ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

WHAT'S VIZ
PROCESS BOOK

COM-480 DATA VISUALIZATION

Tobias Oberdörfer, Jonas Blanc, Hugo Lanfranchi

4th June 2023

Chapter 1
What's Viz' story

1.1 Motivation

The ever-increasing amount of data generated by digital platforms presents both challenges and opportunities. As humans, we are always seeking innovative ways to extract meaningful insights from this vast sea of information. WhatsApp, with its massive user base and extensive message data, caught our attention as a valuable source of hidden knowledge waiting to be unlocked by every one of its users.

The popularity of WhatsApp as a communication tool, both in personal and professional spheres, highlights the importance of understanding and effectively utilizing the information embedded within its chats. By leveraging visualization techniques demonstrated in class, we set out to empower users in inspecting and managing their networks.

1.2 Goals

The primary goal of our project, What's Viz, is to harness the power of data visualization to transform each user's private WhatsApp data into actionable insights. Our aim was to develop a web application that seamlessly integrates with WhatsApp, providing users with dynamic visualizations and statistics that update in real-time.

We want to enable users to gain an understanding of their WhatsApp network, including their contacts and groups. By visualizing the connections between individuals and highlighting statistics about them, we aim to help users answer questions about their network structure and dynamics. Furthermore, we strive to provide insights into the languages used in conversations.

Through this project, we aim to contribute to the field of data visualization by visualizing WhatsApp data in near real-time. By addressing the challenges posed by the sheer volume of WhatsApp messages and the need for in-browser processing to keep privacy, we aim to provide a unique solution for users seeking to gain insights from their WhatsApp networks.

In the following sections of this process book, we will describe the path we took to achieve the final result. We will explain the challenges we encountered during development, the design decisions we made, and the evolution of our initial sketches and plans.

Now, let's dive into the details of our project's journey and explore the challenges we encountered, the design decisions we made, and the evolution of our initial plans.

1.3 Introductory video

In this video, we explain the aim of the project and go through the different visualizations our website has to offer. It's an ideal way for a user to get a complete overview of what's possible, although the website itself is designed to be self-explanatory.

Chapter 2
Implementation Challenges

2.1 Usability of data-handover

Given this whole project works on users' private data, a simple handover of this data to the browser had to be implemented. Users now simply need to scan the QR code displayed on our web application, enabling seamless access to their WhatsApp data. Once logged in, our system retrieves the user's WhatsApp data from the past three years locally in their browser. As long as the user remains logged in, new messages continuously flow in and are dynamically integrated into our visualizations. As soon as the user closes the tab or even just reloads it, all user data will be deleted. The biggest challenge for this usability was most certainly the receiving of user data in a privacy-preserving manner, which is described in more detail in the next section.

2.2 Privacy-preserving calculations on users' WhatsApp data

This subsection starts with how our system communicates with WhatsApp's multi-device API to access the user's messages in a privacy-preserving way, highlighting the use of the open-source project whatsmeow [1]. We then explain the challenges encountered when working with the whatsmeow package for WebAssembly, which is followed by a summary of our pre-processing steps and the challenges encountered during it.

To interact with WhatsApp's API and retrieve each user's private data, we utilized the open-source project called whatsmeow. This Go implementation of WhatsApp's multidevice API provided us with the necessary functionalities to establish communication with WhatsApp servers in an efficient manner. However, it is important to note that utilizing such tools is theoretically against WhatsApp's terms of service, as they prohibit the use of unofficial clients. Despite this restriction, various third-party tools exist that interact with WhatsApp's API, and WhatsApp itself cannot distinguish between a real web client and unofficial implementations, resulting in no adverse effects on our users.

Our self-imposed condition when starting this project was to preserve the privacy of our users' data by performing all data processing and visualization operations locally in the user's browser. To achieve this, all communication to WhatsApp's servers has to be done from within the web environment, resulting in the biggest encountered challenge: The original implementation of whatsmeow was not compatible with WebAssembly and hence not able to run in the browser.

Considerable effort was invested in adapting the code to run successfully in the browser. One part was the re-implementation of the database handling code of whatsmeow to use a custom SQL driver which can run in a web environment. The other part was getting a fully SQL-compatible database integrated into our front end. For the latter we ended up leveraging SQL.js [2], which is a port of SQLite to WebAssembly, providing us with SQL database functionality in the web environment. This eliminates the need for data transmission to any external servers and preserves the user's privacy. The former ended up being a modification of whatsmeow to interact with SQL.js through the browser's JavasSript state in an asynchronous manner.

