Faculty of Informatics and Digital Technologies
University of Rijeka

Andrea Hrelja

Information Spread Analysis on Twitter

Master's Thesis

Mentor: prof. Ana Meštrović, PhD

Rijeka, Croatia
November 2022

FAKULTET INFORMATIKE I DIGITALNIH TEHNOLOGIJA
SVEUČILIŠTE U RIJECI

uniri

UNIVERSITAS STUDIORUM FLUMINENSIS
SVEUČILIŠTE U RIJECI

Rijeka, 2.6.2022.

Zadatak za diplomski rad

Pristupnik: Andrea Hrelja

Naziv diplomskog rada: Analiza širenja informacija na Twitteru

Naziv diplomskog rada na eng. jeziku: Information Spread Analysis on Twitter

Sadržaj zadatka:

Zadatak diplomskog rada je prikupiti i analizirati podatke s Twittera s ciljem istraživanja obrazaca širenja poruka u mreži. Potrebno je implementirati i opisati sustav za automatizirano i kontinuirano prikupljanje tweetova. Potom treba provesti analizu podataka i različitih korisničkih aktivnosti na Twitteru, kao što su: *retweet*, *reply*, *quote*, *mention*. Nadalje, zadatak je analizirati različite načine širenja informacija u mreži.

Mentor: Prof. dr. sc. Ana Meštrović

Voditeljica za diplomske radove:
Prof. dr. sc. Ana Meštrović

Zadatak preuzet: 2.6.2022.

(potpis pristupnika)

Abstract

This thesis seeks answers to what happens with an information when someone contributes to sharing it on a Twitter network composed of Croatian users. It describes methods involved in planning and developing a Twitter *data platform* and introduces a data analysis with the goal of quantifying the way information is spread on the network. The network is analysed by applying different visualisation techniques and data structures to gain insights about the users in this network, the relationships between them, the information they share and the relationships between the information shared.

Keywords - SDLC, data platform, data pipeline, data ingestion, data transform, data analysis, information spread, Twitter, Python, pandas, matplotlib

Contents

1 Introduction

Social networks connect people in ways not even imagined when the Internet had just begun to take over the world's population, let alone in the years before. With a single tap of a touchscreen, we are immersed in a world of endless information disseminated by an ever-growing number of blogs, websites, media platforms and social networks. In this sea of information, it is becoming increasingly difficult to find a clear cove.

Before the Internet took hold around the world, the main sources of information were the mainstream media such as the press, TV channels and radio stations, all kinds of banners and leaflets passed from hand to hand, and finally - word of mouth. This type of interaction between the source of information and the target of information (people) is often direct, but can sometimes be through intermediaries. We can imagine a scenario where a leaflet distributor asks other people to help him share the leaflets - not many people will contribute because they may be too busy or simply not interested. Many people may have already seen the leaflet and simply ignore the distributor. Some would certainly appreciate the cause the fliers are promoting and agree to pass it along

This thesis seeks answers to what happens when someone contributes to share the flyer (tweet) on a Twitter network composed of Croatian users. It collects the available data on Twitter, prepares it for data analysis and aims to quantify the way information is spread on the network in order to identify, among others, the main media and leaflet distributors. The network is analysed by applying different visualisation techniques and data structures to gain insights about the users in this network, the relationships between them, the information they share and the relationships between the information shared.

2 Methods

2.1 Overview

The goal of this thesis is to analyse social trends on Twitter and determine how trends spread among Croatian Twitter users over a period of time. The main challenge in conducting a data analysis is obtaining a credible dataset that allows for insightful analysis. To overcome this challenge, a *data platform* [1] was created to collect, clean, transform, and apply data using a data pipeline developed in Python [1].

The following subsections describe the implementation process of the *data platform*. Work examples for the phases of the System Development Life Cycle (SDLC) are defined and described. The advantages and disadvantages of the software process models and how these methods are used to identify the created platform are shown. In addition, the ingestion, transformation and analysis processes are described in detail. This section concludes with a description of the CI/CD infrastructure.

2.2 System Development Life Cycle

Every software development process has a starting point, but saying that it has an end point can be ambiguous. Once the software is deployed in a production environment and all features have been developed, it can be assumed that the development phase is complete and the end point has been reached. At this point, software maintenance, often referred to as software support, can begin and additional system enhancements can be made. Maintenance usually focuses on bug fixes or new enhancements, but sometimes end users may request features that require additional development, which restarts the development process

Given the cyclical nature of software processes, numerous software process models [2] have been developed by various authors throughout history. The most famous model, the waterfall model, was introduced in the 1970s, but did not address the cyclic nature of software processes and was therefore an inefficient mechanism for software development. Over time, other models were developed to improve the rigidity of the waterfall model, leaving a variety of options to choose from to select the most appropriate software process model and create a flexible System Development Life Cycle.

A software process model is used to identify the system to be built.

[1] Thesis code repository: https://github.com/andhrelja/twitter_scraper

It defines the activities for designing, implementing, testing, and monitoring software systems. Some software process models include the classic Waterfall Model, Incremental and Iterative Models and their combination, the Rapid Application Development (RAD) Model, various Prototyping Models, the Spiral model, the Rational Unified Process (RUP) Model, the V-model (verification and validation model) as an extended implementation of the Waterfall Model, and others [3]. The waterfall model is a precursor to other software process models, but it poses risks to the project outcome due to its linear (sequential) life cycle model.

All the models listed have similarities between the phases they define (analysis, design, development, test, and maintenance). Most models are based on an iterative life cycle model that allows for flexible requirements updates, and they are often user-driven in an Agile manner. The *data platform* created as part of this thesis is a small project guided by an implementation of iterative rapid prototyping through Analysis, Design, and Development & Testing cycles.

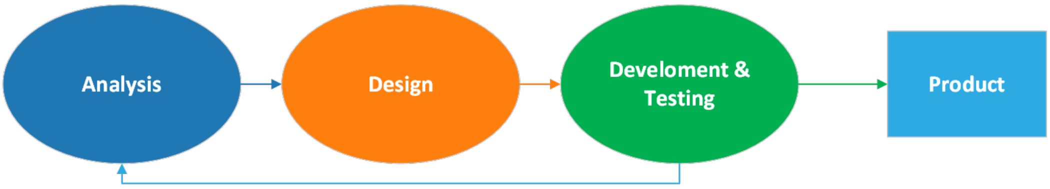

Figure 1: Twitter *Data platform* SDLC

During Development & Prototype, there was frequent switching between the analysis phase and the development and testing phase. This model has proven to be a fast development and release process, especially for features that required additional testing

After the Testing phase demonstrates that the developed system meets the requirements of the Development & Prototype phase, the Maintenance phase is initiated. This transition requires that data output not be compromised by developers, analysts, or others interacting with the *data platform*, so a production environment is introduced. This means that development and testing must occur as infrequently as possible and with as few changes as possible to avoid data integrity issues. To improve the security of the system, native GitHub functions such as branch protection rules can be used.

The following subsections describe the rapid prototyping phases in the platform life cycle.

2.2.1 Analysis

The **goal** of this thesis is to create a dataset that provides valuable knowledge about the information being shared on Twitter and to enable analyzing social trends on Twitter through time.

Analysis phase identifies the **data service provider**, **data source** endpoints, their limitations and restrictions, and finally - **data requirements**.

Data service provider Twitter[2] is the data service provider for this *data platform*. The provided REST service imposes various limitations and restrictions, some of them being Tweet limits - *up to 500k Tweets* are served per month and User limits - *inability to lookup Users* in a given range (location, age...). The identified limitations and restrictions need to be accounted for at the earliest stages of the Analysis phase. Failing to do so may result with disastrous effects on the product at a stage when it is too late to iterate over Analysis again. This *data platform* accounts for Tweet limits with an option of creating multiple accounts if the *500k Tweets per month* limit is surpassed and re-running the full load process (2.3). User limits are accounted for by creating a baseline list of User IDs and expanding it with each data pipeline run using the user's followers and friends IDs [4].

Data source Data source endpoints are used to collect information about Twitter defined User objects and Tweet objects. Endpoints impose technical limitations, with most common limitations including a limited number of API requests that an external system can make to the data source's REST server. These limitations are documented in table 2.2.1.

[2] https://twitter.com

[3] https://developer.twitter.com/en/docs/twitter-api/v1/accounts-and-users/follow-search-get-users/api-reference/get-users-lookup

[4] https://developer.twitter.com/en/docs/twitter-api/v1/tweets/timelines/api-reference/get-statuses-user_timeline

[5] https://developer.twitter.com/en/docs/twitter-api/v1/accounts-and-users/follow-search-get-users/api-reference/get-followers-ids

[6] https://developer.twitter.com/en/docs/twitter-api/v1/accounts-and-users/follow-search-get-users/api-reference/get-friends-ids

Name	Description	Limitations
users/lookup[3]	serves User objects	900 requests / 15 minutes
statuses/user_timeline[4]	serves Tweet objects associated with a user	900 requests / 15 minutes
followers/ids[5]	serves user's follower IDs	15 requests / 15 minutes
friends/ids[6]	serves user's friends IDs	15 requests / 15 minutes

Table 1: Source data Endpoints Descriptions & Limitations

Data requirements Data requirements identify the information that needs to be obtained from the data source. The data service provider must be capable of satisfying the given data requirements by providing curated sets of information. The resulting *data platform* must be designed to support the given data requirements. The following bullet list provides the requirements summary:

- **Users quantity**: Croatian users only
- **Tweets quantity**: from `2022-11-01` onward
- **Data re-usability**: never delete ingested data
- **Object details**: identify attributes that can be used to analyse social trends

Collected (*ingested* afterwards) data needs to provide as much information about *data source* objects as possible. This is accomplished by storing all available source object attributes and only eliminating irrelevant attributes in the Data Transformation process. The ingested data is never deleted, moved or modified (in place).

