Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Distributed Computing

Classify the News

Bachelor's Thesis

Dominik Bruggisser

dominibr@student.ethz.ch

Distributed Computing Group
Computer Engineering and Networks Laboratory
ETH Zürich

Supervisors:
Philipp Brandes, Klaus-Tycho Förster
Prof. Dr. Roger Wattenhofer

April 12, 2016

Abstract

There is simply too much information around us to digest all of it. There should be a simple method to classify news pages in order to filter the irrelevant parts out. This paper introduces methods based on which such a tool can be built. The introduced methods allow searching for subjects on Twitter and clustering them based on meta data. Furthermore, some basic ideas about detecting a *filter bubble* are described, implemented using a dendrogram and deployed in a website.

Contents

Chapter 1

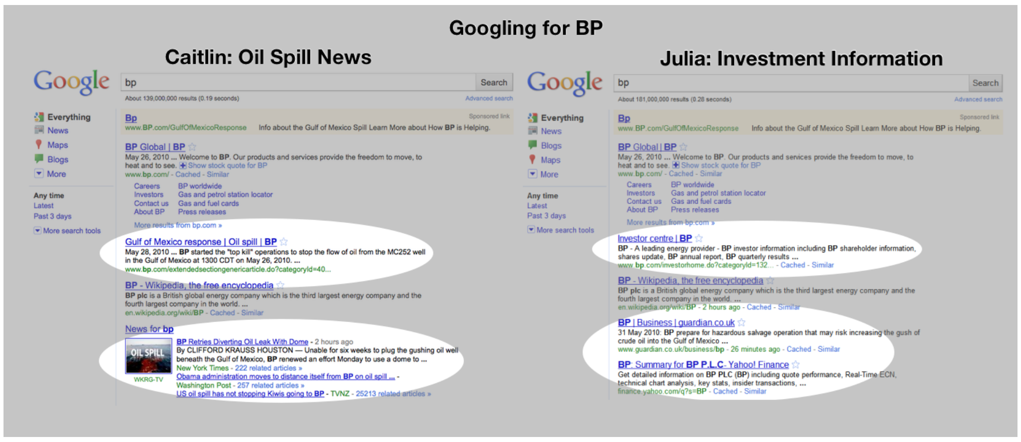
Introduction

In the era of the internet, news is omnipresent. The consumers are exposed to a huge amount of information out of which they are supposed to build their opinion. While intuition gives the impression that this leads to a better understanding of the world and well-grounded opinions, in reality people tend to prefer articles that accredit the opinion they already have. This phenomenon is known as *selective exposure*: An article that confirms the own opinion is taken as fact, other articles that disprove it are believed to be implausible. The effect is that people with certain views prefer certain news sites and rather to talk to people with similar opinions than to those opposing them. With the advent of the Web 2.0, companies started to use this matter of fact in order to attract users to their websites and to make them stay longer. Facebook has introduced the custom news feeds to present content it believes the user prefers. Google presents personalized search results based on guesses about what pages the users would like, even if a user is not logged in [1]. Figure 1.1 an extreme example of such personalized results. These personalized results lead to a phenomenon called *filter bubble.*

1.1 Filter Bubble

The term *filter bubble* was coined by Eli Pariser. It refers to an artificial environment in which contradicting viewpoints are suppressed to give the reader a simplified impression of his surroundings. The environment is generated through algorithms which selectively filter the results based on the user's behaviour. According to Pariser, the result of such a constructed world is "a world in which there's nothing to learn" [2].

Filter bubbles are computer generated, but also desired by the user. Sources that show other opinions and views are avoided and so in order to stay relevant, information providers are often forced to present selective contents. Knowing about the concept of filter bubbles raises the question whether there is a way to visualize or even evade them and to reach content which extends one's opinions.



Caitlin: Oil Spill News

About 139,000,000 results (0.19 seconds)

Bp
www.BP.com/GulfOfMexicoResponse Info about the Gulf of Mexico Spill Learn More about How BP is Helping.

BP Global | BP
May 26, 2010 ... Welcome to BP. Our products and services provide the freedom to move, to heat and to see. Show stock quote for BP
www.bp.com/ - Cached - Similar
Careers BP worldwide
Investors Gas and petrol station locator
Contact us Gas and fuel cards
About BP Press releases
More results from bp.com »

Gulf of Mexico response | Oil spill | BP
May 28, 2010 ... BP started the "top kill" operations to stop the flow of oil from the MC252 well in the Gulf of Mexico at 1300 CDT on May 26, 2010. ...
www.bp.com/extendedsectiongenericarticle.do?categoryId=40...

BP - Wikipedia, the free encyclopedia
BP plc is a British global energy company which is the third largest energy company and the fourth largest company in the world. ...
en.wikipedia.org/wiki/BP - Cached - Similar

News for bp
BP Retries Diverting Oil Leak With Dome - 2 hours ago
By CLIFFORD KRAUSS HOUSTON — Unable for six weeks to plug the gushing oil well beneath the Gulf of Mexico, BP renewed an effort Monday to use a dome to ...
New York Times - 222 related articles »
OIL SPILL WKRG-TV
Obama administration moves to distance itself from BP on oil spill ... - Washington Post - 257 related articles »
US oil spill has not stopping Kiwis going to BP - TVNZ - 25213 related articles »

Julia: Investment Information

About 181,000,000 results (0.28 seconds)

Bp
www.BP.com/GulfOfMexicoResponse Info about the Gulf of Mexico Spill Learn More about How BP is Helping.