After retrieving the users' WhatsApp data, we perform pre-processing steps to extract relevant information for later visualizations. The extracted data includes WhatsApp chats, consisting of messages and metadata, as well as details about contacts and groups. The main challenge of this pre-processing step was the correct identification of where to extract the relevant information from whatsmeow and how to hand this data over to the JavaScript layer.

Given that whatsmeow reimplements the multi-device API of WhatsApp, we first had to understand this API to some degree, applying filters and running queries to retrieve additional information (one example being the avatar URLs, which are not handed over by the WhatsApp API without explicitly requesting it for every single contact and group). The handover of the three types of messages, each waiting in their own go channel queue, is done by co-routines calling JS functions which were handed over at initialization.

To summarize, by utilizing whatsmeow, adapting it for WebAssembly, and reimplementing parts of it with a custom driver utilizing SQL.js, we achieved a robust and privacy-centric solution to get users' private WhatsApp data. Extracting only the relevant information from this data results in a dataset, which comprises tens of thousands of

messages, along with hundreds of contacts and dozens of groups, depending on the user's network and activity. This rich dataset then serves as the basis for our visualizations, enabling users to gain insights into their WhatsApp network.

2.3 MANAGING A DYNAMIC STATE

When receiving updates from the WebAssembly layer (new contacts / new groups / new batch of messages), instead of storing all the information, the JavaScript layer directly computes the statistics needed for visualization on the fly. Using React and its useState, we had to carefully design and implement reducers, calculating and aggregating the data asynchronously while avoiding concurrency issues.

This leads to a dynamic state that is continuously updated with new information retrieved from WhatsApp, either because new messages arrive while What's Viz is running, or older messages are synchronized (retrieving all available data may take a few minutes due to rate limiting). This dynamic state needs to be taken into account when building our visualizations, as we want them to update seamlessly and always be up-to-date with the latest retrieved data.