The following paragraphs describes some valuable *data source* object attributes and other inputs used to apply User and Tweets quantitative restrictions.

Location Inputs A User object is identified as a Croatian user if their `location` attribute **contains** at least one Croatian location from 1. As an example, if a User object's `location` attribute is set to `"Zaprešić"`, he will be identified as a Croatian user because (`"Zaprešić"` is a subset of

`"Zaprešić,Croatia"` **or** `"Zaprešić,Croatia"` is a subset of `"Zaprešić"`).

Twitter objects 1: Croatian Locations Input

```
[
    "Hrvatska",
    "Croatia",
    "Žminj",
    "Zaprešić,Croatia",
    "Virovitica",
    "republic of dalmatia baby!",
    "velika gorica, croatia",
    "RIJEKA"
]
```

The Croatian Locations Input was derived manually by inspecting the Ingest User's `location` attribute values and capturing them in a separate JSON file. The values were only inspected once and the input locations were not revised since.

Additional transformations were performed on Croatian Locations Input to ensure Users are correctly identified as Croatian users. Some transformations include converting the `location` strings to lowercase, punctuation and diacritics removal and white-space and Unicode character removal.

These operations were simple to execute on this input file, but they were time-expensive when applied to the User models and they did not yield results that would justify their use. After performing this due diligence, the additional transformation functionality was discarded and it is no longer in use.

Ingested Users The User object contains a large number of attributes. Only a subset of attributes are presented and described, but all of them are ingested. Details about all attributes can be found at the Twitter's Tweet object[7] page.

[7] https://developer.twitter.com/en/docs/twitter-api/v1/data-dictionary/object-model/user

Twitter objects 2: Ingest User

```
{
    "name": "HNS",
    "screen_name": "HNS_CFF",
    "location": "Hrvatska | Croatia",
    "description": "Službeni Twitter profil Hrvatskog"
                   "nogometnog saveza Croatian Football"
                   "Federation official Twitter feed."
                   "#HNS #Obitelj #Family",
    "protected": false,
    "followers_count": 262456,
    "friends_count": 116,
    "statuses_count": 19515
}
```

- `name`: The name of the user, as they have defined it. Not necessarily a person's name
- `screen_name`: The screen name, handle, or alias that this user identifies themselves with. `screen_names` are unique but subject to change
- `location`: *Nullable.* The user-defined location for this account's profile
- `description`: *Nullable.* The user-defined UTF-8 string describing their account
- `protected`: When true, indicates that this user has chosen to protect their Tweets (Verified Accounts[8])
- `followers_count`: The number of followers this account currently has
- `friends_count`: Number of accounts that the user follows
- `statuses_count`: The number of users this account is following (also known as their `"followings"`)

[8] https://help.twitter.com/en/managing-your-account/about-twitter-verified-accounts

Ingested Tweets Twitter uses the term *status* when referring to a Tweet. The Tweet object contains a large number of attributes. Only a subset of attributes are presented and described, but the entire set is ingested. Details about all attributes can be found at the Twitter's Tweet object[9] page.

Twitter objects 3: Ingest Tweet

```
{
    "created_at": "Tue Nov 22 10:00:18 +0000 2022",
    "full_text": "VATRENI IS LISTED"
                 "\n\nThis is truly a historic moment"
                 "because #VATRENI is so much more "
                 "than just a token. Become a part"
                 "of the greatest fan story and enjoy"
                 "all kinds of benefits.\n\n#VATRENI"
                 "token is now live at @gate_io \n\n",
    "entities": {
        "hashtags": [
            {"text": "VATRENI"},
            {"text": "VATRENI"}
        ],
        "user_mentions": [
            {"id": 912539722, "screen_name": "gate_io"}
        ]
    },
    "user": { ingested_user },
    "retweet_status": { ingested_tweet },
    "in_reply_to_status_id": null,
    "in_reply_to_user_id": null,
    "quoted_status": { ingested_tweet },
    "favorite_count": 41,
    "possibly_sensitive": false,
    "lang": "en"
}
```

- `created_at`: UTC time when this Tweet was created
- `full_text`: The actual UTF-8 text of the status update

[9] https://developer.twitter.com/en/docs/twitter-api/v1/data-dictionary/object-model/tweet

- `entities`: Entities which have been parsed out of the text of the Tweet. Additionally see Entities in Twitter objects[10]
  - `hashtags`: Names of the hashtags used in this Tweet, minus the leading `"#"` character
  - `user_mentions`: IDs of the mentioned users, as an integer
- `user`: Ingested User (Ingest User)
- `retweet_status`: Retweets can be distinguished from typical Tweets by the existence of a retweeted_status attribute. This attribute contains a representation of the original Tweet that was retweeted (Ingest Tweet)
- `in_reply_to_status_id`: *Nullable.* If the represented Tweet is a reply, this field will contain the integer representation of the original Tweet's ID
- `in_reply_to_user_id`: *Nullable.* If the represented Tweet is a reply, this field will contain the integer representation of the original Tweet's author ID
- `is_quote_status`: Indicates whether this is a Quoted Tweet
- `quoted_status`: This attribute contains a representation of the original Tweet that was quoted (Ingest Tweet). Quote tweets are Retweets that contain some original content (`full_text`, `hashtagas`, `user_mentions`)
- `favorite_count`: *Nullable.* Indicates approximately how many times this Tweet has been liked by Twitter users
- `possibly_sensitive`: *Nullable.* This field indicates content may be recognized as sensitive. This may also be judged and labeled by an internal Twitter support agent
- `lang`: *Nullable.* When present, indicates a BCP 47[11] language identifier corresponding to the machine-detected language of the Tweet text, or und if no language could be detected

[10] https://developer.twitter.com/overview/api/entities-in-twitter-objects

[11] http://tools.ietf.org/html/bcp47

2.2.2 Design

Design phase focuses on shaping the mechanisms by which data is ingested and transformed to create a dataset that can be easily used for data analysis

The resulting *data platform* must support storage of the full raw data as-is, without any changes being made in the ingestion process (2.3). The transformation process uses the full data set to apply filters and changes to the raw data. This process performs read operations on the ingested data, transforms in-memory data, and stores the resulting dataset in a new location that is independent of the ingestion location. Before the resulting dataset is stored, it is filtered to contain only Croatian users, and the baseline list of User IDs is overwritten with the available Croatian user IDs. Figure 2.2.2 shows an overview of the implemented *data platform* architecture.

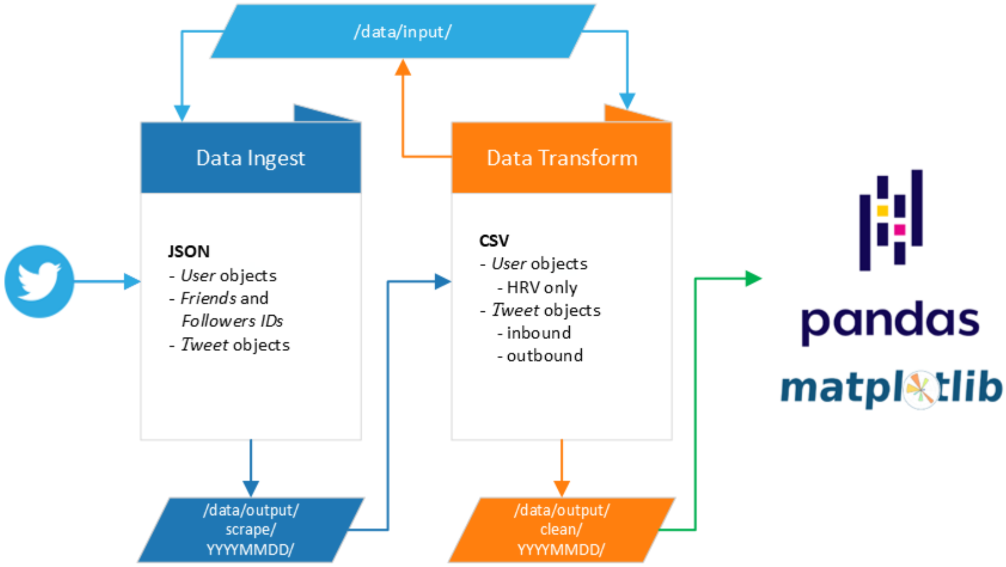

Figure 2: Twitter *Data platform* Architecture

Both ingestion and transformation output locations (in particular, the directories reflecting the ingestion date) are created at runtime. This mechanism allows the ingested and transformed data to be examined per ingestion date, and ensures that only the most recently ingested data is transformed, rather than the entire data for each data pipeline run. The resulting dataset is a union of all files in the date directories in the output file system of the data conversion and is analysed (2.5) using the pandas and matplotlib tools.

Data target filesystem A filesystem is used to store data throughout the data pipeline. The data is stored on an on-premise Linux server provided by the University of Rijeka. The filesystem includes two main directories: `input` and `output`. The `output` directory is then divided into the `scrape` (ingested data) and `clean` (transformed data) directories, partitioned by date.