BP Global | BP
May 26, 2010 ... Welcome to BP. Our products and services provide the freedom to move, to heat and to see. Show stock quote for BP
www.bp.com/ - Cached - Similar
Careers BP worldwide
Investors Gas and petrol station locator
Contact us Gas and fuel cards
About BP Press releases
More results from bp.com »

Investor centre | BP
BP - A leading energy provider - BP investor information including BP shareholder information, shares update, BP annual report, BP quarterly results ...
www.bp.com/investorhome.do?categoryId=132... - Cached - Similar

BP - Wikipedia, the free encyclopedia
BP plc is a British global energy company which is the third largest energy company and the fourth largest company in the world. ...
en.wikipedia.org/wiki/BP - 2 hours ago - Cached - Similar

BP | Business | guardian.co.uk
31 May 2010: BP prepare for hazardous salvage operation that may risk increasing the gush of crude oil into the Gulf of Mexico ...
www.guardian.co.uk/business/bp - 26 minutes ago - Cached - Similar

BP: Summary for BP P.L.C- Yahoo! Finance
Get detailed information on BP PLC (BP) including quote performance, Real-Time ECN, technical chart analysis, key stats, insider transactions, ...
finance.yahoo.com/q?s=BP - Cached - Similar

Figure 1.1: An example of a filter bubble [3]

1.2 Problem statement

Users prefer spending time reading news which support their personal views and beliefs. This leads to a self-inflicted filter bubble. The combination of self-inflicted and algorithm based filter bubble strengthens these beliefs. Considering the diverse possibilities a single user has to redistribute his opinion over the internet, the impact of wrong news spreading can be dangerous, for example considering elections. Yet there is too much information around for a single user, even if there was nothing like a filter bubble, to digest all of it. There should be a way to analyse all the information presented in a simple and effective way.
To avoid filter bubbles and selectively sort the contents of news sites, looking at meta data should be enough to present news in a well-arranged way in order to stay well informed. To do so, it should be enough to visualize a user's filter bubble and show connections to other users with similar interests. With this information, a user can follow others and get a better overview about the whole topic.
Tests with the resulting tool described below showed that up to 75% of users that are clustered end up in a reasonable cluster. The testing is described in Chapter 5.

Chapter 2

Data Source

Social media has evolved from simple newsgroups into complex information networks. Every day, millions of user created content is uploaded to social networks. The content differs from conventional media in two major ways: Content is unreliable since everyone can create content. At the same time, this prevents one-sided, media-monopoly based information. On the other hand, reputation is no longer measured by the name of the publisher, but rather by the count of followers the publisher has. This has again advantages and disadvantages for the user. Partly, it leads to a more diverse media landscape, partly the focus changes from producing good content to gaining as many followers as possible, sometimes even with bought followers as outcome [4].
Other than conventional media, social media adapts fast to changes, presenting news updates usually within minutes of an event. This makes social media an optimal source to stay up to date.

2.1 Twitter

Twitter is a good source for data mining and analysis. Twitter is an easy to use microblogging service that offers huge amounts of data. Its users can post statuses (*tweets*), like and share (*retweet*) other user's tweets and follow other users to receive their contributions in realtime. Within tweets, other users can be *mentioned* using the "@" sign, or tweets can be in *response* to other tweets. Published content is versatile and covers everything from pointless babble to world news, images, videos and even polls. Twitter can be accessed through the web browser, apps and even SMS, which leads to a large amount of user contributions. Currently, over 7100 tweets are posted every second [5]. Unlike on other platforms, on twitter users can generate meta data on their on by adding keywords (*hashtags*) to their tweets. Twitter links tweets with the same hashtag together to categorize tweets by topics and to give best possible search results when a certain topic is queried. Twitter recommends to its users who they could follow to get more content. Again, these recommendations are based on algorithms that find similar content and thus create a filter bubble.

2.2 Crawling

Twitter offers an API through which all data, enriched with meta data, can be crawled. Unlike other social platforms like Facebook, Twitter allows access to every user and every tweet except so called protected tweets. This means that no follower relation is required in order to access the data.

2.2.1 Search and REST API

The *search API* and *REST API* allow developers to search for topics and users, crawl for social relations (like followers) and get status updates. It also offers a lot of options to update user profiles, statuses and follower relations.

2.2.2 Stream API

The stream API lets developers receive real-time data based on keywords, users or *geotags*. This is especially useful to analyse demographic data or to track live events.

Chapter 3

Concept

In this chapter, I will give a brief overview of the ideas behind the application and discuss the applied concepts behind it. For technical details about the implementation, see Chapter 4.
The goal was to create a website which allows the user to cluster content about a specified subject. Such a website should crawl Twitter and show the resulting information in a well-arranged way. The easiest way to do so is to present data as a graph. There exist already such graphs, as shown in Figure 3.1 [6]. This graph was constructed from Instagram hashtags. It shows used co-tags about the Israeli-Palestinian conflict with yellow tags indicating pro-Palestinian posts, orange tags indicating pro-Israeli posts and purple tags representing religious and Muslim contributions. The application should produce similar results, but

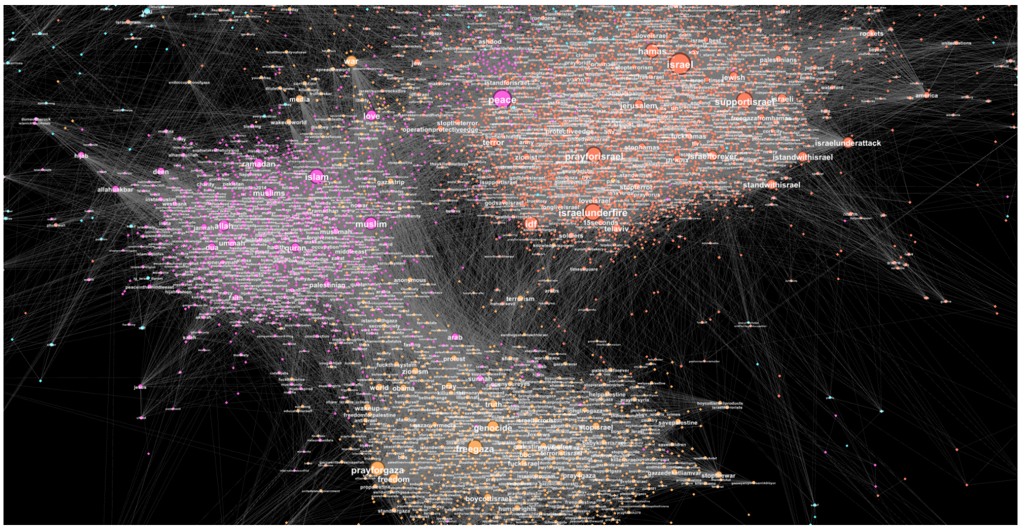


Figure 3.1: Instagram hashtag graph, 1) pro-Israeli (Orange), 2) pro-Palestinian (Yellow), and 3) Religious / Muslim (Purple)

unlike the given example, the application should allow user input, i.e. the user can decide what subject he wants to analyse. In the following Sections, the basic concepts of such an application are discussed.