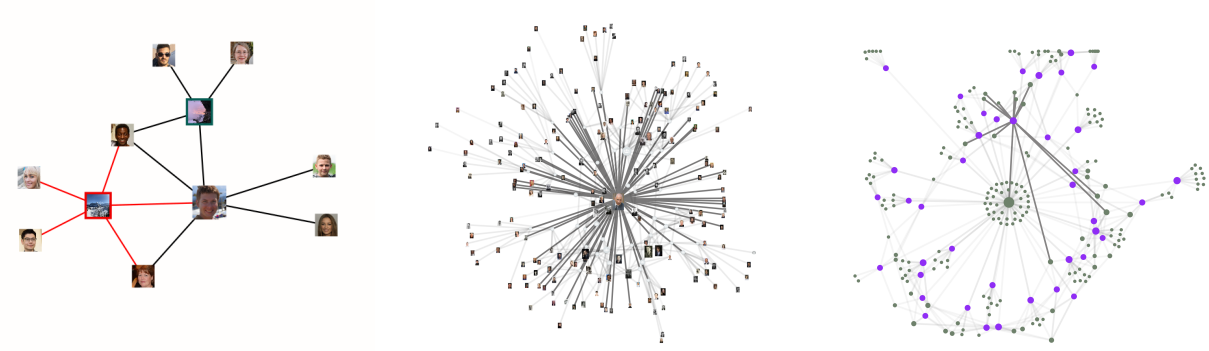
2.4 CREATING DUMMY DATA

To empower users to see a preview of what their data will be used for, we have generated fictitious data that mimics the data of an average user. What better than real people to simulate a user's contacts? We use The Movie Database (TMDB) to create a set of actor names and photos. The real challenge is to generate plausible random network landscapes with communities of contacts and groups. The first step is to create groups and assign participants to each group. We based the generation of dummy data on the observations and statistics of a multitude of cooperative beta testers. To reproduce the architecture of the groups, with a few large groups and many small ones, we chose the heavy-tailed Pareto distribution to dictate the number of members in each group. Between 75% and 95% of contacts are then used to fill the groups. We first divide these contacts into different groups, then use a random proportion of the participants in the groups already created to fill the remaining groups, thus creating inter-group communities. This leaves us with the final and most important stage, the generation of messages. Messages give an insight into both the strength of a community and the language or words used to communicate. For the sake of realism, we rely on the frequency of unigrams in a large corpus of online text to generate the message content. It should be noted that the messages generated do not actually make sense, but this generation of pseudo-random sentences is sufficient for the statistics calculated by the website to appear pleasing. We are just scratching the surface in terms of data analysis of what is feasible with the amount of rich data we have. Now that we have the content of the messages, when we need to select the sender and recipient, we set the number of contacts with whom the user has exchanged private messages at between 25% and 50% (this number is low because all the members of all the groups to which the user belongs are considered to be contacts). This proportion is high enough to take into account all contacts who are not members of groups and a few random group members. Finally, messages are assigned from a contact to a group or to another contact based on the groups previously generated and contact with private message split. We embedded in the website a random generation of 200 contacts, 40 groups and 4000 messages (a graph with such generation can be seen in Figure 2.1c).

2.5 THE NETWORK GRAPH

The quality of the network graph directly impacts the overall quality of the website, as it serves as the primary tool for user interaction. React was chosen as the framework for storing and updating the state, facilitating the seamless integration of new nodes and edges into the graph. The initial stage of the development process involved creating a working prototype. However, it was later decided to drop this prototype to provide greater flexibility to D3.js. The final version of the network graph utilized D3.js' force simulation.

One of the significant challenges encountered during development was ensuring the smooth performance of the graph, given its computational intensity due to the physics simulation. The graph relied on physics-based calculations to achieve a layout that accurately reflects the communities hidden in the data. To overcome this challenge, optimizations were implemented to handle the large number of contacts, groups, and messages effectively. A crucial aspect of the website was the ability to update the network graph seamlessly without restarting the simulation every time new data was received. To achieve this, an internal list of edges and nodes is maintained. Based on this list, new elements are selected and previously existing ones are updated if their statistics are affected



(A) Planned network graph visualization.

(B) Network graph with avatars instead of dots.

(C) Network graph for dummy data with selected node.

FIGURE 2.1
Iterations of the central network graph visualization.

by new data. The graph is then updated by running the simulation, allowing it to find a force equilibrium. The website incorporates a feature where clicking on a node allows users to delve into more detailed contact or group information. To ensure a smooth user experience, a mechanism was implemented to modify the layout of the graph without re-running the entire simulation. This mechanism is used to highlight the connections of the selected contact or group. Node size within the graph is scaled using a logarithmic scale based on the number of connections of the node. This approach effectively represents the importance of nodes within the network graph.

Image-based representations of nodes were explored, as imagined in planning (see Figure 2.1a). However, after receiving feedback in a small user study, it was decided to discard this design choice and instead rely on a carefully selected color scheme to facilitate the distinction of groups and contacts. The inclusion of images added complexity and reduced the readability of the network as seen in Figure 2.1b. As compensation, a tooltip feature was implemented. This tooltip displays the profile picture and contact name when hovering over a node (as seen in Figure 2.2), and the names of the two connected nodes when hovering over an edge. To allow users to get a quick understanding of which cluster corresponds to which of their contacts community we give the user an option to display the biggest contacts/groups names.

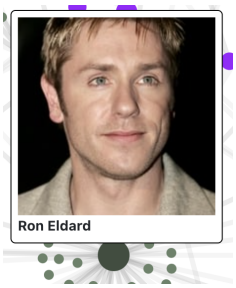


FIGURE 2.2
Example of on hover tool-tip for a contact.

Another significant challenge was setting forces based on the data to accurately reflect the hidden communities within the graph. The following forces were implemented: repulsion between nodes based on their number of connections (nodes with a higher number of connections experience stronger repulsion forces, promoting spacing and avoiding overcrowding), attraction along edges (edges exert attractive forces, with increased strength if one of the nodes has few connections, this facilitates the formation of community clusters), global layout force (a force is applied to center the nodes within the graph, creating a visually balanced layout). Designing the appropriate strength for these forces proved challenging, given the variations in data and screen sizes among users. Extensive experimentation and fine-tuning were performed to strike a balance between maintaining community clustering and accommodating diverse datasets and screen sizes.

2.6 THE CHAT SEARCH BAR

The decision to implement a search bar within our data visualization project stemmed from the recognition that finding specific friends or groups within the complex network graph was not a straightforward task. Navigating and identifying individual nodes or clusters in the graph can be challenging, especially when dealing with a large dataset comprising hundreds of contacts and groups. To address this usability concern and enhance the user experience, we incorporated the search bar feature seen in Figure 2.3.