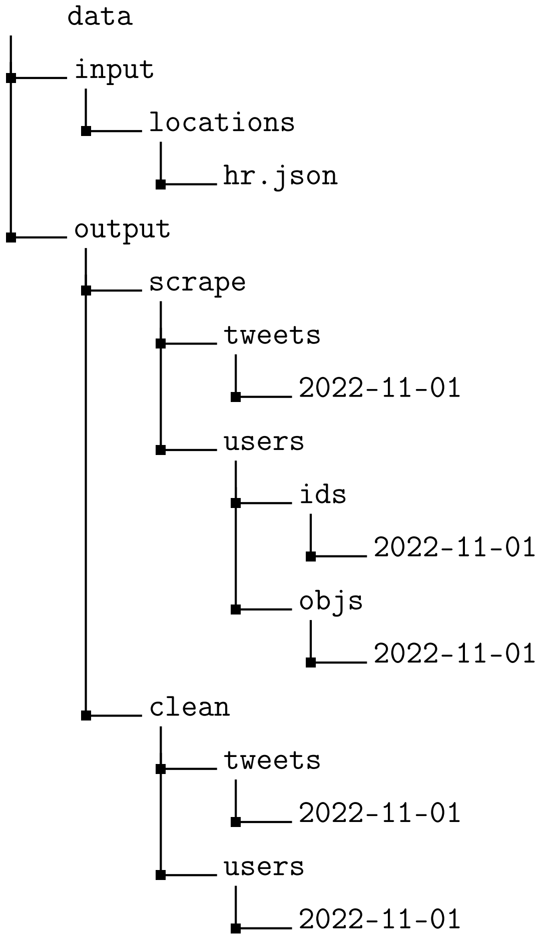

Figure 3: Data target file system structure

2.2.3 Development & Testing

Development (Prototyping) & Testing phase is focused on developing the mechanisms used to ingest, and transform data defined in the Design phase.

This phase ensures that ingestion and transformation are independent (loosely coupled) processes, that enable historical and incremental data ingestion, notification mechanism and CI/CD mechanisms (2.6).

2.2.4 Maintenance

Maintenance phase focuses on supporting the end-users in their *data platform* usage. Once Development & Testing is completed and the system is deployed to *production*, it is maintained to ensure that any unexpected bugs are promptly fixed and supported to accommodate new enhancements. This phase may also support feature development requests, but if a development effort exceeds the scope of the defined Design, a new iteration is initiated across SDLC phases.

2.3 Data Ingestion

Data ingestion is the process of obtaining data from a *data source* to its home system as efficiently and correctly as possible [5]. Home system in the context of this thesis is a file system used as a part of the developed *data platform*. A *data source* is a place where information is obtained - the source can be a database, a flat file, an XML file, or any other format an operating system can read.

The *data source* used by this thesis is Twitter, offering a REST API service to serve their data. High volume of data generated on Twitter makes it difficult for the REST service to enable high data availability, so Twitter limits the amount of data that can be collected per user (e.g. only the **latest 3,200** user's tweets can be obtained). To support historic data storage - the collected data is continuously collected and never deleted.

Data ingestion architecture depends on the *data source* system type and the Data Analysis requirements. Four common architecture patterns are described in the following paragraphs.

Real-Time Ingestion Real-Time Ingestion is usually used for real time data. It applies simultaneous intermittent processing to data in small sizes (order of Kilobytes). This is an event-based ingestion system.

Streaming Ingestion Streaming Ingestion is usually used for streaming data. It applies simultaneous continuous processing of data in small sizes (order of Kilobytes). This is an event-based ingestion system.

Batch Ingestion Batch Ingestion is usually used for big data. It applies non-simultaneous processing of data in large sizes (batches, order of Megabytes). This is a schedule-based ingestion system.

Lambda Architecture Lambda Architecture is usually used for a combination of streaming data and big data. It applies simultaneous and non-simultaneous processing of data in all sizes. This can be event-based and schedule-based ingestion system.

Twitter *data source* provides both real time data and big data. Real-time ingestion is not required by the Data Analysis for this *data platform* because Data Analysts conduct their analysis on weekly or monthly bases. Given the requirements, the **batch ingestion** architecture pattern is selected to guide the *design* for this *data platform*, with the ingestion schedule set for every **Monday at 12AM UTC**.

After the architecture pattern is curated, the methods for moving the data need to be defined. It is important to note that all architecture patterns require data movement methods, but each pattern only supports a subset of data movement methods. Two of the most commonly used data movement methods for batch ingestion are described in the following paragraphs.

Full Load Full loads are also known as *Historic* loads. They takes place the first time a data source is loaded into the home system.

Incremental Load Incremental loads are also known as *Delta* loads. They take place on each subsequent time a data source is loaded into the home system. Delta loads are usually tracked by a date and time value (last received record date and time, last historic or incremental load date and time and similar) for the next incremental load to correctly identify the starting point of the data being collected.

This *data platform* supports full and incremental data movement methods. Because of the Twitter REST limitation where only the latest 3,200 user's tweets can be obtained, it is essential that the collected data is stored and never deleted. If the collected data gets deleted, it is unrecoverable because Twitter will never serve the user's 3,201st tweet using the current version of their REST service again.

By creating software support for incremental loads, support for full loads is implied. The data pipeline reads the following JSON inputs to determine what is the starting point of the data being moved:

- `baseline-user-ids`: array of user IDs, manually created to collect tweets from
- `processed-user-objs`: array of user IDs, processed in a previous ingestion
- `missing-user-objs`: array of user IDs, non-existing profiles for a given user ID
- `max-tweet-ids`: object, last received tweet record for a user represented by a key-value pair (`{user ID: last tweet ID}`)

First time the data is loaded into the home system, the full load takes place collecting the manually created `baseline-user-ids`. At this point, the remaining inputs still do not exist. After the first user ID is processed, the respective user object is created, adding the processed user ID to `processed-user-objs`. Once all the users are processed, they are filtered to Croatian users only (using the Croatian Locations Input JSON input) within the Data Transformation process, the `baseline-user-ids` gets updated with the Croatian user IDs and the tweet ingestion starts. Since there isn't a `max-tweet-ids` to determine the last received user's tweet, all the latest $3,200$ user's tweets (from today) until `2022-11-01 00:00 UTC` are obtained and the `max-tweet-ids` file is created.

All the next loads are incremental ones, reading inputs created by the full load. Incremental loads only collect the `baseline-user-ids` that do not exist in `processed-user-objs` and `missing-user-objs`.

2.4 Data Transformation

Data transformation is the process of converting data from one format or structure into another format or structure. This is often done to make the data more useful or easier to work with for specific purposes, such as analysis or machine learning. Data transformation can involve a wide range of techniques, such as cleaning and preprocessing, normalization, aggregation, and feature extraction. The specific steps involved in a data transformation process will depend on the specific data and the desired end result.

The data transformation process applied to the collected data is designed to be re-runnable in a way that does not affect the stored raw data. This process usually consumes a large amount of time, so it only runs once. The transformed data is then reused throughout the 2.5 process. To support the given data requirements, the transformation process filters all collected Users to Croatian users only (Croatian Locations Input) and applies other filters (`statuses_count > 10`, `followers_count > 10`, `friends_count > 10` and similar), to ensure the collected Users represent a legitimate sample.

The resulting transformed data is used to create data views to be used by the Data Analysts.

Twitter objects 4: Transform User

```
{
    "name": "HNS",
    "screen_name": "HNS_CFF",
    "location": "Croatia",
    "is_croatian": true,
    "description": "Službeni Twitter profil Hrvatskog"
                   "nogometnog saveza Croatian Football"
                   "Federation official Twitter feed."
                   "#HNS #Obitelj #Family",
    "followers_count": 253625,
    "friends_count": 117,
    "statuses_count": 19209,
    "created_at": "2022-08-30T05:56:36+00:00"
}
```

The User object contains a small number of transformations compared to the Tweet object. The following attributes and transformations are applied to the User object:

- `location`: extract city name text value if it is included in the original value, otherwise set to `Hrvatska`

- `is_croatian`: create boolean value to indicate whether or not the user is from Croatia

- `description`: User provided profile description

- `followers_count`: number of users following this User

- `friends_count`: number of users this User follows

- `statuses_count`: total number of published Tweets since `created_at`

- `created_at`: evaluate string to date-time object; represents profile creation date and time

Twitter objects 5: Transform Tweet

```
{
    "created_at": "2022-11-22 10:00:18+00:00",
    "created_at_year": 2022,
    "created_at_month": 11,
    "created_at_week": 47,
    "created_at_day": 22,
    "full_text": "VATRENI IS LISTED"
                 "\n\nThis is truly a historic moment"
                 "because #VATRENI is so much more "
                 "than just a token. Become a part"
                 "of the greatest fan story and enjoy"
                 "all kinds of benefits.\n\n#VATRENI"
                 "token is now live at @gate_io \n\n"
                 "Get it here: "
                 "https://t.co/sN8mWtUac6"
                 "https://t.co/XfqNfwUeCS",
    "hashtags": ["VATRENI", "VATRENI"],
    "user_mentions": ["gate_io"],
    "is_retweet": true,
    "retweet_count": 15,
    "retweet_created_at": "2022-11-22 10:29:47+00:00",
    "retweet_from_tweet_id": 1594994170858463232,
    "retweet_from_user_name": "vatreni_token",
    "retweet_timedelta_sec": 960,
    "is_reply": false,
    "reply_to_tweet_id": null,
    "reply_to_user_name": null,
    "is_quote": false,
    "favorite_count": 41,
    "possibly_sensitive": false,
    "lang": "en",
    "transform_date": "2022-11-22"
}
```

The Tweet object contains a large number of transformations. The following attributes and transformations are applied to the Tweet object:

- `created_at_year`: extract year number from `created_at`
- `created_at_month`: extract month number from `created_at`
- `created_at_week`: extract week number from `created_at`
- `created_at_day`: extract day number from `created_at`
- `hashtags`: extract hashtag text from `entities.hashtags`
- `user_mentions`: extract mentioned user's screen_name from `entities.user_mentions`
- `is_retweet`: create boolean value based on existence of `retweeted_status`
- `retweet_count`: create numeric value based on all other Tweet objects where their `retweeted_status.id` equals this Tweet object's `id`
- `retweet_created_at`: extract date-time object from `retweeted_status.created_at`
- `retweet_from_tweet_id`: extract numeric Tweet identifier from `retweeted_status.id`
- `retweet_from_user_name`: extract text User identifier from `retweeted_status.user.user_id`
- `retweet_timedelta_sec`: create timedelta[12] value based on the difference between `retweet_created_at` and `created_at`
- `is_reply`: create boolean value based on existence of `in_reply_to_status_id`
- `reply_to_tweet_id`: rename `in_reply_to_status_id`
- `reply_to_user_name`: rename `in_reply_to_screen_name`
- `is_quote`: rename `is_quote_status`
- `lang`: apply a language detection function using langid[13] if the original value was undefined
- `transform_date`: create text value based on the transformation date

[12] https://pandas.pydata.org/docs/reference/api/pandas.Timedelta.html

[13] https://github.com/saffsd/langid.py

2.5 Data Analysis

Data analysis is a process of inspecting, cleansing, transforming, and modeling data with the goal of discovering useful information, informing conclusions, and supporting decision-making [6]. It is usually conducted by a Data Analyst. Depending on the context to which the term is applied, data analysis can imply additional methods such as ingesting, transforming, modeling or other data processing methods. Within the context of this thesis, data analysis includes preprocessing transformed data, analysing it and interpreting the analysis to draw conclusions about the ways users share information and what information they share on this social network.

