# Facebook Network Analysis of News Pages

Project report for *Introduction to Digital Humanities*


Team members:

| | | |
|---|---|---|
| Burfeind, Thees | Matrikelnummer: | 3700947 |
| Eberling, Christian | Matrikelnummer: | 3718725 |
| Herre, Donatus | Matrikelnummer: | 3744292 |
| Linker, Felix | Matrikelnummer: | 3758276 |
| Nüßler, Jana | Matrikelnummer: | 3676246 |
| Ruschke, Tim | Matrikelnummer: | 3714699 |
| Stengel, Samuel | Matrikelnummer: | 3739949 |

# Abstract

Motivated by the idea of effects social media can have on groups and their political understanding, our aim is to examine user activity on public Facebook news pages in the German speaking area. We thereby want to examine whether there are any communities inside Facebook's social graph that can be clearly distinguished from one another. It is a necessary condition for communities of this type to exist if terms like "post-truth", "fake news" and "filter bubble" describe actual conditions that are relevant to our society.

Using Facebook's Graph API, we built our own graph reflecting Facebook's social graph structure. Abstracting from this structure allowed us to see that there are, in fact, communities of pages which can be associated with specific ideologies.

# Introduction

In the recent months, the interplay of politics, news and social media has been a highly discussed topic. One of the indications for this phenomena is the noticeable trending of buzzwords like “post-truth”, “fake news” and “filter bubble”, exemplifying the critical potential emerging from the interaction of politics, news and social media.
Whether it be the suspicion to be trapped in a “filter bubble”, losing sight of prevalent political developments and changes. Or the conversation partner and his inflationary use of “fake news”, a term omnipresent whenever the recently elected American president visits press conferences. Or whether it be the Oxford word of the year “post-truth”, expressing a fundamental doubt of objectivity in conventional news sources and political actors.
What connects these definitions are the deep trenches they narrate, that dig deeper into the fabric of segregated political groups, users of social media and news editorials. A sociopolitical background fraught with tension that inspired us to examine the relation of politics and news in the german speaking facebook communities.
Particular attention will be paid to user activity in relation to news distribution. The data surveyed for the purpose of analysis should enable us to analyse networks and point out connections between users, parties and news. In reference to the aforementioned imagery of trenches, we formulate the following questions: Are there clear divisions between political camps within the German Facebook community? Or will we find bridges connecting different parties? How do users, pages and posts relate to one another in terms of distribution and political bias?
Following these questions, we found no research regarding German speaking Facebook communities and initiated our own project. We begin talking about the technical background by introducing Facebook as well as its Graph API, followed by our main scripting language Python and finish the technicalities elaborating on our database in ArangoDB. The following chapter deals with the methodology and concepts of our data collection process, starting with the crawler and one of its key components, the entry point supplier. After going into the theory of our data structures and utilised algorithms of this project as well as the processes of data cleaning required to create insightful graphs, we eventually present our data visualisation. Finally, we conclude with a summary of our findings and an outlook regarding future research.

# Composing a Social Graph

## Technical Background

### Facebook

Facebook is a social media and networking platform allowing individuals and groups to create public profiles. The service is commonly used by private individuals to keep in touch with friends and source media related content. Private users extend their social network by sending ‘friend requests’ to other individuals, who can reciprocate and, depending on the

privacy setting of the user, only then acquire full access to their profile details, photo albums and other information provided by the users themselves. Individuals can express their interest through likes, reactions, shares and comments, but can also create content through ‘posts’, which can be liked, reacted to, shared or commented on by others.
Facebook also appeals to companies, political parties and other corporate entities as well as public figures due to its potential to reach more audiences through advertising but also through ‘pages’, a type of profile used to represent one’s public identity and generate content through ‘posts’. To access page information, friend requests are not required; instead, users are offered to ‘like’ them.
Being the biggest social network in Germany with over 28 million users[1] of which many varying political and media institutions are part of, as well as providing comprehensible access to its user interaction data through its Graph API, Facebook lent itself as an attractive network to examine.

## Graph API

Facebook provides a REST-based interface known as Graph-API used to send and receive data between facebook and the API user. Depending on the request sent to the API, a JSON file can be returned, consisting of nodes, edges or fields meeting the criteria specified in the request.

Returned elements consisted of nodes (e.g. users, pages or posts) and edges connecting related nodes (e.g. user comment on post). Fields represent the content of the nodes (e.g. name of a page). The Graph API offered itself because it supplied enough viable data to support our research and constituted the most convenient option.

## Python

We decided to use Python 3.6 as our main programming language since it was easy enough to learn and somewhat intuitive. It supported all external plugins we required for our project and allowed compatibility with ArangoDB supported through a driver. Being an interpreted programming language, it also made simple real time analysis of data possible. JSON files supplied by the Facebook Graph API were easily converted into dict objects without requiring further coding.

## ArangoDB

To store the data from our crawling process we decided to use a database system, as it allowed us to save our data in a very abstract form ideal for analysis. Any file based system would leave us with a huge overhead for the amount of data we expected and would render us with the need to implement even the most basic functionalities to inspect our data.
ArangoDB is a multi-model database that stores data in key-value json documents. It allows the user to differentiate between document collections and edge collection whereas both store json documents. Document collections store the raw data while edge collections store

[1] https://buggisch.wordpress.com/2016/01/04/social-media-nutzerzahlen-in-deutschland-2016/ visited at 25th of february 2017

links between documents of possibly different document collections. With edge collections it is possible to define graphs on our document collections. ArangoDB comes with a web-interface to add, inspect and maintain data and queries. It also includes a query language making graph traversals as well as more simple queries possible, along with a REST API compatible with ArangoDB drivers for several programming languages - including python.
ArangoDB allowed us to store results from our crawling process with more or less no additional processing since the Graph API returns json documents which can directly be stored. The additional graph functionality afforded us to define a data structure for our crawling results closely resembling the structure we needed to work with as defined by the Graph API's specifications. Factoring in ArangoDB's easy to use and yet powerful query language, it provided a reasonably solid basis for our data maintenance .

# Methodology

This section covers our attempts and methods used while crawling facebook and transforming crawled results into a graph for analysis.
It is divided into four sub-sections: *Selecting Points of Entry* covers our approach to getting initial data to base our crawling upon; *Crawling* covers our actual crawling algorithm; *Cleaning* explains the process of filtering our crawling data; and finally, *Selection* will explain how we came to a graph that allows us to draw conclusions from .