3.1 Data Model

Twitter offers a whole range of meta data. Figure 3.2 shows the resulting data model used for the application (figure 4.2 shows a more detailed version of the data model). Based on the model, there is a natural way to construct a graph: Each user (*twitter_user*) represents a node. Edges between the nodes are created as a result of common characteristics of the nodes. The most obvious characteristic is the follower relationship, where a user following an other user implies that there are common interests. Mentioned users or replies allow a similar argument. The other edges are based on retweets, common hashtags and websites that both users share in their tweets. In addition to these edges, a connection between two nodes is established based on the number of common followers divided by the maximum number of followers of both users, i.e. $\frac{\#commonfollowers}{\max\limits_{user\ i}(followersCount_i)}$.

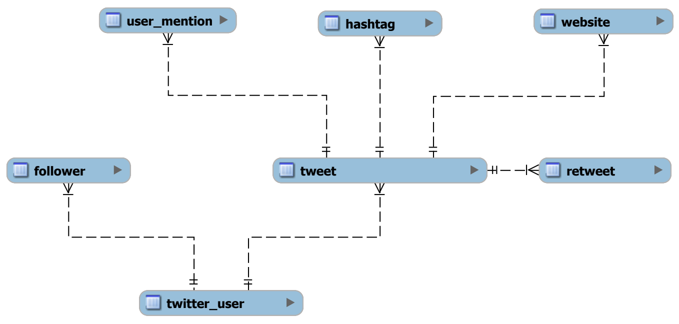


Figure 3.2: relations between tables: Users are stored in the table *twitter_user*. Each user has followers, i.e. users who follow them. Posts of users are stored in the table *tweet*. Each tweet can conatin user mentions (*user_mention*), hashtags and websites. If a tweet gets retweeted, the relation is stored in the table *retweet*.

There are no edges based on the actual content. Comparing the user's profile descriptions and analysing the subjects of their tweets could improve the quality of the clustering. Conversely, it opposes the hypothesis that clustering is possible based on meta data. And there is an other issue: Simply comparing the text is not enough, as the examples in Figures 3.3 and 3.4 show. More elaborate algorithms would be required to analyse the semantic meaning of each user's tweets. Even then the quality of results may vary because tweets are limited to 140 characters. Other information offered by Twitter, like geographic data, is not used to create a graph.

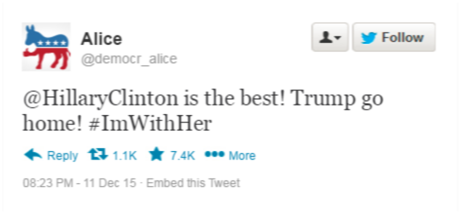

Alice
@democr_alice
@HillaryClinton is the best! Trump go home! #ImWithHer

Figure 3.3: Tweet of a Hillary voter

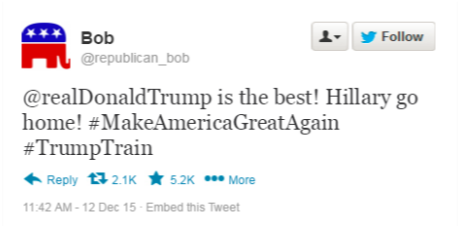

Bob
@republican_bob
@realDonaldTrump is the best! Hillary go home! #MakeAmericaGreatAgain #TrumpTrain

Figure 3.4: Tweet of a Trump voter

A different approach to create a graph is to create a node for each hashtag. To define edges, users are distributed over the different nodes based on the hashtags they used in their tweets. In addition to the edges mentioned above, an additional artificial edge indicates that two hashtags are used by the same user. This approach has a big drawback: While hashtags like ”#ImWithHer” or ”#TrumpTrain” have a clear connotation, other hashtags like ”#election2016” are used in multiple contexts, so a resulting graph is incomplete.

3.2 Clustering

Given a clustering algorithm as described below, the data is clustered in the following way: First, the graph is clustered based on following relationships, user mentions, replies and retweets. The resulting clusters represent users that are ”socially” close to each other. The hypothesis states that these users have similar opinions. In a next step, in order to find similar interests, the resulting clusters are clustered again, but this time based on used hashtags and websites. That way filter bubbles become visible: Users that end up in the same cluster in the second clustering process tend to be caught in a filter bubble, users that end up with those of different clusters presumably evaded it.
In a further step, opinions can be visualized by extracting hashtags from each cluster. The graph topology between clusters stays the same, within each cluster common hashtag nodes are joined. That way the closer in the middle a node is, the more commonly is it used. This does not directly contribute to the problem statement, but gives a further insight into prevalent opinions and frequent subjects (also, hashtags that appear in every cluster indicate more popular subjects).

3.2.1 Approaches

In a first approach, I applied hierarchical ”bottom up” clustering in a simple fashion by joining clusters until a certain similarity threshold was reached. The similarity is measured as the proportion of the edges inside the cluster to the edges that go outside of the cluster. The threshold is the value below which two clusters are not joined anymore because these are considered too different. In order to cluster, every node is viewed as a small sub-cluster. The edges between these clusters are inherited from the nodes contained in it, which are created as stated above. Then, each cluster is compared with all others. The clusters are merged based on the result from this comparing. In the comparing process, the relation between nodes inside the cluster and those going outside of it are compared. In order to compare, the average function is used, that is, each node of one cluster is compared with all nodes of the other cluster. Then, the average represents how similar the two clusters are. The higher the fraction of edges within, the better.