In statistical applications, data analysis can be divided into descriptive statistics, Exploratory Data Analysis (EDA), and Confirmatory Data Analysis (CDA) [7]. Descriptive statistics is the process of using and analysing quantitative descriptions or feature[14] summaries from a collection of information [8]. EDA focuses on discovering new features in the data while CDA focuses on confirming or rejecting existing hypotheses. Additionally, in social networks and similar applications, graph analytics is frequently used to observe the relationships between objects which are being analysed.

The analysis conducted as a part of this thesis combines descriptive statistics, EDA and graph analysis approaches to describe the available dataset, summarize its main characteristics and derive new data views that allow for tracking social trends on Twitter through time. In the scope of this thesis, a trend is defined as a set of topics consisting of hashtags (often referred to as Tweet content).

2.5.1 Descriptive Statistics

Descriptive statistics applications include data statistics calculation techniques to provide a quantitative summary of the analysed data. Additional data metrics include the earliest and latest tweet date and time (`2022-11-01 00:00:12+00:00 - 2022-11-30 23:59:52+00:00`), total number of tweets ($386,168$), number of all Croatian Users ($48,954$) and the number of Users who are active within the described time range ($6,887$). Additional information available in AppendixA. Table 2. provides a summary about the measures of the data. Some information that can be interpreted is the average number of followers a user has (`mean` of `followers_count`: $1,408.73$,

[14] The terms *attribute*, *field*, and *column* from previous sections are synonyms to the term *feature*. The main difference between the terms is that Data Analysts tend to use the term *feature* to empathize the importance of an *attribute*

however this average is unreliable because of a high standard deviation - the `90%` column shows that the least amount of users has the most followers) and the average number of original tweets a user posted (*mean* of `original_tweets_cnt`: 95.34). It is surprising to see that only a small group of users' tweets get retweeted or quoted (*mean* of `in_retweet_cnt`: 2.67, `in_quote_cnt`: 1.0). Related works on the topic of Twitter analysis have shown that retweets are the most common way of spreading information on Twitter [9], so it can be expected that the information spread among Croatian Twitter Users is seeded from a very small number of users - those with the highest retweets count.

Table 2: Statistical measures describing numeric data features in Users, post preprocessing

	count	mean	std	min	30%	60%	90%	max
followers_count	6,887	1,408.73	21,214.16	11.0	68.0	262.0	1,662.8	1,541,746.0
friends_count	6,887	568.93	751.9	11.0	168.0	410.0	1,334.6	5,002.0
total_out_tweets_cnt	6,887	138.76	396.97	1.0	7.0	34.0	320.0	5,342.0
original_tweets_cnt	6,887	95.34	298.68	0.0	3.0	19.0	211.0	5,337.0
out_retweet_cnt	6,887	43.41	216.0	0.0	0.0	5.0	65.0	5,093.0
out_reply_cnt	6,887	59.38	221.26	0.0	1.0	7.0	127.0	5,180.0
out_quote_cnt	6,887	8.89	48.69	0.0	0.0	1.0	15.0	1,479.0
out_retweet_timedelta_sec	4,598	809,044.12	6,993,425.17	8.5	26,884.7	62,736.0	948,475.5	315,960,857.0
out_quote_timedelta_sec	2,344	1,717,686.97	10,939,711.03	20.33	20,514.07	66,729.8	1,786,563.6	281,322,829.8
total_in_tweets_cnt	6,887	16.18	75.21	0.0	0.0	1.0	28.0	1,844.0
in_retweet_cnt	6,887	2.67	22.95	0.0	0.0	0.0	3.0	1,266.0
in_reply_cnt	6,887	12.69	62.69	0.0	0.0	0.0	19.0	1,499.0
in_quote_cnt	6,887	0.82	6.13	0.0	0.0	0.0	1.0	374.0
in_original_favorite_cnt	6,887	581.45	6,796.17	0.0	1.0	22.0	661.0	458,954.0
in_retweet_favorite_cnt	6,887	910,049.29	5,439,527.81	0.0	0.0	12,987.0	1,342,942.2	198,673,449.0
in_quote_favorite_cnt	6,887	128,179.92	711,156.8	0.0	0.0	0.0	139,581.6	21,591,725.0
in_retweet_timedelta_sec	6,887	1,150.15	14,274.44	0.0	0.0	0.0	440.07	793,138.2
in_quote_timedelta_sec	6,887	235.93	2,114.61	0.0	0.0	0.0	49.93	82,308.88

2.5.2 Exploratory Data Analysis

EDA applications include data preprocessing and data visualization techniques to provide valuable data insights and allow for different types of Data Analysis. Data preprocessing was an integral part to the conducted analysis as it provided deeper insights into User interactions with Tweets, by aggregating tweets data. Produced aggregations expanded the original User and Tweets objects with information about the user's outbound (how many tweets they published and count of favorites the user gave out) and inbound (how many other users retweeted or quoted this user and count of favorites the user received) interactions, supporting drill-downs through date and time, hashtags and user mentions. Twitter object 6. describes a User object after it was preprocessed.

Twitter objects 6: Preprocessed User object

```
{
  "screen_name": "HNS_CFF",
  "followers_count": 253625,
  "friends_count": 117,
  "original_tweets_cnt": 670.0,

  "total_out_tweets_cnt": 779.0,
  "out_retweet_cnt": 109.0,
  "out_reply_cnt": 2.0,
  "out_quote_cnt": 119.0,
  "out_retweet_timedelta_sec": 4978.899083,
  "out_quote_timedelta_sec": 18670.231707,

  "total_in_tweets_cnt": 1248.0,
  "in_retweet_cnt": 1266.0,
  "in_reply_cnt": 204.0,
  "in_quote_cnt": 374.0,
  "in_retweet_timedelta_sec": 5720.02172,
  "in_quote_timedelta_sec": 2261.416613,

  "in_original_favorite_cnt": 209617.0,
  "in_retweets_favorite_cnt": 277942.0,
  "in_quotes_favorite_cnt": 418973.0,

  "original_hashtags":  [ "FIFAWorldCup", "Family"],
  "retweet_hashtags":   [ "MiaSanMia", "FCBayern"],
  "quote_hashtags":     [ "ForzaInter", "dinamozagreb"],

  "original_user_mentions": [ "lukamodric10", "DalicZlatko"],
  "retweet_user_mentions":  [ "HNS_CFF",   "lukamodric10"],
  "quote_user_mentions":    [ "lukamodric10",   "staderennais"]
}
```

2.5.3 Graph Analytics

Graph analytics, or Graph algorithms, are analytic tools used to determine the strength and direction of relationships between objects in a graph. The focus of graph analytics is on pairwise relationships between two objects at a time and structural characteristics of the graph as a whole[10]. A graph data structure comprises a distinct set of *nodes* (often referred to as *vertices* or *points*) and a sequence of *edges* (also referred to as *links* or *lines*), where each *edge* contains a pair of nodes $(node_i, node_j)$. Various types of graphs exist based on the representation of an *edge*. This thesis focuses on directed and undirected graphs; **directed** - where Users are presented by *nodes*, with *edges* representing a "retweet" relationship between the Users (if $user_i$ retweets $user_j$, that does not mean $user_j$ retweeted $user_i$); and **undirected** - where Hashtags are presented by *nodes*, with *edges* representing a "mutually shared" relationship between each pair of Hashtags in a Tweet. The number of relationships a *node* is a part of is denoted as the *node's* degree. Degrees differ by the relationship type - the number of Incoming links a node has is denoted as **in-degree** and the number of Outgoing links a node has is denoted as **out-degree**.

The following paragraphs describe some additional measures used to quantify the analysed graph.

Density Graph density measures how many edges are close to the maximum number of edges (where every pair of vertices is connected by one edge). The opposite, a graph with only a few edges, is a sparse graph. Depending on the size of the graph and techniques used to manipulate it's size, density can vary.

Betweenness Centrality Betweenness centrality is a measure of centrality in a graph based on shortest paths. Nodes with higher degrees are more centered in a graph than nodes with lower degrees.

PageRank PageRank is an algorithm used by Google Search to rank web pages in their search engine results. It works by counting the number and quality of links to a node to determine a rough estimate of how important the node is. The underlying assumption is that more important nodes are likely to receive more links from other nodes[11].

Clustering Coefficient A node's clustering coefficient measures the degree to which nodes in a graph tend to cluster together. A high clustering coefficient signals that there are a lot of nodes clustered around the observed node, while a low clustering coefficient signals that the observed node is isolated, it is not close to other nodes in the network.