## Selecting Points of Entry

Our goal when selecting entry points was to generate a list of facebook pages to be used in our crawling algorithm as starting points.

### Initial Attempt

We initially intended to use the search function provided by the Facebook Graph API, allowing us to search for specific profiles through the use of keywords, in our case pages. After we set up a script searching for media related keywords (such as "*nachrichten"*, "*zeitung"*, "*presse"...*), the retrieved data was then filtered by the corresponding categories for each page. We decided on taking pages into account only when categorized as magazine, newspaper, media company and news website. Finally, we filtered the results based on location data provided by the pages, only considering pages located in Germany, Austria and Switzerland.

The approach lead to sub par results, primarily due to low hit rates: Pages originally part of the search result ended up missing, while irrelevant data such as non German based pages and wrong categories were returned, which altogether did not satisfy our expectation.

### Final Attempt

In our second and final attempt, we decided to use parties likely to be present in the next German parliament[2], namely the CDU, CSU, SPD, Grüne, FDP, Die Linke and AFD. By selecting these parties, we wanted to cover a broad spectrum of opinions and news pages. We observed that the official facebook pages of parties were actively sharing news articles and commenting on recent incidents, making them eligible to use as pre-initial starting points. Each party's facebook page feed was inspected with an algorithm, creating an initial entry-points-list through the following method:

**Input:** A list of facebook pages $P$
**Output:** A set of facebook pages $P'$ initialized as an empty set
**Procedure:**
For each page $p \in P$ :

[2] http://www.infratest-dimap.de/umfragen-analysen/bundesweit/sonntagsfrage/ visited at 27th of february 2017

For each post $x$ that is on the feed of $p$ and created since the first of January 2017:

If $x$ shares a post of a page $p'$, add $p'$ to $P'$

If $x$ is authored by a page $p'$, add $p'$ to $P'$

For each page $p'$ that is liked by $p$:

Add $p'$ to $P'$

A manual filtering by our team members followed, including only pages that reflect, comment or report recent happenings of public interest. Therefore, we inspected the page's corresponding feed and decided on a case by case basis whether the given page qualified as relevant.

The resulting pages made up our entry-points-list of pages.

# Crawling

## Structure of the Results

Our crawling algorithm is based on a multi partite, directed graph realized in ArangoDB. This section will not go further into the specific implementation details but rather give formalisms that reflect implementation in detail.

The graph resulting from our crawling algorithm consists of four disjunct sets of nodes:

- $U$ - the users
- $N$ - the pages or news pages
- $P$ - the posts
- $C$ - the comments

It also consists of five types of edges:

- $comments \subseteq C \times P$ - the comments-on relation; an edge $(x,y) \in c$ means that the comment $x$ was written on the post $y$.
- $authors \subseteq (U \cup N) \times (P \cup C)$ - the authors relation; an edge $(x,y) \in a$ means that the user or page $x$ has written the post or comment $y$.
- $shares \subseteq P \times P$ - the shares relation; an edge $(x,y) \in s$ means that the post $x$ shares post $y$.
- $reacts \subseteq (U \cup N) \times P$ - the reacts relation; an edge $(x,y) \in r$ means that the user or page $x$ reacted to the post $y$.
- $on \subseteq P \times (U \cup N)$ - the is-on relation; an edge $(x,y) \in o$ means that the post $x$ is on the timeline of the user or page $y$.

Finally, our resulting directed graph can be described as $G = (V, E)$ with $V = U \cup N \cup P \cup C$ and $E = comments \cup authors \cup shares \cup reacts \cup on$.

This structure does not make full use of the possibilities provided by the Facebook Graph API. For instance, comment tags,which are links allowing users to mention other users in comments, often used to inform one's' friends about posts they find interesting, were not included. Reactions to comments and comments on comments were also excluded, due to lack of resources to manage the crawling data set size. Additionally, we found that users being linked by other users in a comment is no sign of user activity for the linked user. And lastly, crawling reactions and comments on other comments ended up being inefficient for our purposes, since we observed that comments and reactions to comments didn't add new users to the field for the most part. Facebook's Graph API permitted a limited number of API calls, so crawling these sub-structures in the facebook network was ruled out.

## Crawling Algorithm

The crawling algorithm initializes the resulting graph $G$ as described in the previous section with only the set of the pages being initialized to the pages in the entry-points-list. That means that there are no users, post or comments and no edges in the graph.

Note that whenever we talk about crawling posts and comments in the following procedures, we also retrieve their respective authors. This is left out to increase ease of reading since it is does not contribute to the understanding of our algorithmic procedures.

At first we crawled every post of every page that was present in our entry-points-list to a set date. We tried several dates but ended up considering only posts of the last ten days relative to the day we ran our crawler so to allow us to handle the amount of data returned.

In a second step we crawled posts that took part in the share relation by using following procedure:

**Procedure:**

Do:

For each post $p \in P$ :

$shared \leftarrow$ query the post that is shared by $p$

If $shared \neq NULL$ :

$P = P \cup \{shared\}$

$shares = shares \cup \{(p, shared)\}$

$s \leftarrow$ query all posts that share $p$

For each post $p' \in s$ :

$P = P \cup \{p'\}$

$shares = shares \cup \{(p', p)\}$

Until no further changes to $P$ occur

In a third step we crawled comments and reactions to posts present in the graph by using following two procedures:

**Procedure:**

$p \leftarrow$ randomly select a post that has no comments crawled in $P$
$c \leftarrow$ query all comments of $p$
For each comment $x \in c$:
$C = C \cup \{x\}$
$comments = comments \cup \{(x, p)\}$
Mark $p$ as having all comments crawled

**Procedure:**
$p \leftarrow$ randomly select a post that has no reactions crawled in $P$
$r \leftarrow$ query all users that have reacted to $p$
For each user $x \in r$:
$U = U \cup \{x\}$
$reacts = reacts \cup \{(x, p)\}$
Mark $p$ as having all reactions crawled

We acquired approximately 1.2 million users through repeatedly executing the aforementioned procedures for our database.