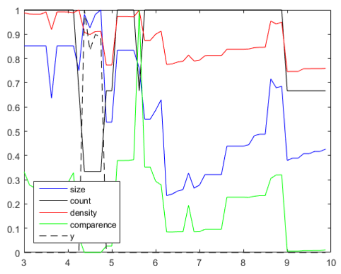


Figure 3.5: Plot of used parameters for simple bottom up clustering

The advantage of such an approach is its simplicity. Figure 3.5 shows the result of multiple runs, where each time the same data was used, but the threshold was changed, which leads to more sub-clusters being joined into bigger clusters. The x-axis represents the threshold. The quality of the resulting clusters depends on the density and the difference (in the figure called *comparence*) between the clusters, that is how much are nodes connected within a cluster, and how many connections are there in between the clusters. The goal is to find an optimal threshold at which the least amount of edges between the clusters exist, while inside the cluster, nodes are connected well, i.e. there are no nodes that are only connected loosely. Experimental data with different datasets showed that such a threshold does not always exist. Also, performing search for the correct threshold is expensive because results are arbitary so that algorithms like binary search are not applicable. That is, the qualitative outcome of changing the threshold cannot be predicted. An example outcome of the algorithm is shown in Figure 3.6. The graph shows three clusters based on the search for NFL teams. Grey edges indicate edges between the clusters. Blue edges indicate connections within *Seattle Seahawks* related tweets, red ones stand for *Denver Broncos* and green ones for *Philadelphia Eagles*. The quality of the resulting graphs vary strongly depending on chosen parameters, namely the threshold, the level of dropping clusters and the expected amount of clusters in the result. For some subjects, the amount of

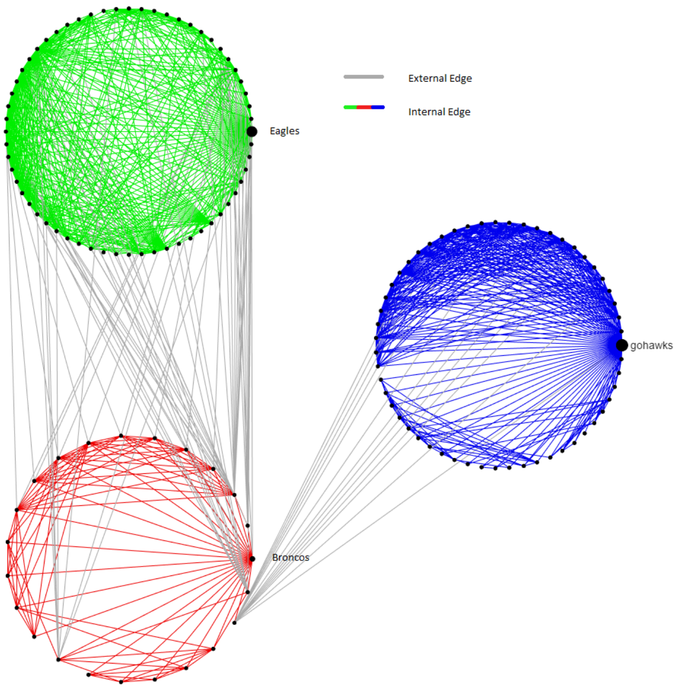


Figure 3.6: The resulting hashtag graph of bottom up clustering. Edges within clusters are coloured red, green and blue, edges that connect the clusters with each other are coloured grey.

resulting clusters is important, as it would not make sense to create two clusters for three sports teams. Other subjects require a very small threshold to avoid one huge and one tiny cluster. This requires manual fine tuning of parameters for each new dataset, so the clustering process becomes cumbersome.

3.2.2 Dendrogram

A dendrogram is a tree diagram that illustrates the arrangement of clusters. Each leaf corresponds to a user, each node corresponds to the merging of two sub-clusters. The root of the tree is the union of all clusters. Contrary to the hierarchical clustering described above, no threshold is defined, but the clustering is continued until the root is reached, i.e. only one cluster containing all data is left. This allows multi step clustering as described above, which also solves the problem statement. In addition, to the user it is more obvious how clusters were created. No further measures or analysis is required, so the algorithm is only applied once. The results of this approach are described in Chapter 5.

3.3 Problems

Full automated perfect clustering is impossible with Twitter data for a few reasons:

- Trolls
- Pointless babble
- Indetermined users
- Passive users

The internet is full of trolls. Twitter is no exception. Tweets from trolls are impossible to cluster because their opinion does not represent a real view and, even worse, their followers often represent an opposing group in respect of their expressed opinion. The same problem arises from users who post a lot of pointless babble, which, according to a study conducted by *pearanalytics.com*, account for up to 40% of the overall tweets [7].

Chapter 4

Implementation

In this chapter, implementation technologies and some implementation details are described.

The whole project was implemented in Java with the program structure shown in Figure 4.1. Note that the data collection and data analysis are completely separated, making it easier to extend or alter the project. There are four main packages: The package *twitter* handles all interactions with the Twitter API and schedules new tasks, the package *datamodel* contains classes to manage all data, the package *graph* contains the graph logic as described above, and the package *controller* handles requests from the website. In addition there is a util package which offers functionality for networking (e.g. looking up web domains), preprocessing edges and accessing the file system. The full data model is shown in Figure 4.2. To allow parallel processing of more than one subject, to each subject a unique id (*twitter_query.id*) is assigned. Related subjects are combined in the table *dataset*. Within the project, the following technologies were used:

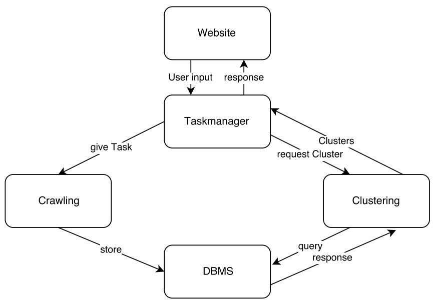


Figure 4.1: Component relationship

- **Eclipse**: Eclipse offers an easy to use environment for programming. It has a lot of extensions and built-in tools for development.
- **Maven**: Apache Maven helps integrating external modules and configuring the project information using the POM (*Project Object Model*) and supports the whole development cycle of the project.
- **MySQL**: All collected data is stored in a relational database using MySQL. The complex DBMS carries out all data related operations and takes care of issues like parallel access and query optimization which leads to a performance increase. The connection is established through the *JDBC* driver. This allows full control over the executed operations (like projections) and received data.

- **Twitter4J**: Twitter4J is an unofficial Java library for the Twitter API. It is easy to integrate into projects and allows access to all Twitter API functionality.
- **sigmajs**: Sigma is a JavaScript library dedicated to graph drawing. It is used to create the resulting graphs.
- **svn tortoise**: Subversion is used to avoid data loss.

In the following Sections, the crawling process, clustering and the functionality of the website are described.