Distance Graph distance is measured by a finite or infinite sequence of edges which joins a sequence of distinct nodes. A "walk" from one User to another is accomplished by following a path created by the User's relationships. If a User at the other end of the relationship has relationships with a third User, the initial "walk" is extended by one more path.

Analysis results are captured and described in the Results section.

2.6 CI/CD Overview

CI/CD is a software development practice that involves a continuous cycle of building, testing, and deploying software applications. The acronym CI/CD stands for Continuous Integration/Continuous Delivery (or Deployment).

CI refers to continuous integration, which is an automation process for developers. Successful CI means changes to software is regularly built, tested, and merged to a shared code repository. **CD** is the practice of automatically building, testing, and deploying code changes to a production environment. This allows for a quick and confident delivery of new features and updates to the target system. CI/CD mitigates risks of errors and delays that often occur when deploying code manually.

This *data platform*'s CI/CD practice uses a **Git Flow**[15] branching strategy on GitHub. It comprises two long-lived branches: *develop* and `main` (denoted as *production*), short-lived branches like `feature` and `hotfix`, driven by GitHub Actions. Additionally, a scheduling trigger is set up using GitHub Actions to run the data pipeline. This strategy is not being validated by an automated process, it is a verbal convention recommended for a supported, easily managed code repository.

Enhancements and bug fixes are developed inside `feature` and `hotfix` branches that developers create prior to starting development. After **development** and **unit testing** is completed, a **pull-request**[16] from the short-lived branch to *develop* is opened on GitHub, requesting to merge the short-lived branch to the long-lived branch. Once the **pull-request** is **merged**[17], an automated process versions the software and tags the code repository using **semantic release** mechanisms. At the time of writing this thesis, an automated testing process does not exist. Changes are now deployed to *production* by merging *develop* to `main`.

[15] https://www.gitkraken.com/learn/git/git-flow

[16] A mechanism for a developer to notify team members that they have completed a development effort [12]

[17] "Pull Request Close" action in a GitHub repository is a GitHub Action event trigger

3 Results

3.1 Overview

Data analysis was conducted on multiple data views: *Tweets*, *Users* and *Hashtags*. The difference between them is the way they are aggregated: *Tweets* view is aggregated by `week`; *Users* are aggregated by `user_id`; *Hashtags* are exploded[18] and aggregated by `hashtags` (content afterward).

Analysis begins by describing what kinds of reactions are happening between Croatian Users on Twitter, and who are the Users driving these reactions. Reactions are categorized into `Original Tweets`, `Retweets`, `Replies` and `Quotes`. Original Tweets represent all tweets that are not Retweets, Retweets represent tweets shared by other users *without* additional content from the retweeting User to the topic being shared. Quotes represent tweets shared by other users *with* additional content from the quoting User to the topic being shared (a Quote can contain Retweet's content, but a Retweet cannot contain Quote's content), and Replies represent replies to tweets. Additional concepts of **Incoming links** and **Outgoing links** (**reactions**) are introduced (`Incoming Retweets`, `Incoming Replies` and `Incoming Quotes`). An incoming link represents the number of times external users interacted with the User in focus, while an outgoing link represents the number of times the User in focus posted a reaction category.

After the reaction categories are quantitatively described, the content (`hashtags`) behind those interactions is identified in 3.4. Once the most popular content is described, outliers ("spam") are detected and filtered to support a resilient qualitative analysis. Graph analysis is introduced to identify and analyse relationships between *Users* and *Hashtags*. Finally, information sharing is visualized as the number of times a `hashtag` has been shared in a given time range and information spread is visualized as the `hashtag` percentage share in the total number of Tweets in the collected network.

The analysis is conducted on a total number of $386,168$ Tweets ranging from `2022-11-01` to `2022-11-30`, including $6,887$ Croatian Users. The total number of Tweets includes $265,155$ `Original Tweets` ($165,719$ `Replies` and $24,170$ `Quotes`) and $121,013$ `Retweets`. Incoming links are represented by $7,034$ `Incoming Retweets`, $32,728$ `Incoming Replies` and $2,120$ `Incoming Quotes` (one original Tweet can have multiple incoming links).

[18] The term "explosion" in this context refers to extracting multiple values from a given column into multiple table records such that each value from the column can be uniquely identified by the new table record (row); this process is the opposite of data aggregation

3.2 Tweets

The *Tweets* view is aggregated by `week`, enabling fine data granularity. Figure 3.2 shows what reactions are Croatian Users on Twitter mostly engaged with in the weeks of November, 2022.

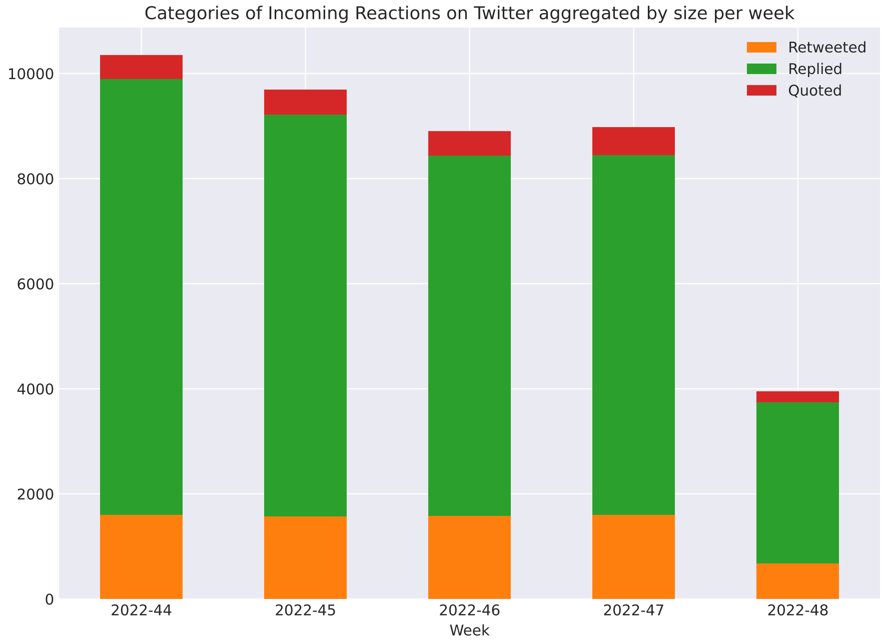

Figure 4: Categories of Incoming Reactions on Twitter aggregated by size per week

This visualization shows that the largest amount of Users is engaged in *Reply* reactions - however, the *Hashtags* analysis shows that most content is shared using the *Retweet* reactions, disregarding that most incoming reactions are really *Replies*.

3.3 Users

This view is aggregated by `user_id`, disabling a fine time aggregation granularity but enabling a total overview of a User's reactions. All visualizations based on the Users view display the total amount of a User's reactions in the given time range and his collected Tweets. Figure 3.3 shows which Users are the incoming reaction initiators.

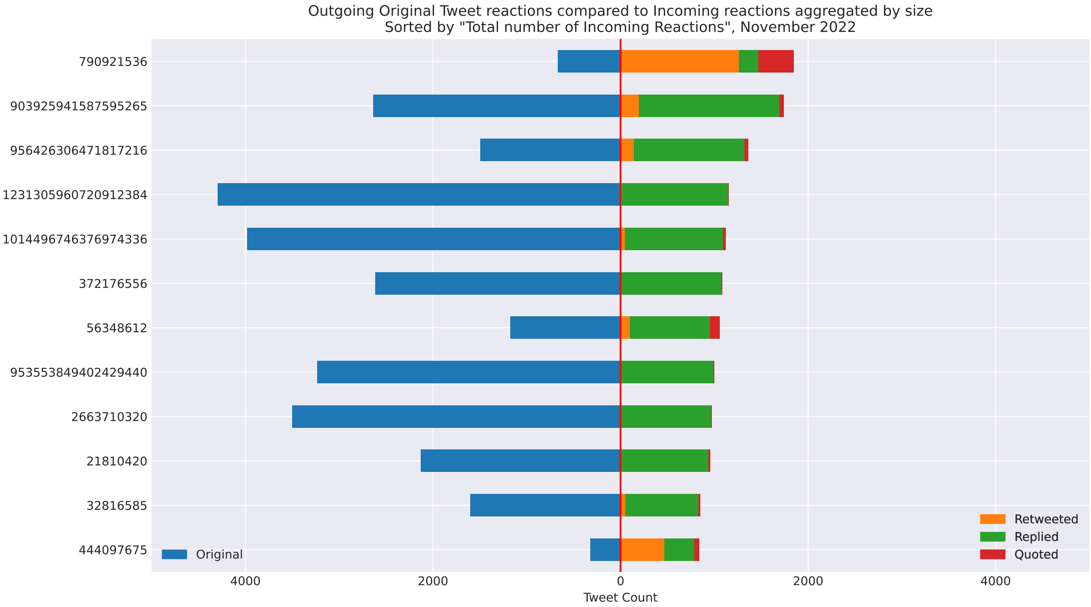

Figure 5: Outgoing Original Tweet reactions compared to Incoming reactions aggregated by size

This visualization shows the categories of reactions that the initiators participate in and it confirms that the largest amount of Users is engaged in *Reply* reactions. See the amount of users engaged in *Retweets* in Appendix A.

To understand how much users are involved in each reaction categories, figure 3.3 shows the percentage distribution of a User's involvement in `Outgoing Original Tweets`, `Incoming Retweets` and `Incoming Replies`.