# Cleaning

Since we selected posts randomly - but evenly - to crawl their comments and reactions we ended up with a big amount of posts not having comments or reactions crawled. In the data cleaning step, we eliminated all those posts from our graph. Every post that did not have all comments or all reactions crawled was deleted. Consequently, we also deleted all comments and reactions to those posts as well as all users and pages that ended up with no activity in the graph.

After cleaning our data set, we ended up with following collection sizes:

| $\|U\|$ | $\|N\|$ | $\|P\|$ | $\|C\|$ | $\|reacts\|$ |
|---|---|---|---|---|
| $962,917$ | $3,082$ | $7,674$ | $183,227$ | $1,436,887$ |

# Selection

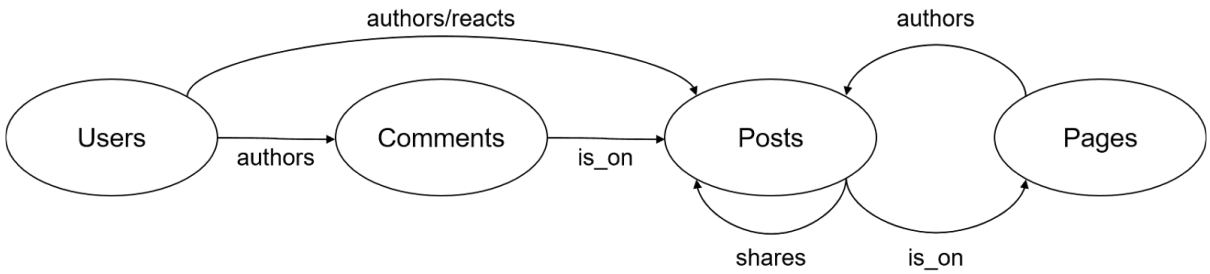
By selecting the data we wanted to analyse, we created two new graphs, each of which will be outlined in the next two subsections.

## User Graph

The user graph links users to pages by their activity. Through this, we wanted to inspect the connections that can be found between users and pages. The user graph was created by taking specific paths in our crawling results graph $G$ and combining them to a new directed graph $UG = (V, E, c)$ with $V$ / $E$ being nodes/edges and $c : E \rightarrow \mathbb{N}$ as weighting function for the user graph graph.

Our goal when creating the user graph was to capture both the pages users were generally connected to but also to consider a weighting for user activity. We considered a user as being connected to a page $p$ when he had authored, reacted to or commented on a post that was on $p$'s timeline, was authored by $p$, shared a post on $p$'s timeline, or shared a post authored by $p$. The weight of an edge was determined by counting a user's connections to the page that is connected to the user by the respective edge.

The paths in our crawling result graph we thereby took are illustrated below:




Having an initial user graph $UG = (U \cup N, \varnothing, c)$ with $c$ mapping each element in its range to $0$ that only consists of all users and pages in the crawling graph it's construction is described more formally by following two procedures:

**Procedure:**

For each user $u \in U$:

For each comment $c \in C$ with $(u, c) \in authors$:

$p \leftarrow$ get the post that $c$ commented on

$alterEdges(u, p)$

For each post $p \in P$ with $(u, p) \in authors$:

$alterEdges(u, p)$

For each post $p \in P$ with $(u, p) \in reacts$:

$alterEdges(u, p)$

**Procedure** $alterEdges$:

**Input:** A user $u$ and a post $p$

$pages \leftarrow \varnothing$

If there is a page $p' \in N$ with $(p', p) \in authors$:

$pages = pages \cup \{p'\}$

If there is a page $p' \in N$ with $(p, p') \in on$:

$pages = pages \cup \{p'\}$

For each page $p' \in pages$:

$E = E \cup \{(u, p')\}$

Increment $c((u, p'))$ by $1$

$\overline{p} \leftarrow$ get the post that is shared by $p$ at the topmost level, i. e. the post the is the supremum to $p$ in the $shares$ relation

If $\overline{p} \neq NULL$ :

$alterEdges(u, \overline{p})$

Note that this procedure won't end up having multiple edges between a user and a page.

## Page Graph

To further analyse connections of users to multiple pages, we decided to further simplify the graph structure. Therefore, we created the non-directed page graph $NG = (V, E, c')$ with $V/E$ being nodes/edges and $c' : E \to \mathbb{N}$ being a weighting function for the edges. This graph combines edges and user nodes in $UG$ to edges in $NG$. For every user node in $UG$, we inspect all of its edges which connect to pages and iterate pairwise over them. Each pair of edges is composed to a new edge with its weight being the sum of the both edges weight.

Initially, the page graph's node set $V$ contains all pages in $N$ whereas $E$ is the empty set and $c$ maps every element in its range to $0$:

**Procedure:**

For each user $u \in U$:

$e \leftarrow$ get all edges outgoing from $u$ in $UG$

For each pair $(e_1, e_2)$ of edges in $e$:

$p_1 \leftarrow$ get the page that is connected to $u$ by $e_1$

$p_2 \leftarrow$ get the page that is connected to $u$ by $e_2$

If $p_1 \neq p_2$ :

$E = E \cup \{\{p_1, p_2\}\}$

Increment $c'(\{p_1, p_2\})$ by the value of $c(e_1) + c(e_2)$ in $UG$

# Findings

## Visualisation Methods

### Gephi

In need for a visualisation tool able to process large amounts of data into well structured and manageable graphs, our decision fell for Gephi, an open source software designed for network analysis and visualisation. It was important to us that clustering, graph coloring and some analysis tools were featured. Gephi provided an adequately shallow learning curve and required only fundamental understanding as a starting point.

### ForceAtlas2

ForceAtlas2 is a force directed layout available in Gephi and was used to arrange spatial positioning for our graph. Edges attract the nodes they are connected to, while nodes repel one another. Thereby, a movement is created that slowly approaches a balanced state where nodes are near standstill.
Graphs do not always converge to the same end-configuration. The result depends on the force value selected, the initial state and the approximation of the algorithm.
We decided on using ForceAtlas2 for visualisation of our graphs because it was fast on the one hand, and provided solid results on the other hand. Other layout algorithms that were available to us did either not alter the layout in meaningful ways, or were too slow.