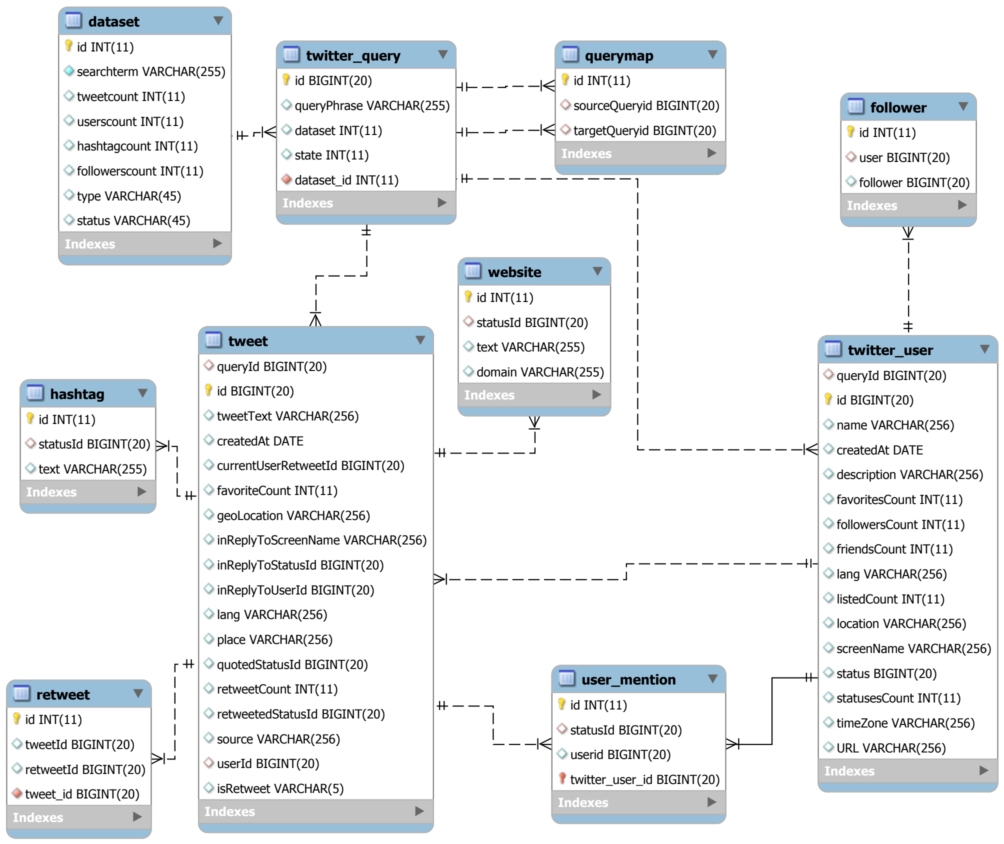


Figure 4.2: The full datamodel used in the application

4.1 Crawling

Twitter offers two APIs to find tweets: The *search* API, which is part of the REST API, and the *stream* API. The search API focuses on relevance of the

result, which means that not all tweets that match the query are returned. The stream API gives access to tweets in real-time.
Behind the implementations of each interface, a scheduler manages the tasks to avoid exceeding rate limits. This is crucial because Twitter offers 15 minute windows to limit traffic. A possible approach to increase program performance would be to circumvent these limits by using more than one account. However, Twitter threatens to suspend access if violation is detected (*"Use a single application API key for multiple use cases or multiple application API keys for the same use case.", developer policy, 6.e.i., May 18, 2015*). When a user assigns a new query to the scheduler, the scheduler manages it by splitting it into one or more of the following tasks:

- **Search news**: Search for news with a given subject.
- **User feed**: Read up to 3200 tweets the specified user has posted.
- **Listen to stream**: Listen to a news stream with a given subject. This captures all tweets within a given time span together with the tweeting user.
- **Followers**: Finding followers is the bottleneck of the application. This is because some users have far over a million followers. With a rate limit of 15 requests per 15 minute window and a maximum response size of 100 items, it takes 6 days 22 hours and 40 minutes to query one million followers.
- **Retweets**: With *GET statuses/retweets/:id*, retweets can be queried. This is not necessary this case because the graph represents a closed system, i.e. only retweets within the graph are sought-after. Each retweeted status contains a field named *retweetedStatusId* (see 4.2), so retweets are queried directly from the database with a simple join operation.
- **Analysis**: Once all operations for a query are completed, this task identifies the most used hashtags and invokes a new search for the specified subjects. This leads to a better coverage of the requested topic.

After all tasks of are finished, edge creation is invoked. This reduces the costs of graph creation.

4.2 Dendrogram

Once all edges are stored, the graph becomes available on the website. With all edges pre-calculated, only user ids and edges are loaded. The clustering algorithm creates a priority queue of each combination of nodes. It then merges node pairs until a pair cannot be merged and starts again. This results in $\mathcal{O}(\log n)$ cycles in the best case, where all cluster pairs are merged, and $\mathcal{O}(n)$ cycles in the worst case, where only one cluster pair is merged each cycle. Each cycle costs $\mathcal{O}(m^2)$ where m is the current amount of sub-clusters left as top nodes, because all sub-clusters have to be compared with each other. As described in Section

3.2, this process is executed twice. There are no mechanisms implemented to detect trolls, but users with no connection to others are excluded from clustering to avoid a random combination of nodes.
Each graph is produced and stored in a container, so more than one graph can be generated at the same time. The container allows access to graph data to query for hashtags and websites associated with a certain cluster.

4.3 Website

The website inherits the structure from the underlying program, containing one tab for each program component. Forms to create datasets, search for subjects, listening to streams and to add a user to a dataset let the user control the crawler. The dataset tab gives access to the data model. From each, a link leads to the graph tab where the different graphs are displayed. To avoid SQL injections, two simple security mechanisms are implemented. Regular expressions are used for queries and prepared statements are used to insert data. More complex security measures exceed the scope of the project.