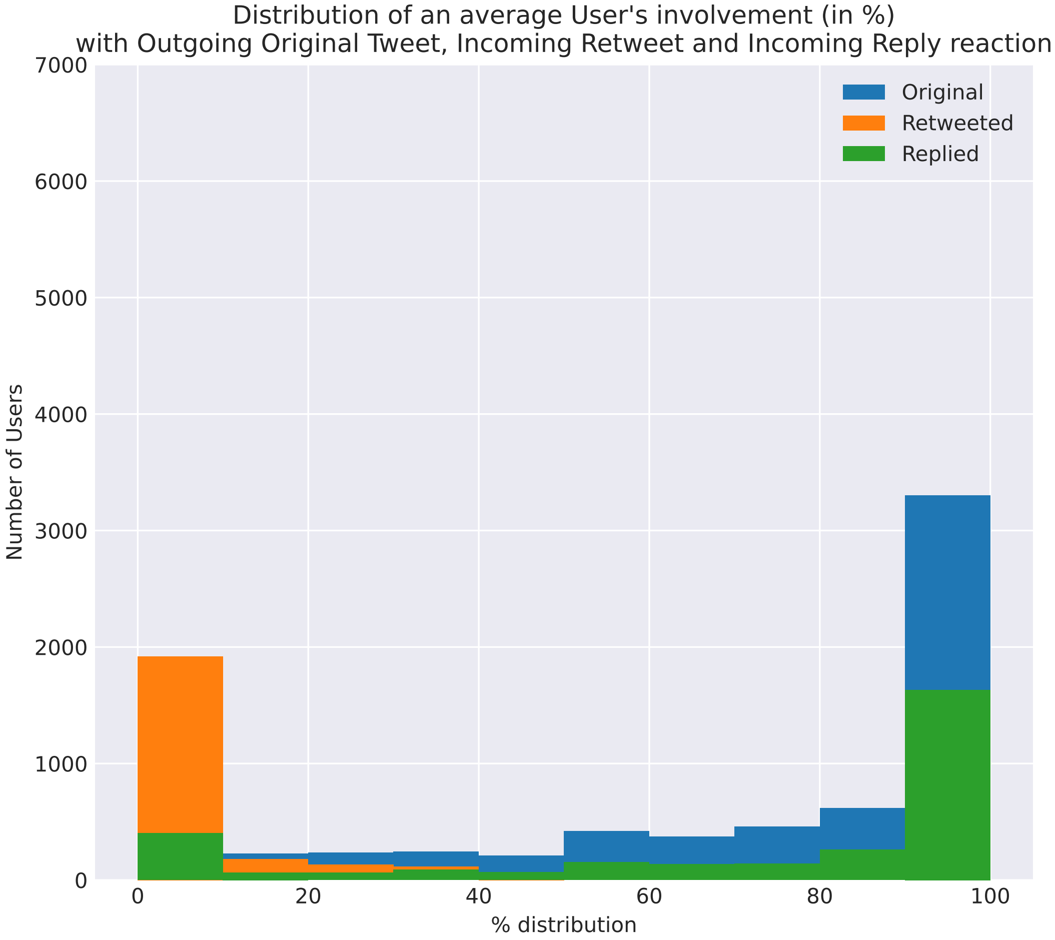

Figure 6: Distribution of an average User's involvement (in %) with Outgoing Original Tweets, Incoming Retweets and Incoming Reply reactions

This visualization shows how most Users on Twitter mostly tweet `Original Tweets` and how most Users get replied to (`Incoming Replies`) (more than $3,000$ Users). About $1,500$ Users' tweets get retweeted (`Incoming Retweets`), in a very small amount.

3.4 Hashtags

Previous results have shown what are the reactions on Twitter that Users are most likely to engage with. This section analyses what content is being shared using the described reactions. Figure 3.4 shows what content is being shared the most, sorted by the number of total incoming reactions.

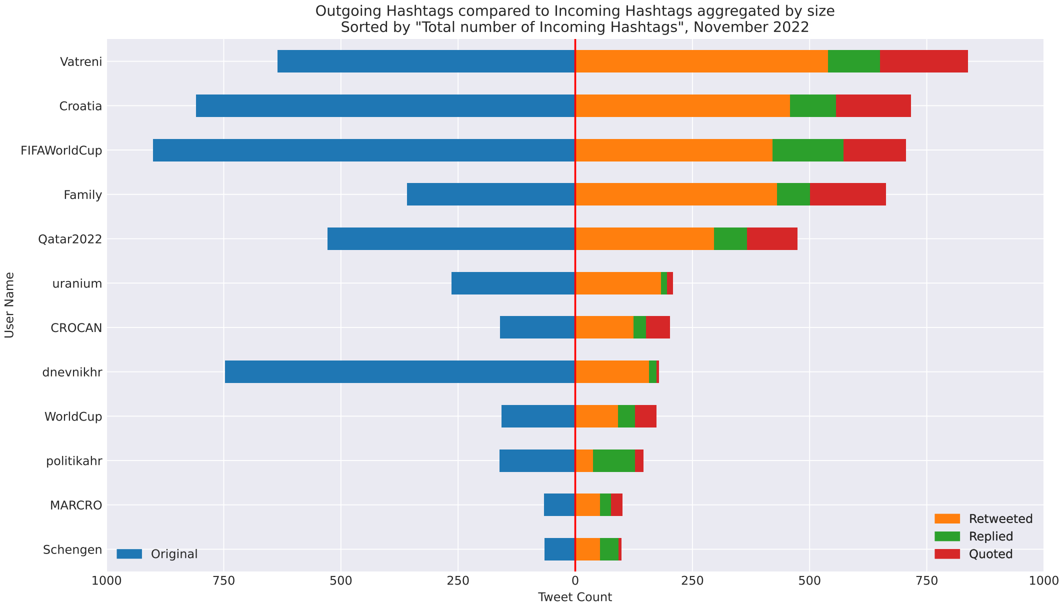

Figure 7: Outgoing Hashtags compared to Most Popular Incoming Hashtags aggregated by size

This visualization confirms that the most popular content is spread using the *Retweet* reaction. The spread of the most popularly shared content, `Vatreni` hashtag, is related to the FIFA World Cup which is taking place in the analysed time span. It is important to note that this analysis is short-sighted because of the short time span it covers. More visualizations on reaction categories are available in Appendix A.

While performing this analysis, some content was identified as "spam", i.e. there is a specific group of Users who continuously share the same content, without external users sharing that same content. This behavior causes a quantitative bias where some content appears to have a large number of incoming reactions, even though they are all coming from the same User and are not being shared across the network. To identify such biases, additional data features are introduced - α (`alpha`) and β (`beta`).

α and β are calculated using the unique number of Users who at least once shared the hashtag, and the total number of times that the hashtag was shared. The collected data is referenced as D, Users are referenced as $u \in D_u$, and Hashtags are referenced as $h \in D_h$:

$$HU = \{(h, u) \mid h \in D, u \in D\}$$
$$w(h)\colon = \#i\colon (h, i) \mid h \in HU$$
$$w(hu)\colon = \#i\colon (h, u, i) \mid (h, u) \in HU$$

$$\alpha = \{w(hu)/w(h) \mid (h, u) \in HU, h \in D_h\}$$
$$\beta = \{w(hu)/\sum_{i=0}^{|D_h|} w(h) \mid (h, u) \in HU, h \in D_h\}$$

Finally, α and $\beta$are normalized.

$$\alpha = \{a\colon \alpha * (1/max(\alpha)) \mid a \in \mathbb{R}, a \geq 0.0, a \leq 1.0\}$$
$$\beta = \{b\colon \beta * (1/max(\beta)) \mid b \in \mathbb{R}, b \geq 0.0, b \leq 1.0\}$$

Figure 3.4 displays α as the normalized number of unique Users who shared the content, β as the normalized ratio a content has in the entire set of available contents. Popular content that is shared among a large number of users has a high α, while content shared among a small number of Users has a low α. Popular content that is included in the majority of the content being shared has a high β, while content that is included in the minority of the content being shared has a low β.

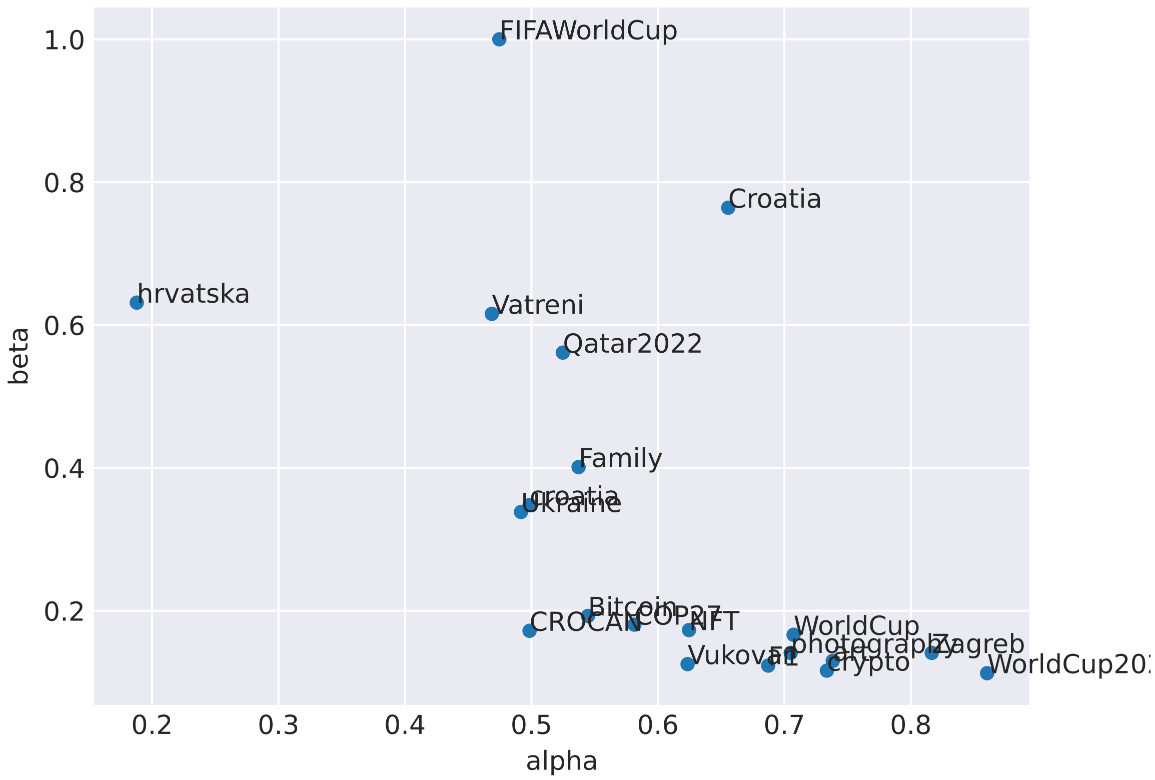

Figure 8: Hashtag Popularity based on the derived *alpha* and *beta* (normalized) features

This figure shows that the content `Vatreni` is shared among 50% of users and it is included in 60% of the content being shared. On the other hand, contents `WorldCup` and `photography` are shared among 75% of users, but it is included only in < 20% of the content being shared.

