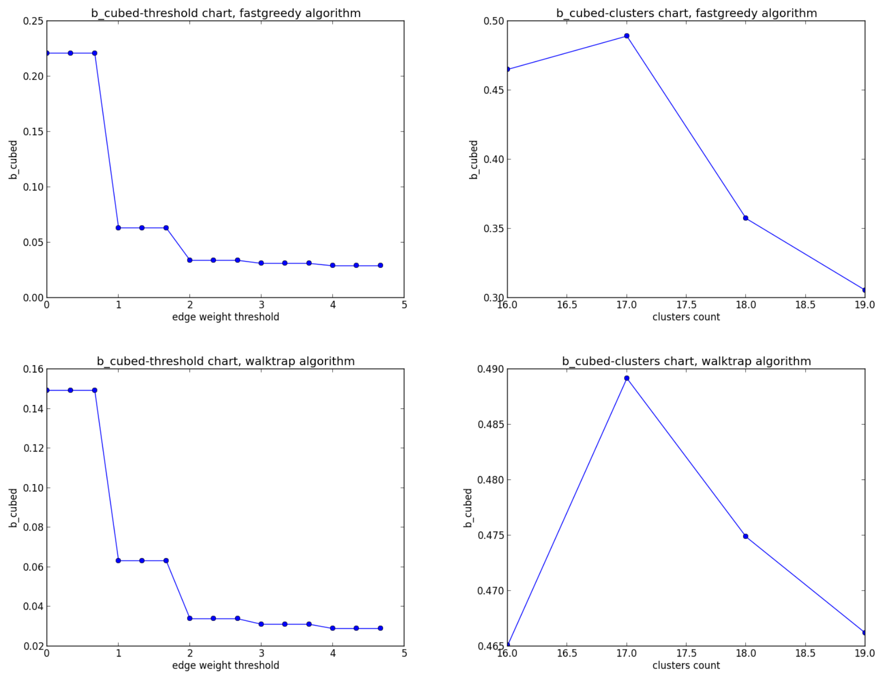


Figure A.6: Retweets dimension results

Aggregated interaction dimensions results

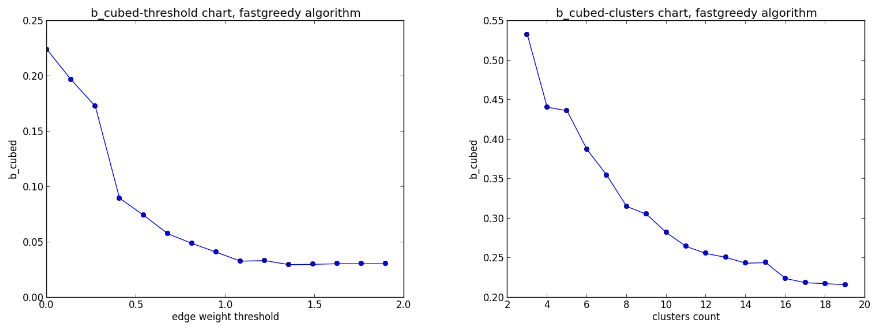


Figure A.7: Aggregated interaction dimensions dimension results

Aggregated interaction excluding Replies dimensions results

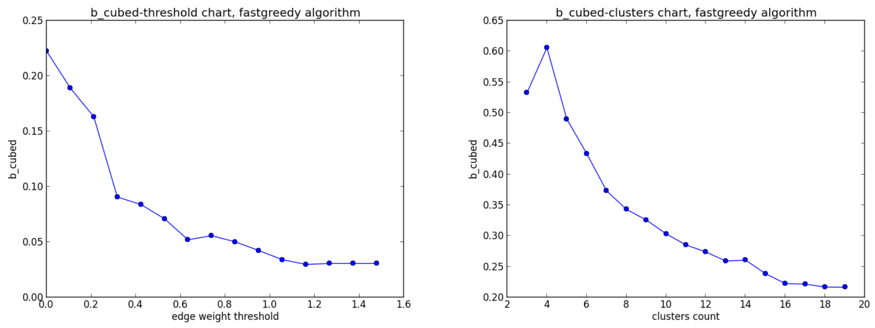


Figure A.8: Aggregated interaction excluding Replies dimensions dimension results