The search bar serves as a convenient and intuitive tool for users to quickly select specific friends or groups of interest. By entering a partial name of the contact or group, users can initiate a search query to show all contacts or groups with a partial match. This significantly simplifies the process of locating specific contacts or groups, enabling users to efficiently navigate and explore their network connections. Finding all partial matches and displaying those

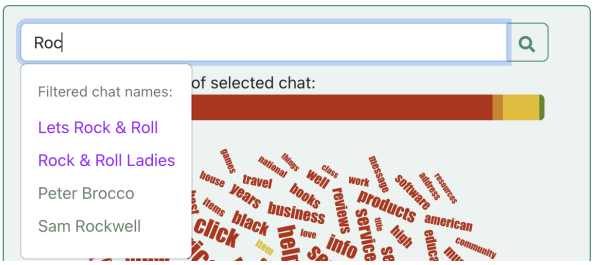


FIGURE 2.3
The search bar feature allows users to quickly locate and access specific friends (green) or groups (violet) within their WhatsApp network, simplifying the exploration of their connections.

correctly was the biggest challenge for the search bar because multiple different interactions by the user can change the selection. On one hand, clicking on a node in the graph changes the selection, which arrives at the search bar through an updated state, and on the other hand, it is the search button, an enter key press while the focus lies on the search bar input or lastly a click on a drop-down list item.

2.7 THE LANGUAGE STATISTICS

For the initial prototype, some simple statistics were implemented, which when extended to the full project posed their own set of challenges. We had to gather and analyze data from multiple sources, including messages, contacts, and groups, while all of those sources return the data dynamically and out of sync. Ensuring accurate and real-time updating of the statistics required careful handling of data synchronization and efficient processing techniques.

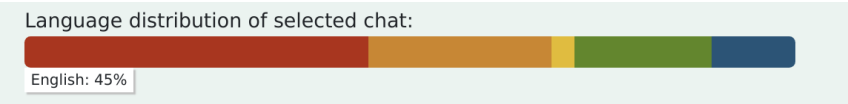


FIGURE 2.4
Visualization of language statistics for selected chat / contact.

While playing with our prototype we recognized the importance of incorporating language statistics and decided on the percentage bar visualization for it. This visualization, shown in Figure 2.4, allows users to identify the predominant languages utilized across their WhatsApp conversations or the currently selected chat. By aggregating message data and counting the occurrences of each language using natural language processing, we provide users with an overview of the languages they interact with most frequently on a per-chat basis. This information can be particularly valuable for individuals engaging in multilingual conversations which most of our peers in Switzerland are.

Implementing the language statistics visualization presented the challenge of extracting and processing the language information from the message data. Because machine learning is often not a precise technique in itself and our requirement of doing the extraction locally in real-time, the language extraction has a relatively low accuracy of only around 80%. Still, we utilize color scales to represent each language appropriately. Additionally, we ensured that the visualization adjusted dynamically based on the selected chat, enabling users to explore language patterns specific to their interactions.

2.8 THE WORD CLOUD

FIGURE 2.5
Word cloud of most frequently used words.

Another important visualization in our project is the word cloud, which was selected for its ability to assist in initial queries and facilitate the identification of prominent words within the WhatsApp user data. Despite the inherent drawbacks of word clouds, such as sub-optimal visual encoding due to word size variations, lack of structure, and unstable layout, the benefits of intuitive interpretation and familiarity outweigh these limitations in our specific context.

By incorporating language color coding into the word cloud, as

demonstrated in Figure 2.5, we further enhance the rapid identification of relevant information in a conversation, presenting a concise representation of the message data per selected chat.

Creating this word cloud did provide multiple challenges, the first of which was how to process the message data into words. We apply tokenization to break down the messages into individual words as the first step. We then initially experimented with showing the individual words stemmed, but found that the resulting word cloud was not easily readable by humans. Furthermore, no implementation of the inverse process of stemming for near-real-time updates existed. While lemmatization could potentially improve the word representation further, we did not find a suitable implementation that could handle the near-real-time updates of hundreds of words either. As such we decided to use the original words as they appear in messages, but with filtering applied. Our custom filtering mechanism effectively captures the relevant words and ensures a visually appealing representation in the word-cloud visualization. It does so by removing stop-words (using stop-word lists [3]), numbers, and symbols like hashtags or usernames.