## User Graph Visualisations

The visualisation properties were modified based on following criteria:

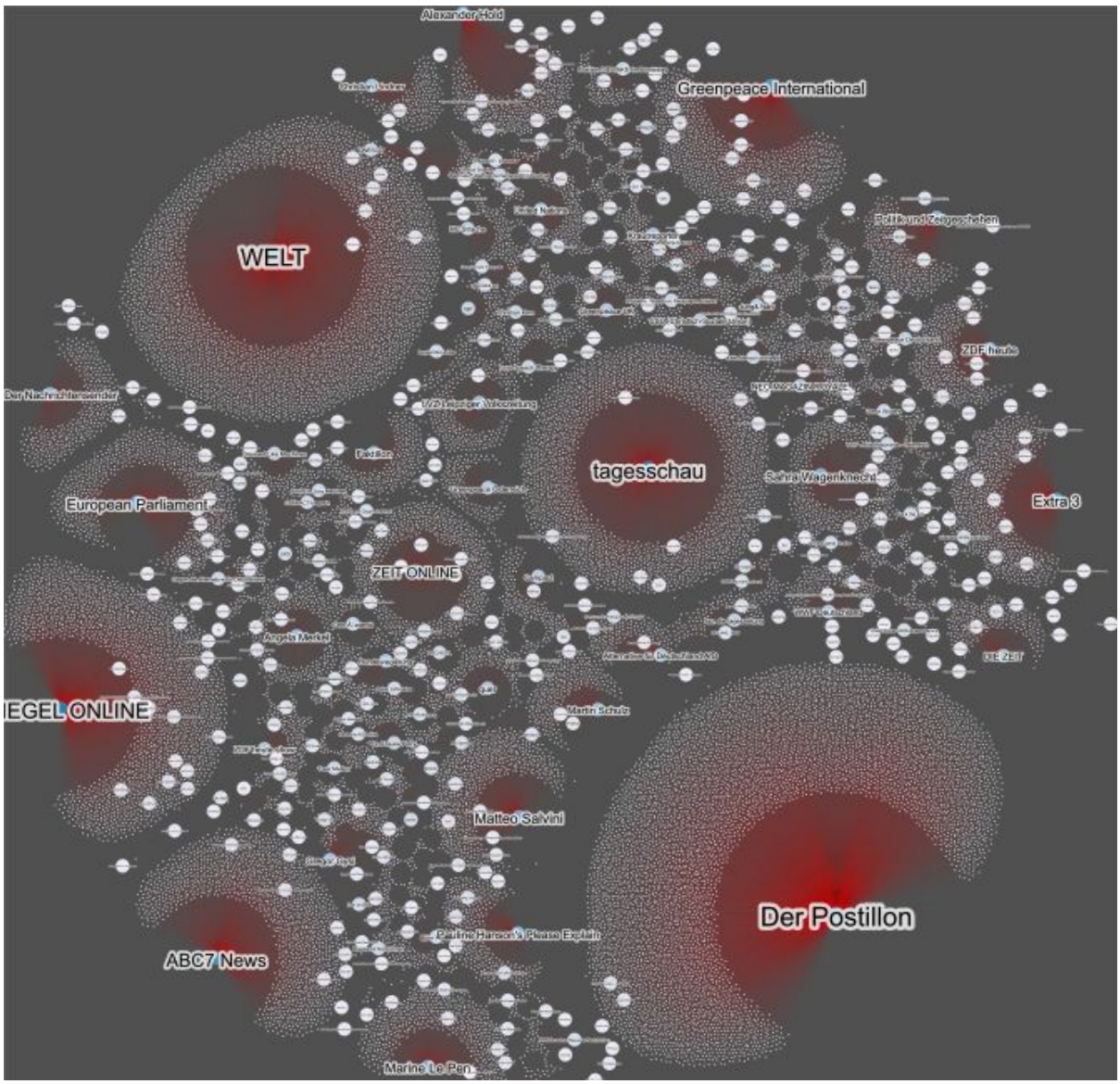
Nodes were sized according to their respective type: User nodes were kept much smaller than page nodes in order to differentiate them.
Nodes were colored by their degree on a scale from white to blue. A darker color signifies a higher degree of activity. The labels of the page nodes are sized according to their degree of activity as well. General layout was determined by applying ForceAtlas2.

Edges were sized and colored according to their weight. Edges were colored on a scale from white to red. A darker color and a bigger size means a higher weight. The weight of an edge reflects the respective user's activity on the connected page.

Note that each scale is only relative to the values that are provided in each sub-graph. For example: If the weight of every edge is one, then one is also the highest degree leading to all edges having the color value that is highest on the scale.