Links+ Hashtags aggregated dimensions results

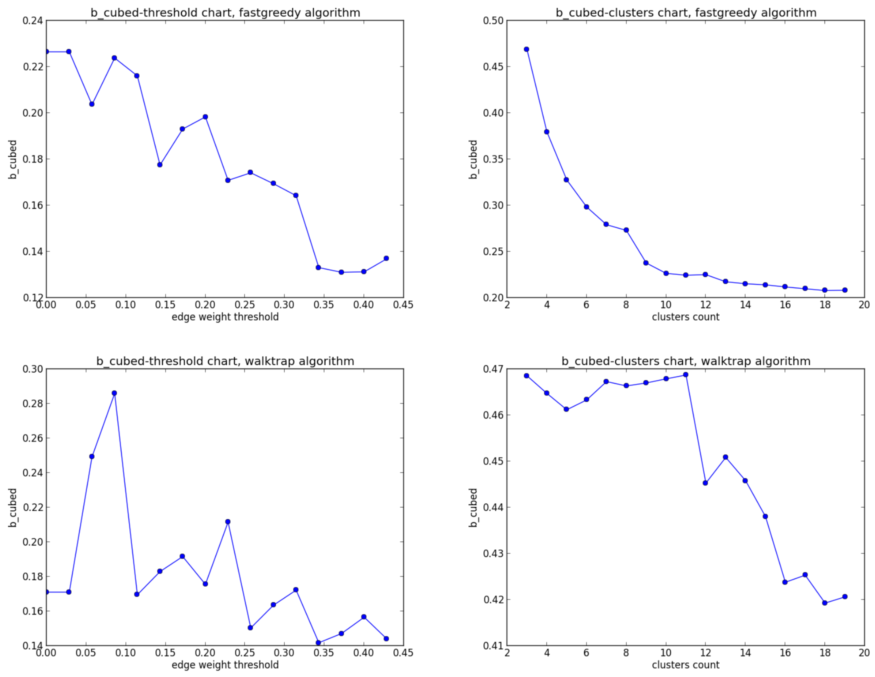


Figure A.9: Links+Hashtags aggregated dimension results

All aggregated dimensions results

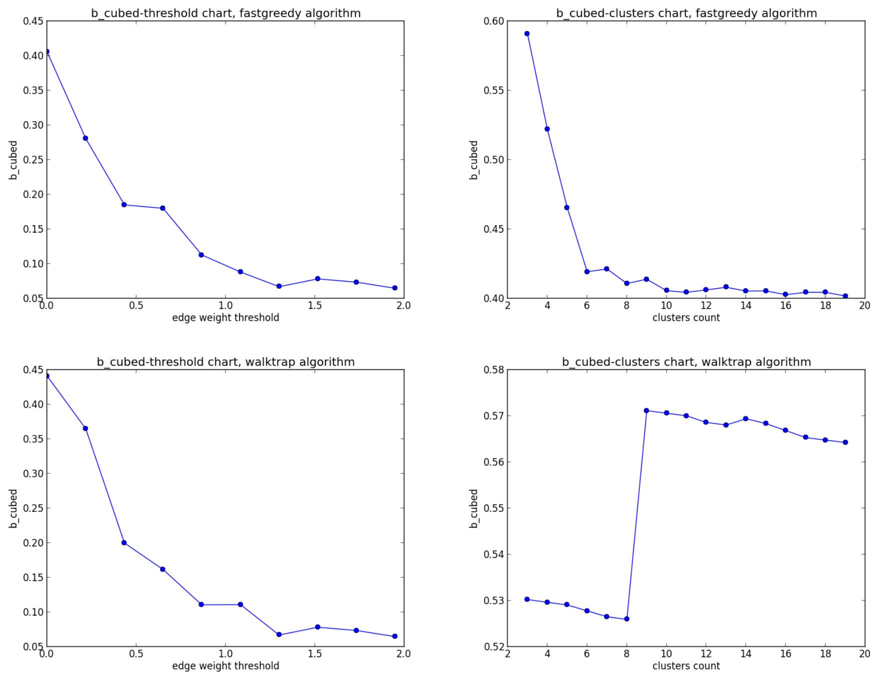


Figure A.10: All aggregated dimension results

Bibliography

[1] Jie Tang-Long Jiang Ming Zhou Chenhao Tan, Lillian Lee and Ping Li. User-level sentiment analysis incorporating social networks. In *Proceedings of KDD 2011*, 2011.