Chapter 5

Results

The result of the project is an interactive tool in form of a website which lets the user input any current topic and cluster it. The input can either be a live stream or based on a search term. Once the data is crawled, four different kinds of visualizations are available. The following descriptions are based on the subject "#rapefugees" which came up after the New Year's events in the city of Köln.
The first visualization shows a graph representing groups depending on their social surroundings. In this graph, the website user can either click on a user node to see details about the user and see who he is following, or click on clusters to compare their top hashtags and top websites. This graph is the basis of the whole clustering process and the other graphs are derived from this one. Figure 5.2 shows an example of such a graph. Each leaf represents a user. The intermediate nodes indicate two joined clusters. Some of these are coloured green, indicating that the nodes beneath form a cluster as used in the fourth graph.
The second graph contains the same nodes and edges as the first graph, but shows the distribution of topics by colouring each node with the most used hashtag. This gives an idea of what users tweeted about. Figure 5.3 is the result based on the first graph. The algorithm chooses the least used of the top hashtags so that not all leafs are coloured in the colour of the prevalent subject. In the given example, *2A* is a hashtag introduced pretaining the *Second Amendment* and gun rights. *TCOT* stands for *Top Conservatives on Twitter*. In this graph it becomes visible that some groups contain a predominant hashtag while others have a combination of all hashtags.
The third graph, as shown in Figure 5.4, is the most important one considering the problem statement. It is the result of reclustering the data, but this time based on discussed subjects. This allows conclusions about the filter bubble in which each user is trapped. Each user keeps the colour from the first graph mentioned, but is put into a new cluster. From the colour of the surrounding graphs, conclusions can be drawn about who each user should follow to get the widest range of information about the subjects of his interests. The better this graph is mixed, the better are groups with different opinions connected with each other.
The fourth graph displays all hashtags used in the tweets of the clustered users. The cluster colours are inherited from the first graph as described in Section 3.2.

The resulting graph gives an impression of the intended impact of each hashtag. Hashtags closer to the middle of the graph are more generally used within the dataset than those close to the sides. Note that some hashtags appear more than once in the graph because these are used in more than one cluster. Such a graph is displayed in Figure 5.5. The colours of the nodes are inherited from the clusters in Figure 5.2.

Contained in the website are two subjects that are already crawled. More subjects can be added either to the existing datasets or to a new dataset with the forms on the *create* page of the website.

5.1 Swiss Votation

On February 28, 2016, Switzerland voted about four voting proposals. I captured the whole traffic through the Streaming API. On the website, the dataset is split into two subsets: The *Durchsetzungsinitiative* and the overall data.

5.2 American Elections

During the TV debates mid-December 2015, a lot of tweets with the hashtag *election2016* were posted. These, combined with users who publicly endorse the presidential candidates, were used as a ground truth.

These endorsers build a simple ground truth because we know in advance who they endorse, so users who endorse the same candidate should lie close together in the resulting graph. After the crawling, I applied the clustering algorithm to the data. In the resulting graph, all users except the endorsers as well as all intermediate nodes were removed from the graph, so the resulting graph only contains the endorser nodes, as shown in Figure 5.1. The colours of the nodes indicate who they endorse. The result shows that the clustering works to a certain degree, as a majority of same coloured nodes lie close to each other. In Section 3.3, reasons are given why a better clustering is not possible with the given algorithm.
The example shows that clustering Twitter data based only on meta data is possible. With the procedure described in Section 3.2.2, clustering works for any given input. This makes it possible to detect filter bubbles and find the most discussed themes within it.
Aside from being used as ground truth, the dataset brings some interesting other results. Figure 5.6 shows the result of clustering the dataset based on followers. Single nodes are omitted in the graph. In Figure 5.7, three nodes are highlighted. Of the two big clusters, the one where the node is coloured orange has *Trump*, *Trump2016* and *job* as dominant hashtags, the other, coloured in red, contains

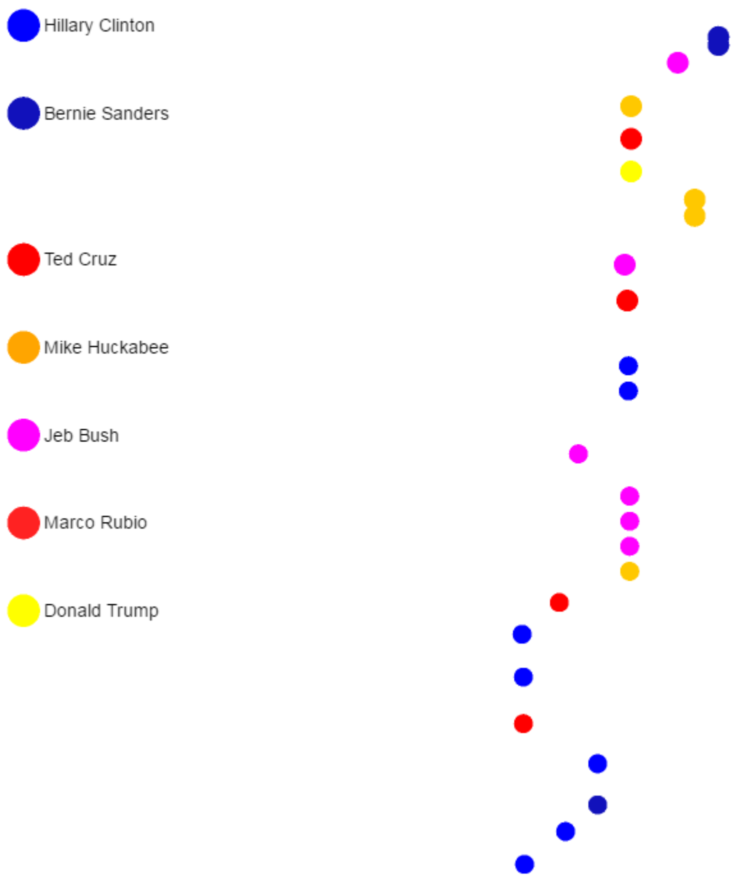


Figure 5.1: Endorsements

FeelTheBern, *immigration* and *climatechange* in the top 10. Out of 133 appearances of the tag *Trump*, 111 are in the cluster containing half the nodes. A subcluster (green) of it containing 27 nodes has 68 out of 206 overall mentions of the tag *Trump16*.