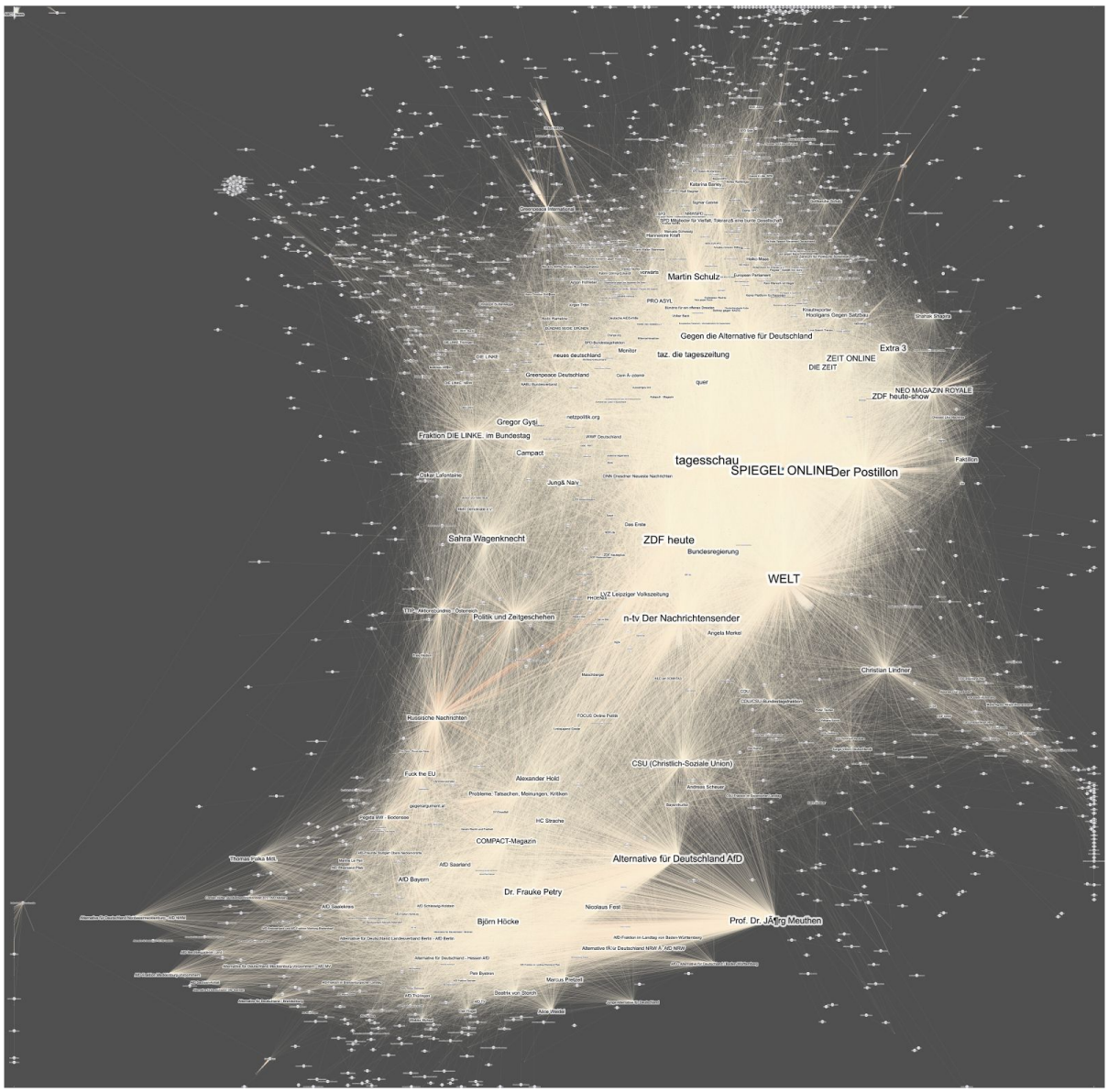
We decided to visualise two scenarios: the most active users and the least active users. With these two scenarios, we tried to observe both extremes of user interaction. It was not possible for us to visualise the user graph as a whole since it contained approximately one million users. The hardware available to us wasn't capable of visualising such amounts of data. We tried however to analyse random subgraphs of the user graph but this didn't show anything that wasn't already included in the pages graph.

## Least Active Users




All pages were included in the graph along with the 30.000 least active users. A user qualified as being least active if he was under those who had the fewest edges in the user graph.
All users share a nearly equal degree of connections, with some exceptions being connected to more than only one page. All edges share the same weight, implying nearly identical interaction with their respective page.
However, differences can be made out in the number of connections per page, apparent through the varying sizes of page nodes. News and satire pages have the most users, followed by public individuals, primarily politicians. It's worth noting that political parties play much less of a role, with a few exceptions being the European parliament and Greenpeace.