A design decision we encountered was the inclusion of emojis in the word cloud, but because they are an integral part of modern communication and can convey important sentiment and context we decided to keep emojis in the word cloud.

CHAPTER 3
PEER ASSESSMENT

A breakdown of the parts of the project completed by each team member is presented below, highlighting the specific contributions of each member, and outlining the key responsibilities and tasks they undertook.

Tobias Oberdörfer:

- WhatsApp message data gathering: WhatsApp user data retrieval system from the past three years by porting an existing WhatsApp API client to WebAssembly to be used in the browser.
- WhatsApp data pre-processing: Design and implementation of the pre-processing pipeline for the received WhatsApp user data to bring it into the agreed-upon structure, ensuring consistency and compatibility with the visualization components.
- UI layout using React-Bootstrap [4]: Design and implementation of the user interface, including page layout, color scheme, login and disclaimer modals, and utilizing React-Bootstrap library to create a visually appealing UI.
- Network graph: Implementation of dynamic adding of nodes and edges.
- Chat search bar: Implementation of the search functionality within the web application, allowing users to search for specific chats or groups to filter other visualizations.
- Bag of words with NLP: Bag of words with the addition of Natural Language Processing [5] techniques to build a representation of the vocabulary and language used in the conversations.
- Language statistics bar: Developed and implemented the visualization of the language statistics bar to showcase the distribution of languages used in WhatsApp conversations.
- Word cloud: Leveraging D3cloud [6] and D3.js [7] to dynamically render the most frequent words contained in the bag of words per conversation.
- Screen-cast: Recording of audio and video according to decided-upon content with Jonas.

Jonas Blanc:

- Network graph: Design and implementation of the user's contacts landscape using D3.js force simulation based on [8]. This included data processing for nodes and edges, forces tuning for community layout, and increased interactivity via the addition of top naming buttons.
- Dummy data: Generation of plausible dummy data with actors' data, groups, and communities.
- Dynamic state: Implementation of React dynamic state by designing types and state reducers.
- Language statistics bar: Augmentation of the language bar with a more reactive legend tooltip.
- UI layout using React-Bootstrap: Support for the design and implementation of the user interface, utilizing React-Bootstrap library to create a visually appealing UI, and refactoring components to fit all screen sizes.
- Screen-cast: Creation of a dynamic presentation video based on Tobias' voice and screen recording.
- Pre-visualization: Creation of visual assets using InkScape for visualization planning.

Hugo Lanfranchi:

- Bart chart for most send messages per chat: Leveraging D3.js implemented a dynamic bar chart showing the most talkative contacts in chats.
- Pre-processing of WhatsApp messages to get time values: from WhatsApp messages get the associated timestamp
- Chart to display most active hours per chat: From whatapp messages and chat ids plot the share of the most active times for the users of a given chat
- Graphic improvements: made several graphics enhancements to the website for the user to understand more how the flow of information is fragmented
- Text for milestones.
- Tried on various devices/OS/versions to debug to make the website as easily reproducible as possible

By combining our individual expertise and working collaboratively, we were able to bring together the different components of the project, ensuring a cohesive and impactful outcome.

BIBLIOGRAPHY

[1] Whatsmeow contributers. *whatsmeow: Go library for the WhatsApp web multidevice API*. https://github.com/tulir/whatsmeow. 2023.

[2] SQLjs contributers. *SQLite compiled to JavaScript*. https://github.com/sql-js/sql.js. 2023.

[3] Damian Doyle. *Ranks NL*. https://web.archive.org/web/20230223121203/https://www.ranks.nl/stopwords. [Online; accessed 23-Feb-2023]. 2023.

[4] React-Bootstrap contributers. *React-Bootstrap*. https://github.com/react-bootstrap/react-bootstrap. 2023.

[5] natural contributers. *natural*. https://github.com/NaturalNode/natural. 2023.

[6] Jason Davies. *Word Cloud layout*. https://github.com/jasondavies/d3-cloud. 2023.

[7] D3 contributers. *D3: Data-Driven Documents*. https://github.com/d3/d3. 2023.

[8] Datartists. *Actors Galaxy*. https://github.com/com-480-data-visualization/datavis-project-2022-datartists. 2022.