At the time of writing this thesis, an automated process of topic detection is not implemented, so topics are manually derived using the observations from preceding figures, and defined as: "World Cup", "Ukraine/War", "Schengen" and a minority of others such as "Crypto" or "SpotifyWrapped" topics. These topics can now be fine-grained by their related content, that is `FIFAWorldCup`, `CROCAN`, `MARCRO`, `Vatreni` and similar for "World Cup", `Ukraine` for "Ukraine/War" and so forth. The content is additionally fine-grained by "day" and "week". Figure 3.4 shows the number of hashtag occurrences per day in November, 2022.

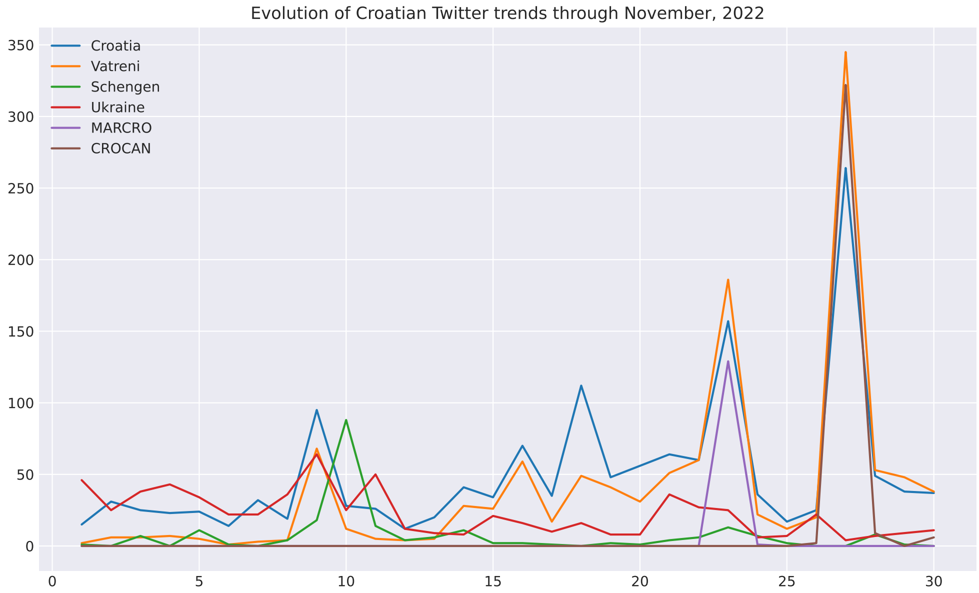

Figure 9: Evolution of Croatian Twitter trends through days in November, 2022

This figure shows certain hashtag spikes on specific days, like the spike of `Ukraine` content happening on November 9th, when Russia backed out their troops from Kherson and the EU offered a significant financial support for Ukraine in 2023[13], the spike of `Schengen` and `Croatia` content happening on November 10th when Croatia's Schengen request was approved[14], and the spike of `Vatreni`, `MARCRO` and `CROCAN` content happening at the time with World Cup matches between Morocco and Canada versus Croatia (November 23rd, November 27th respectively). Additional analysis on content changing through time is not created as a part of this thesis.

3.5 Graph Analysis

Previous sections provided insights into the contents' quantity and ways that they are shared throughout the network. This sections aims to provide an insight into the relationship between *Users* and *Hashtags*, and finally about the way content is spread throughout the network.

In the following network, a directed graph is presented with *nodes* representing Users who tweet specific content by *edges* where an edge ($user_i$, $user_j$) is represented by an arrow, pointing from the User whose tweet was Retweeted to the User who Retweeted the original tweet (an **inverse** Incoming link). Greater arrow thickness represents greater number of Retweets between $user_i$ and $user_j$. The nodes are colored by their clustering coefficient, which is discussed later in this section.

The number of available tweets in the network increases every day, while the analysis created as a part of this thesis only displays information from a fixed time range. By running this same analysis on a dataset in a different time range, the results could change dramatically. This effect makes this network a **temporal network**, also known as a **time-varying network**.

Figure 3.5 shows the spread of the `Vatreni` content. The node that was Retweeted the most (most Incoming links) is positioned in the center of the graph, it represents the mainstream medium.

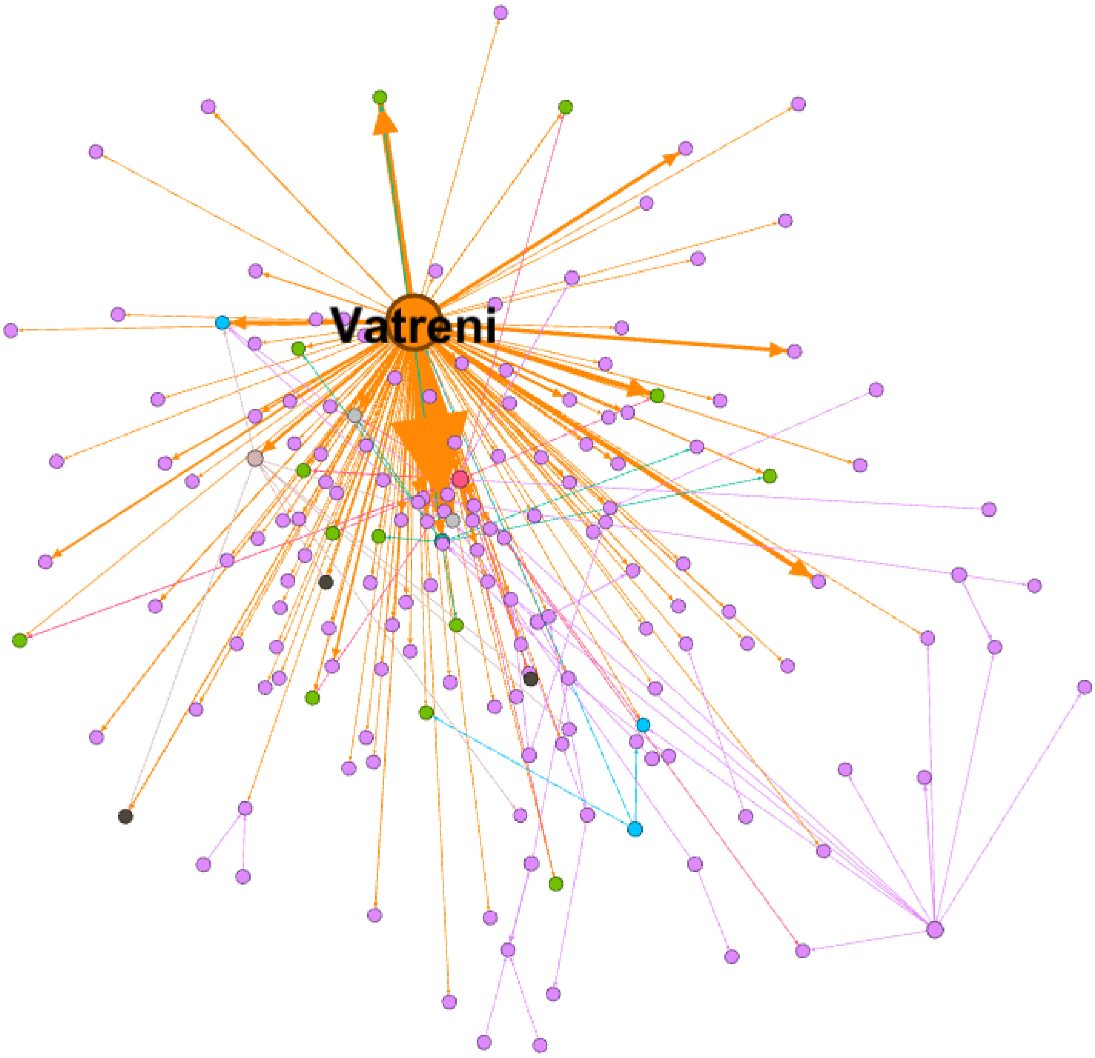

Figure 10: Hashtag Spread across all Users who retweeted Vatreni

Figure 3.5 shows the spread of `crypto` content. The center of the graph is a node that was retweeting the most (most inverse Outgoing links), the leaflet distributor.