## Most Active Users



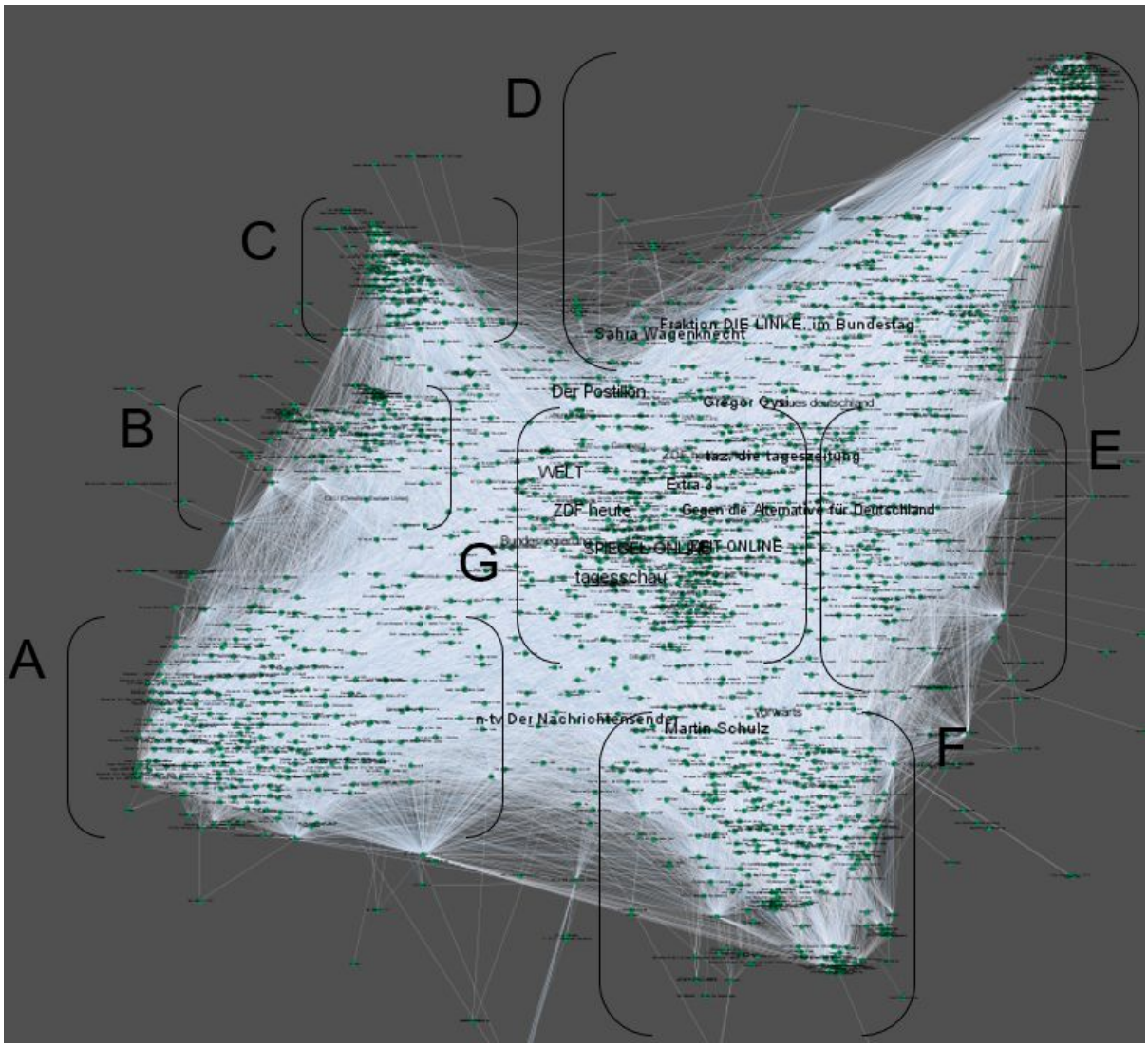

All pages were included in the graph along with the 30.000  most active users. A user qualified as being most active if he was under those with the most edges outgoing in the user graph.
Coloration appears much weaker due to the distribution of interactions of users. Although some  users appear very active, most users seem to be relatively inactive. We can observe that news pages attract most very active users, as also observed in our previous graph.
In most cases, pages belonging to political parties and organizations associated with them appear closer to each other. We can assume that it is due to the user's inclination to visit pages with similar content or orientation. It is worth noting that single members of political parties such as Sarah Wagenknecht, Christian Lindner or Martin Schulz show more connections than the parties they are associated with. Another observation can be made about the divide between two more or less distinguishable major groups drifting apart, with most parties connected to one another on one side, with the AfD and its associated pages

appearing separated. CDU and CSU have a relatively small number of user connections, appearing less bright than other groups. Party member and chancellor Angela Merkel’s page is positioned closer to the media outlets within the big cluster than her own party.

# Page Graph

The page graph was visualised using ForceAtlas2 without considering the edge weight. Our analysis showed that 76.11% of all edges had a weight smaller than ten and 91.54% of all edges had a weight smaller than 50. There were, however, very few strongly weighted edges which biased the visual representation of the graph heavily and is the reason why we decided to exclude the edge weight from ForceAtlas2.

The edge weight was also visualised using a white to blue scale where white meant the edge had the smallest possible weight and blue meant the edge had the highest possible weight. The above mentioned ratio of edges with small weight to edges with high weight can also be seen clearly in our visualisation where almost all edges are slightly blue and some hardly noticeable edges hide in between.



By visual analysis of our graph, we found out that you could clearly divide it into seven areas which are highlighted with brackets in the picture.
Area A contains pages that mainly belong to the AFD or position themselves critically towards refugees and immigration in Germany.
Area B contains pages that belong to the the parties CDU/CSU.
Area C contains pages that belong to the party FDP.
Area D contains pages that belong to the party Die Linke.
Area E contains politically left news pages,positioning themselves against other groups, such as *Blumen statt Hass* (flowers rather than hate), *Bottrop gegen NAZIS* (Bottrop against NAZIS) and *No Hate Speech Movement Deutschland*; some of which have a humorous side to themselves.
Area F contains pages that belong to the party SPD.
And Area G contains general news pages and pages that belong to the party Die Grünen.

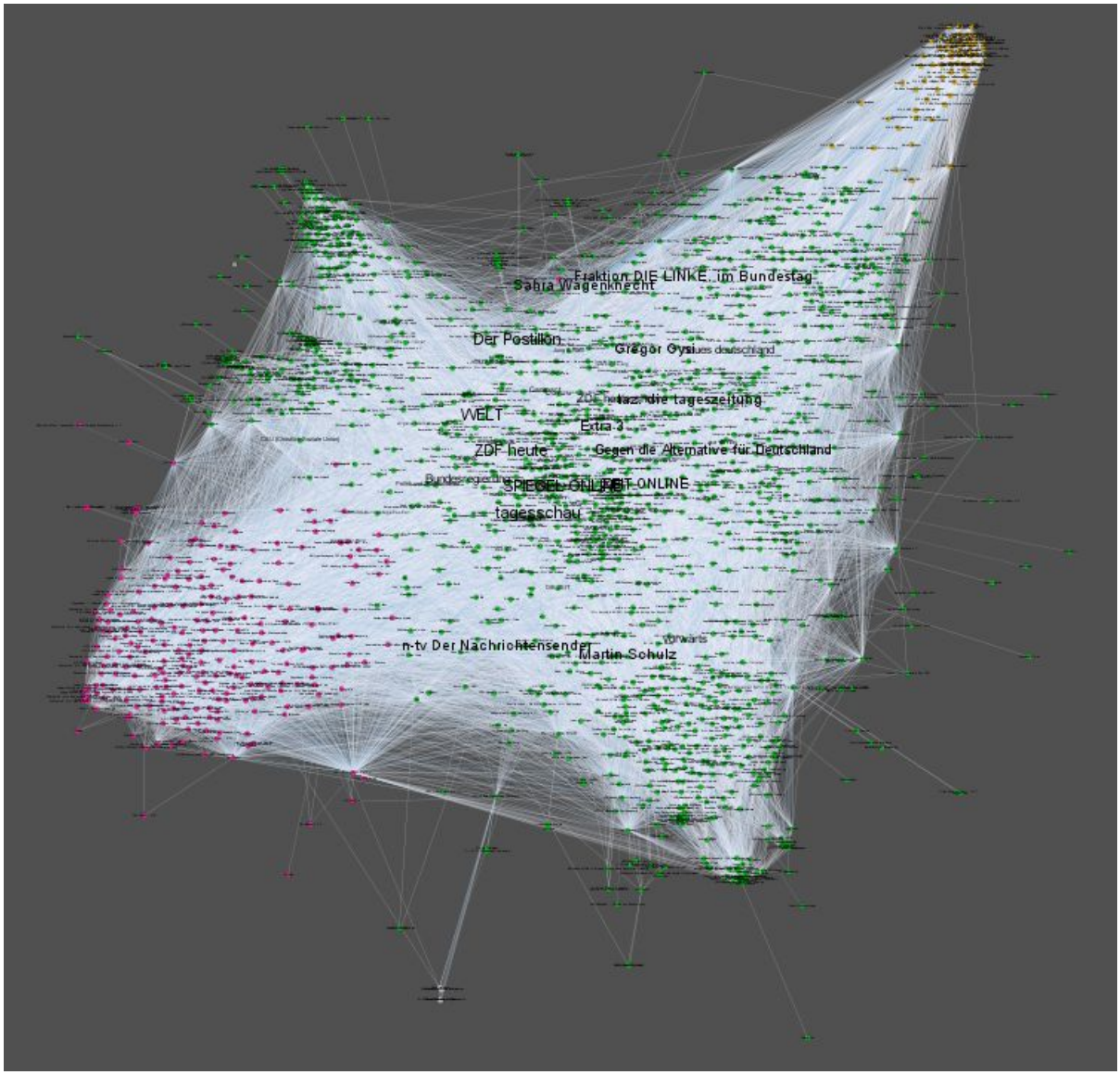
You can see that the areas A, B, C, D and F have most of their edges going towards the center but fewer connecting each of them. The nodes of these areas that are more towards the center of the graph mostly correspond to famous politicians that belong to the parties, such as Sarah Wagenknecht, Martin Schulz or Angela Merkel.