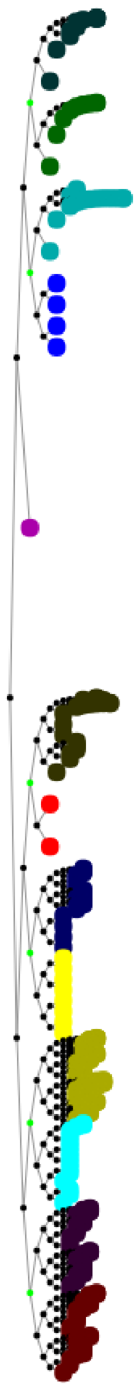


Figure 5.2: Data clustered based on user's followers, nodes below green intermediate nodes form a cluster

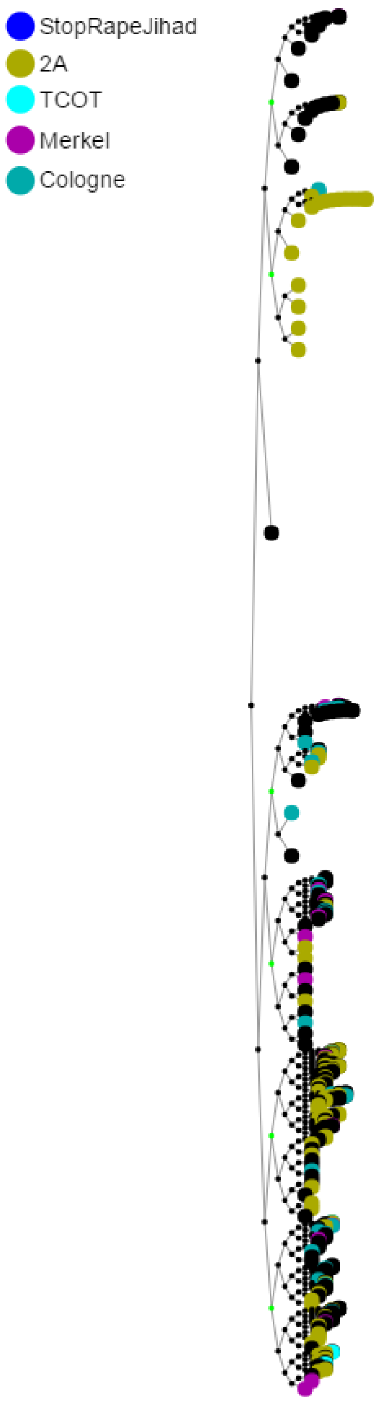


Figure 5.3: Data clustered based on user's followers, colours based on most common hashtags

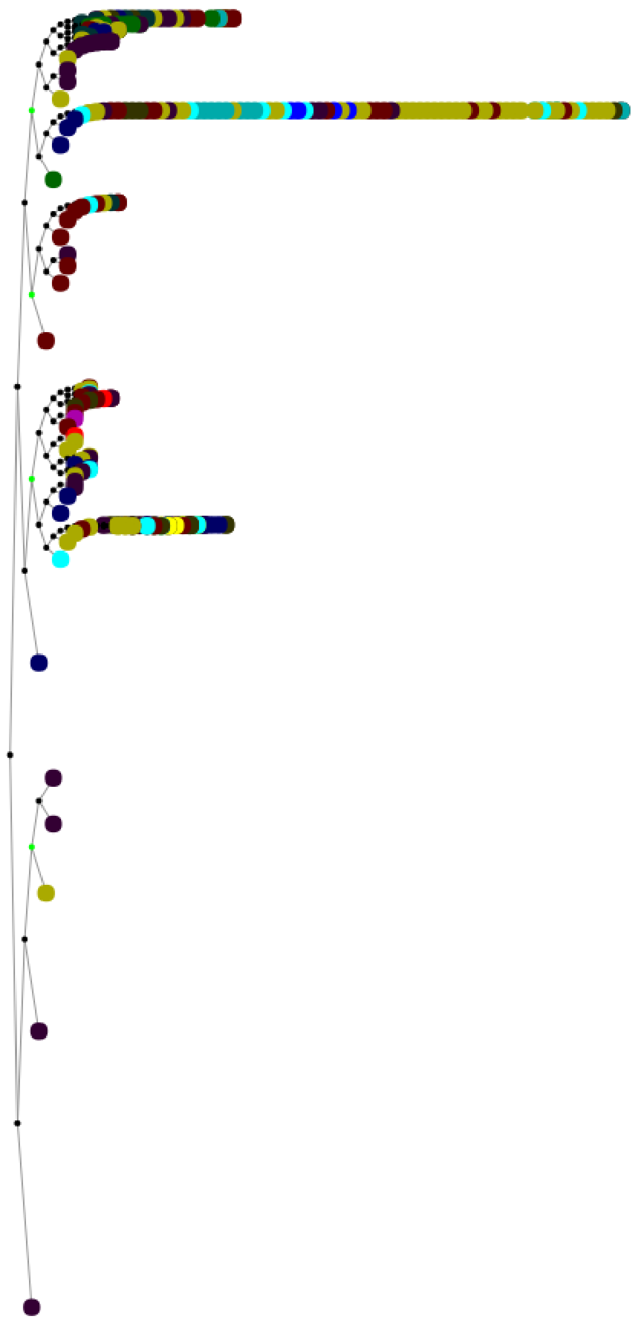


Figure 5.4: Data clustered based on hashtags used by users, colours based on user's followers