[2] Jonathan Clark. Text mining and scholarly publishing. *Publishing Research Consortium 2013*, 2013.

[3] David Combe. A comparative study of social network analysis tools. *International Workshop on Web Intelligence and Virtual Enterprises 2*, 2010.

[4] Javier Artiles Enrique Amigo, Julio Gonzalo and Felisa Verdejo. A comparison of extrinsic clustering evaluation metrics based on formal constraints. *Departamento de Lenguajes Sistemas Informaticos UNED, Madrid, Spain*, 2009.

[5] Vincent Labatut Günce Keziban Orman and Hocine Cherifi. On accuracy of community structure discovery algorithms. *Galatasaray University, University of Burgundy*, 2011.

[6] Yulan He Hassan Saif and Harith Alani. Semantic sentiment analysis of twitter. *Knowledge Media Institute, The Open University, United Kingdom.*

[7] igraph.sourceforge.net/. igraph network analysis tool.

[8] igraph.sourceforge.net/doc/python/igraph.pdf. igraph network analysis tool, documentation.

[9] internetworldstats.com/stats.htm. Internet world statistics.

[10] Finn Jensen. *Introduction to Bayesian Networks*. Springer, 1997.

[11] K McKeown and V. Hatzivassiloglou. Predicting the semantic orientation of adjectives. In *Proceedings of the Joint ACL/EACL Conference*, 1997.

[12] Lynn Smith-Lovin Miller McPherson and James M Cook. Birds of a feather: Homophily in social networks. *Department of Sociology, University of Arizona, Tucson, Arizona. Department of Sociology, Duke University, Durham, North Carolina*, 2001.

[13] Richard Neapolitan. *Probabilistic Reasoning in Expert Systems*. John Wiley, 1990.

[14] newsroom.fb.com/Key Facts. Facebook news room.

[15] nltk.org/. Natural language toolkit.

[16] Bo Pang and Lilian Lee. Opinion mining and sentiment analysis. *Now Publishers Inc*, 2008.

[17] Symeon Papadopoulos. Community detection in social media. *Center for Research and Technology - HELLAS, Information Technology Institute*, 2011.

[18] pytables.org/moin. Pytables, managing hierarchical datasets.

[19] C. Banea R. Mihalcea and J. Wiebe. Learning multilingual subjective language via cross-lingual projections. *Proceedings of the Association for*, 2007.

[20] Van Rijsbergen. Foundation of evaluation. *Journal of Documentation*, 1974.

[21] Martin Kilduff Ronald Burst and Stefano Tasselli. Social network analysis: Foundations and frontiers on advantage. *The Annual Review of Psychology*, 2013.

[22] scipy.org/. Scientific library for python (scipy).

[23] sentistrength.wlv.ac.uk/. Sentistrength short text sentiment analysis tool.

[24] statisticbrain.com/twitter statistics/. Statistics brain.

[25] support.twitter.com/articles/14023-what-are-replies-and mentions. Twitter help centre, replies and mentions.

[26] support.twitter.com/articles/77606-faqs-about-retweets rt. Twitter help centre, retweets.

[27] Lei Tang and Huan Liu. *Community Detection and Mining in Social Media.* Morgan & Claypool Publishers, 2010.

[28] Mike Thelwall and Kevan Buckley. Topic-based sentiment analysis for the social web: The role of mood and issue-related words. *Journal of the American Society for Informaton Science and Technology*, 2012.

[29] Paltoglou G.-Cai D. Thelwall M., Buckley K. and A. Kappas. Sentiment strength detection in short informal text. *Journal of the American Society for Information Science and Technology*, 2010.

[30] trendak.com/trendak/. Trendak social media data analysis.

[31] tweetminster.co.uk/. Tweet minister.

[32] Zhi-Feng Hao Ying-Jun Wu, Han Huang and Feng Chen. Local community detection using link similarity. *Journal of Computer Science and Technology*, 2012.