In the next step, we applied a community algorithm to the graph. We did this with different values for the community resolution which influences the size of the communities. A higher value for resolution means fewer but bigger communities whereas a lower value for resolution means more but smaller communities. The default value for resolution is 1.

In the first step, we set the resolution to 2 to isolate communities that are connected in general.Next, we set the resolution to 1 to isolate the “normal” communities. Lastly, we set the resolution to 0.8 in order to isolate the stronger connected communities.

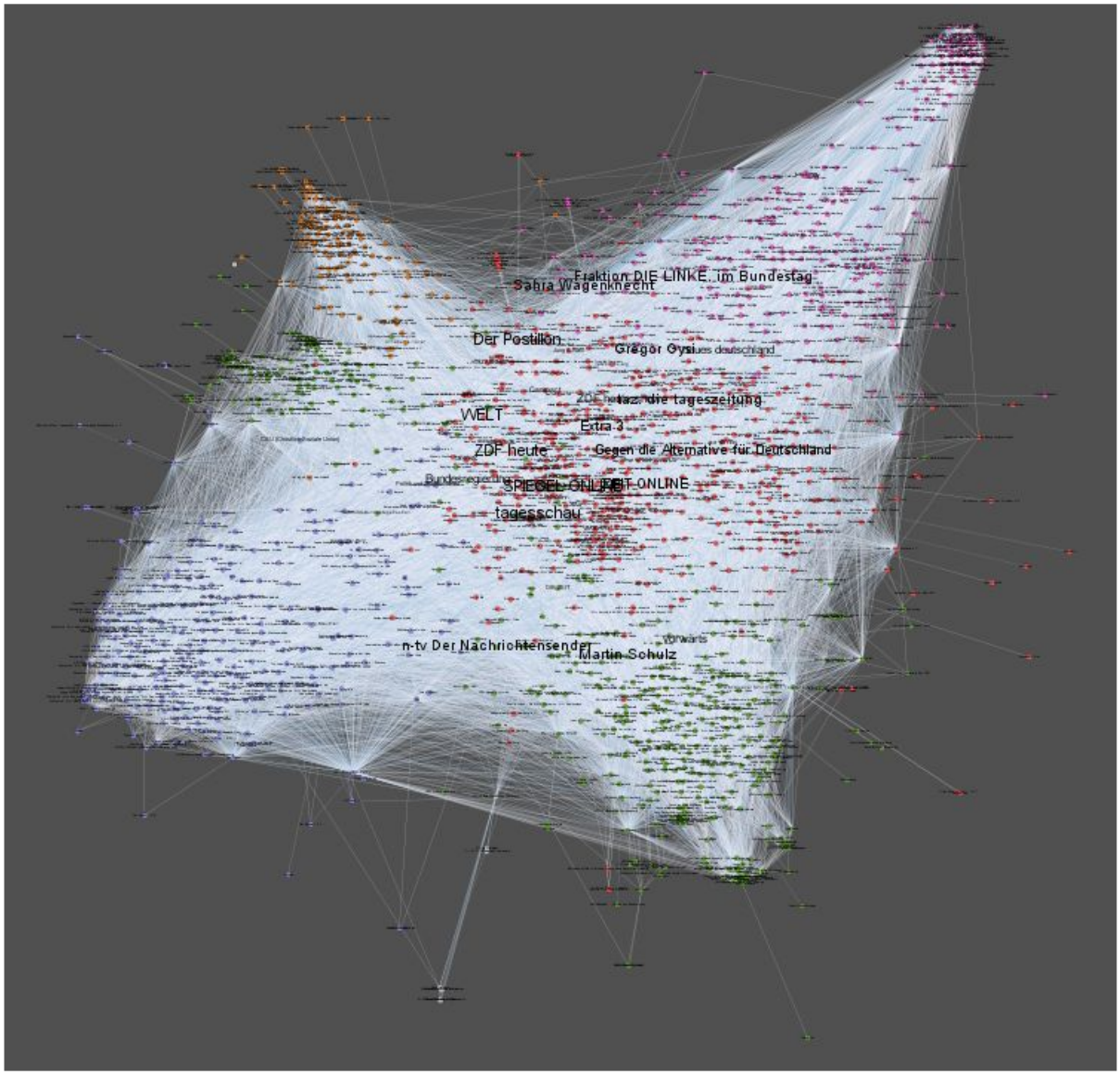
## Page Graph Communities

### High Resolution Communities



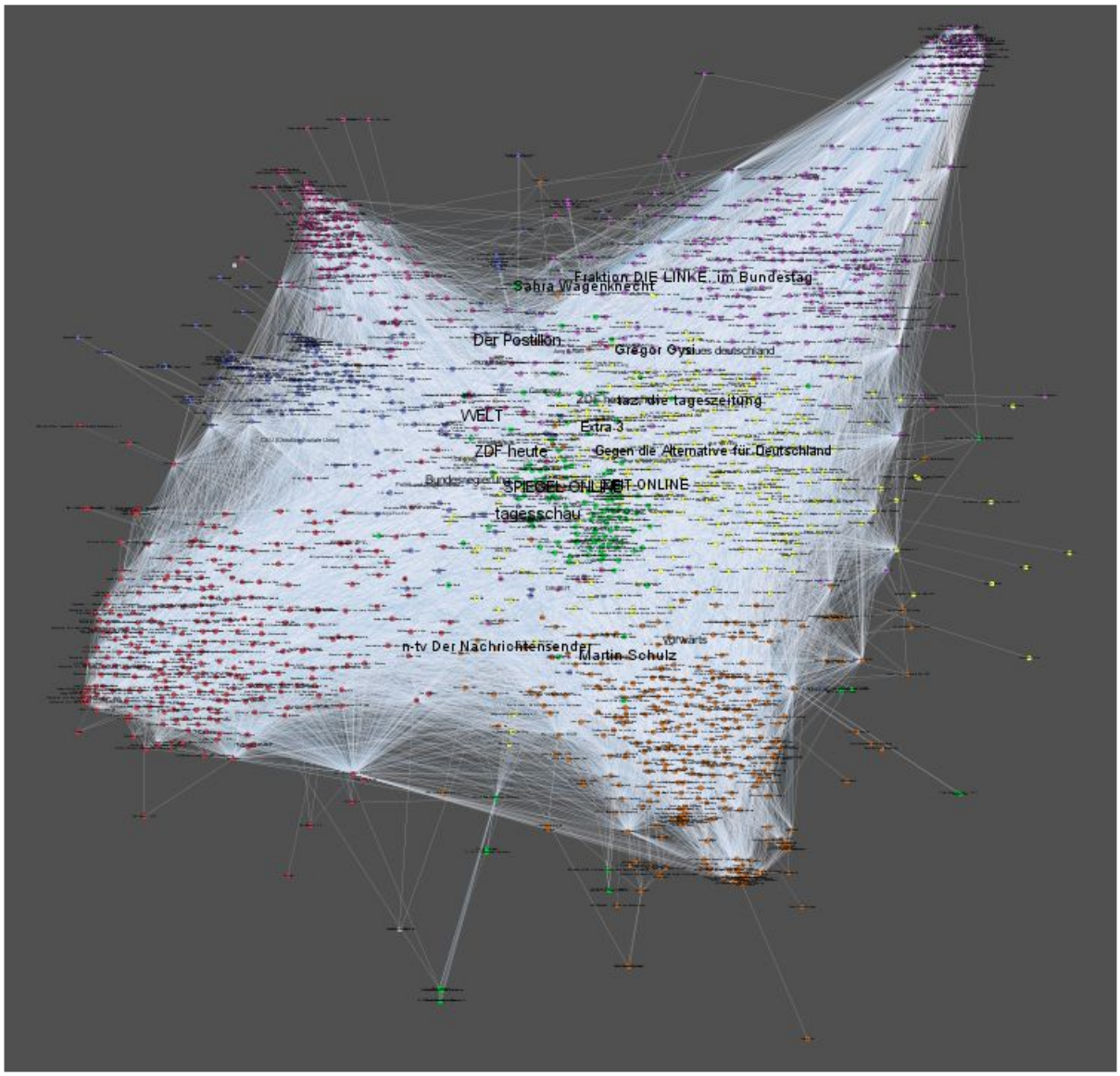

In the picture above, you can see the visualisation of communities that came out when a high resolution value (2) was applied. There are three communities each of which is highlighted in a different color: Pink highlights the AFD area, orange highlights the Die Linke area and green highlights everything else. We executed the community algorithm multiple times for each resolution value, since it is random based. However our results were stable. Sometimes the AFD group was taken into one community with the centered community, sometimes the Die Linke community was taken into one community with the centered community. But all in all, you can see that the more extreme parties separate themselves from the rest of the pages.

Normal Resolution Communities




In the picture above, you can see the visualisation appearing when a “normal” resolution value (1) was applied. There are five communities, each of which is highlighted in a different color: Blue highlights the AFD area, green highlights the CDU/CSU and SPD area, orange highlights the FDP area, pink highlights the Die Linke area and red highlights the news pages area. In multiple executions, the communities were stable with one exception: Sometimes the CDU/CSU area with the SPD area was considered a single community, but the CDU/CSU area with the FDP area was considered a single community just as often. This is interesting because these three parties are often called the parties in the middle.

## Low Resolution Communities




In the picture above, one can see the visualisation appearing when a low resolution value (0.8) was applied. There are six communities, each of which is highlighted in a different color: Red highlights nodes of area A, blue highlights nodes of area B, pink highlights nodes of area C, violett highlights nodes of area D, yellow highlights nodes of area E, orange highlights nodes of area F and green highlights nodes of area G. This visualisation is the least interesting because it reflects the areas that we were able to identify manually. However this outcome is important because it verifies our initial observation of the graph as well as the visualisation process in general.