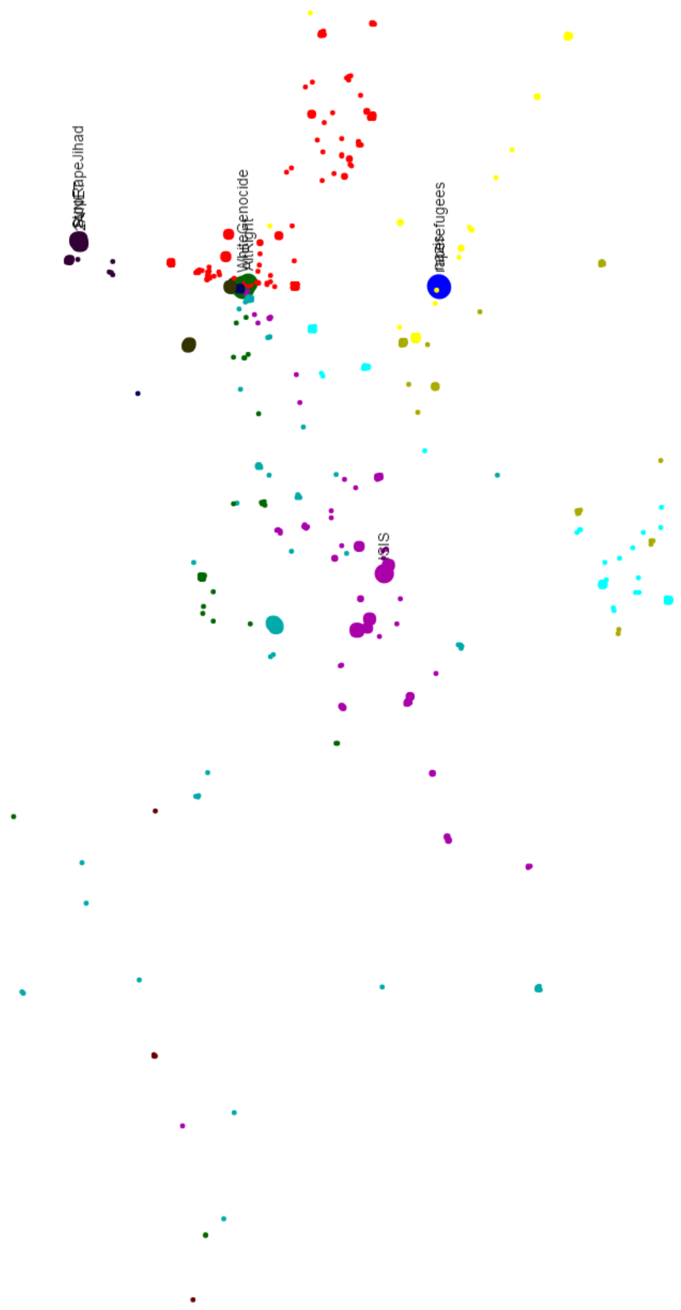


Figure 5.5: Hashtags plotted based on connections between users, hashtags close to the middle of the image have more connections to other nodes. Size depends on frequency of hashtag (roatated 90° counterclockwise)

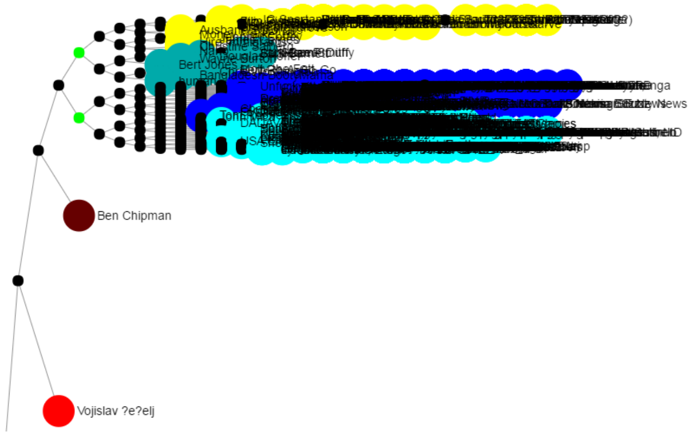


Figure 5.6: Cluster of the search results for hashtag *election2016*

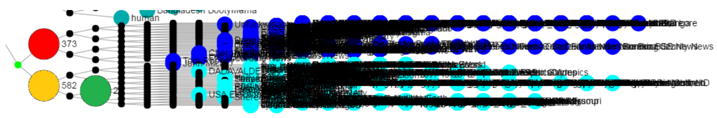


Figure 5.7: Cluster of the search results for hashtag *election2016* with highlighted cluster roots

CHAPTER 6

Conclusions

Detecting filter bubbles is possible up to a certain degree solely based on the user's meta data. Yet it is difficult because content is produced by humans, so it has a lot of irregularities in it and is unpredictable as well as unreliable. The tool described in this paper manages to cluster data and visualize filter bubbles in a simple way, which gives a good basis for future work. The tool gives an impression of how good or bad informed the users are (based on their followers), which user groups (based on their used hashtags) prefer what websites and how hashtags are distributed within a subject. Yet it leaves the question open whether this is good or bad. It's up to the reader to decide whether a user should follow someone or not. Before extending the tool, some basic questions about the impact of filter bubbles have to be answered, and especially, whether it could even be appropriate in certain situations.
The concepts introduced allow clustering the data, but leave open the question of the result's quality. On the other hand, it showed with the example of the Twitter API that the huge amount of social data present in the World Wide Web is not as intransparent as it appears to be and that it is possible to evade filter bubbles and user fitted content using exclusively the supplier's own data.

6.1 Future Work

To continue with the analysis and clustering of the data, a good measurement of the results has to be defined. The described ground truth suffices to prove the functionality of the clustering, but does not allow a universal proposition. Further, a better way of quantifying and visualizing the results has to be found to allow a user to draw conclusions quickly and in a more obvious way. With the algorithm as a basis and the stated enhancements implemented, a tool could be created which analyses the user's behaviour on Twitter, or even on other social platforms, and gives better suggestions about who he should follow.
A special application of the tool would be an extension with which datasets like the *Swiss Votation* set can be clustered in multiple passages to detect groups who have a common opinion about each of the submitted voting proposals.

Bibliography

[1] Pariser, E.: The Filter Bubble: What the Internet Is Hiding from You. Penguin Press HC, The (2011)

[2] : Invisible sieve. http://www.economist.com/node/18894910?story_id=18894910&fsrc=rss Accessed: 2016-04-01.

[3] : Personalized search and its discontents. https://blog.nus.edu.sg/is1103grp203/2013/03/31/personalized-search-and-its-discontents-ii/ Accessed: 2016-04-01.

[4] : Poultry markets: on the underground economy of twitter followers. http://dl.acm.org/citation.cfm?id=2342551 Accessed: 2016-04-02.

[5] : Internet live stats. http://www.internetlivestats.com/one-second/ Accessed: 2016-04-02.

[6] : Israel, gaza, war & data. https://medium.com/i-data/israel-gaza-war-data-a54969aeb23e#.3159sab1p Accessed: 2016-04-05.

[7] : Twitter study - august 2009. https://web.archive.org/web/20110715062407/www.pearanalytics.com/blog/wp-content/uploads/2010/05/Twitter-Study-August-2009.pdf Accessed 2016-04-06.