# Conclusion

The page graph demonstrates in detail and according to one's expectation how pages can be divided into different groups or communities that reflect different political orientations. We were able to identify these communities by taking only user activity of a short period of time into account. There is a chance that the small sample we gathered from the huge data set being Facebook's social graph is biased. But it is not difficult to figure out that pages with similar political ideas attract the same users.

Similar results were shown in our user graph visualisations for the most active users. Its structure was very similar to the one of our page graph whereas the visualisation for the least active users foreshadowed the degree of the most important nodes.

It should however be noted, what our page graph actually represents: It only shows activity of users that were active on more than one page. About 83% of all users crawled were only active on one page. It is likely that this is influenced by our (relatively) small sample size, but it should also be considered when interpreting the page graph. It means that clusters might exist but only when looking at those parts of our data that allows to assume the existence of connections. The majority of users do not act on a broad set of pages (a part of which were visualised in the least active users graph), let alone those who do not act at all. In summary, that means: User activity only leads to very weak communities of pages.

The term "filter bubble" describes the horizon of events that a user is likely to see in his timeline. By observing clusters in the overall structure of the facebook graph, making an educated guess about the presence of filter bubbles would be unrealistic. However,it is a necessary condition for communities to exist in this overall structure, as long as we assume that there are different types of filter bubbles.

Answering the following two questions that our research approach was unable to explain could bear further possible contributions to the field:
Firstly, how does different content come to a user's timeline? This would take into account the silent majority we were unable to observe looking only at user activity; those who only watch, read and form their opinion without ever engaging in active participation on Facebook.
And secondly, how does information spread over the communities we were able to find? This would take qualitative aspects of news on Facebook into account. Having found communities does not necessarily mean that those tend to reflect a narrow set of opinions/ideologies - although assuming them would be somewhat based on reason .

However: We were able to show that the assumption of communities in the field of public opinions on Facebook is a plausible one.