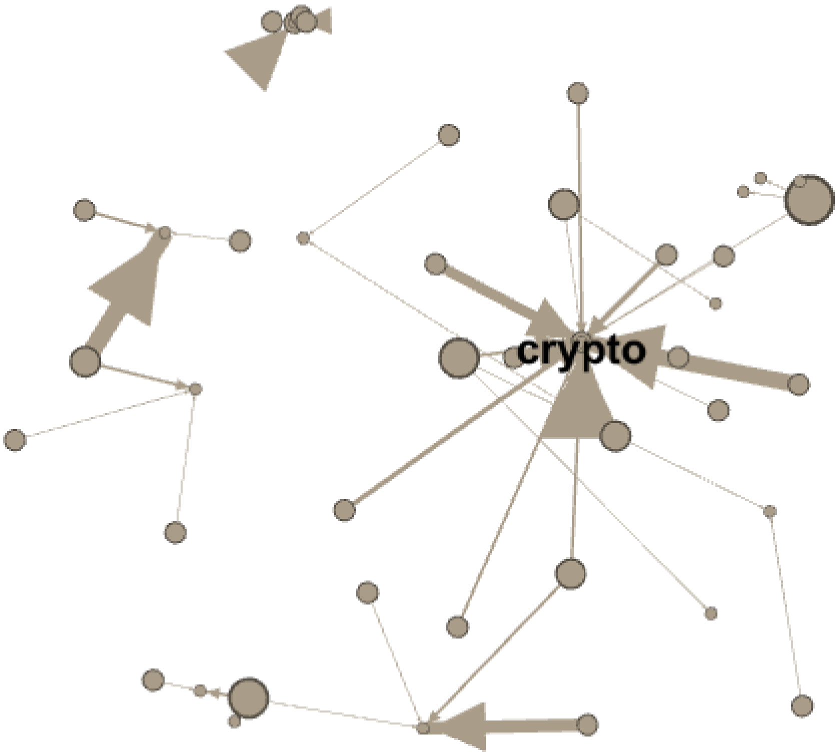

Figure 11: Hashtag Spread across all Users who retweeted crypto

The main differences between these two graphs are their in and out-degrees and their clustering coefficients. `Vatreni` content was shared by a lot of Users who would usually retweet it from the same User. The User tweeting `Vatreni` has a high in-degree and low out-degree with a high clustering coefficient that positions the User in the middle of the displayed cluster of nodes. In contrast, `crypto` content was shared by a small number of Users from different sources. The Users tweeting `crypto` have a low in-degree and high out-degree with a low clustering coefficient. This pattern is explored using a correlation matrix.

The correlation matrix in figure 3.5 shows correlations between graph measures filtered by `FIFAWorldCup`, `Vatreni`, `Croatia`, `Ukraine`, `Schengen` and `CP027` (subgraphs representing Retweets between Users who tweeted the selected hashtags). Prior to plotting the matrix, an additional feature *popular* was added to the matrix representing the popularity of a hashtag by a boolean variable. Content `FIFAWorldCup`, `Vatreni` and `Croatia` is labeled "popular", while `Ukraine`, `Schengen` and `CP027` are labeled "not popular". Label definitions are inspired by the differences identified between figures *Vatreni* 3.5 and *crypto* 3.5.

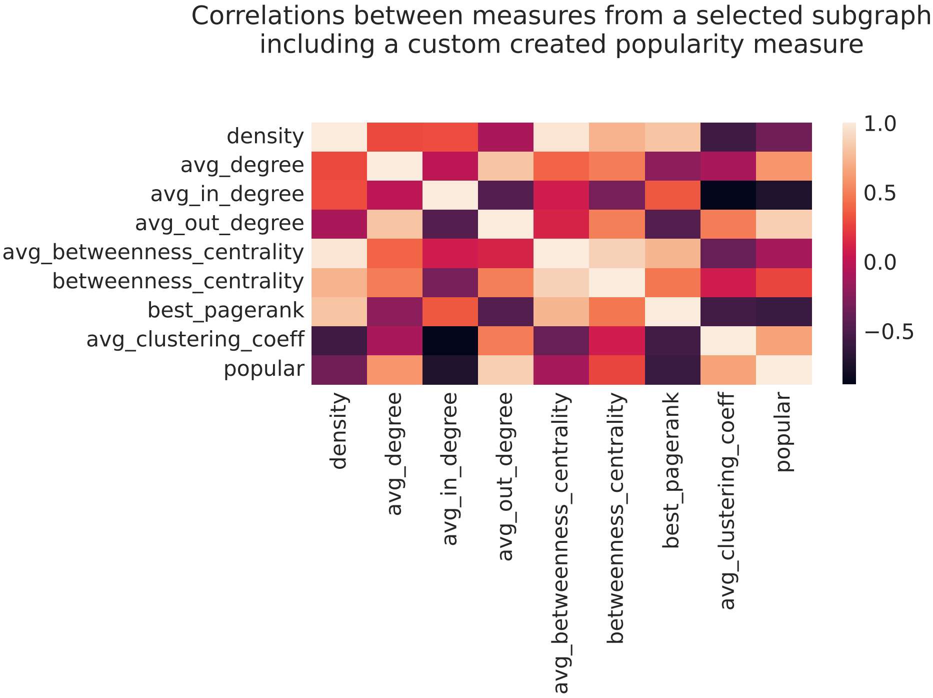

Figure 12: Correlations between measures describing selected subgraphs that represent "popular" or "not popular" content

Following features share the most extreme correlations with the *popular* feature: *avg_ out_ degree* (average Incoming number of links; 85.34% correlation), *avg_ clustering_ coeff* (average clustering coefficient; 65.07% correlation) and *avg_ degree* (average Incoming and Outgoing number of links; 59.35% correlation). These features attribute to the *popular* feature's positivity. On the other hand, features *avg_ in_ degree* (average Outgoing number of links; −73.43% correlation) and *best_ pagerank* (node "importance"; −59.65% correlation) attribute to the *popular* feature's negativity.

The entire correlation matrix is available in Appendix A.

4 Conclusion

Information spread on Twitter is a broad topic that can be analysed in different ways. This thesis describes the methods used to create a *data platform* that continuously collects Twitter data and creates a dataset that can be used to gain various insights. The methods discussed include the system's development life cycle and the specific software process model that was used to plan and develop a robust data pipeline. Data Ingestion and Data Transformation methods are then discussed, followed by Data Analysis methods and an overview of CI/CD.

Finally, the results provide relevant insights from the dataset created by defining, categorizing, and quantifying the available responses on Twitter from the perspective of time, users, and content, as well as from the perspective of relationships between users and content. They support related work by recognizing the most common category of reaction sharing on Twitter - Retweets. Graph Analytics shows how popular content (`Vatreni`) is shared compared content that is not so popular (`crypto`). The User sharing popular content is very similar to a mainstream medium - they have a wide reach (number of reverse Incoming links) and are well connected with their audience (clustering coefficient). In contrast, the Users who share not-so-popular content have a short reach and are not well connected with their audience. They represent the leaflet distributors, and their leaflets are not interesting for the people walking next to them. The network that these Users represent resembles a network that connects only the distributors, with very few consumers.

Retweets are useful for answering the question of how information is spread in a network, but they don't answer the question of why it is spread. Graph Analytics attempts to provide an answer by using a correlation matrix to identify features attributing to information popularity, but the given answer is short-sighted. The network analysed is a temporal network with one month's worth of data, so the insights gained can be biased by some trends that are unique to the month analysed. Once the analysis is performed on a larger dataset, biases and patterns can be identified and interpreted as an overall picture of information spread.

This thesis serves as a starting point for further research that utilizes the collected data. Existing analysis can be extended or improved in numerous ways, by tracking reaction and content changes over time, or by creating a graph structure in which users represent nodes and a *following* relationship represents a link between users. In such a network, each user shares their own sets of content, which may or may not be similar to the contents of the users around them. By examining the content and comparing it to the content of their *followers* (or *followees*), information paths can be constructed to provide a better view of information spread.

References

[1] Barr Moses and Lior Gavish. What is a data platform? and how to build one (montecarlodata.com), Nov 2022.

[2] Mile Pavlić. *Information systems.* Odjel za informatiku Sveučilišta u Rijeci, 2009.

[3] Sanja Čandrlić. Uvod u programsko inženjerstvo, 2019/2020.

[4] Milan Petrović, Andrea Hrelja, and Ana Meštrović. Prediction of covid-19 tweeting: classification based on graph neural networks. *MIPRO*, 45:307–311, 2022.

[5] John Meehan, Cansu Aslantas, Stan Zdonik, Nesime Tatbul, and Jiang Du. Data ingestion for the connected world. In *CIDR*, 2017.

[6] Meta S Brown. *Transforming unstructured data into useful information.* Auerbach Publications, 2014.

[7] Nancy Leech, Karen Barrett, and George Morgan. *Data Coding and Exploratory Analysis (EDA) Rules for Data Coding Exploratory Data Analysis (EDA) Statistical Assumptions.* Routledge, Jan 2015.

[8] P.S. Mann. *Introductory Statistics.* Wiley, 1995.

[9] Ana Meštrović, Milan Petrović, and Slobodan Beliga. Retweet prediction based on heterogeneous data sources: The combination of text and multilayer network features, 2022.

[10] Nvidia. What is graph analytics? (nvidia.com), Dec 2022.

[11] Wikipedia. Pagerank (wikipedia.com), Dec 2022.

[12] Atlassian. Making a pull request (atlassian.com), Dec 2022.

[13] euronews. Ukraine war: Russia orders troops to quit kherson; eu €18bn aid offer; mariupol mass graves (euronews.com), Nov 2022.

[14] Reuters. Croatia gets green light for schengen admission from european parliament (reuters.com), Nov 2022.

[15] Red Hat. What is a REST API? (redhat.com), Nov 2022.

[16] Atlassian. What is a Agile? (atlassian.com), Nov 2022.

[17] Salisu Borodo, Siti Mariyam Shamsuddin, and Shafaatunnur Hasan. Big data platforms and techniques. *Indonesian Journal of Electrical Engineering and Computer Science*, 1:191, Jan 2016.

[18] J. Efrim Boritz. Is practitioners' views on core concepts of information integrity. *International Journal of Accounting Information Systems*, 6(4):260–279, 2005.

[19] Tableau. Data cleaning: Definition, benefits, and how-to (tableau.com), Dec 2022.

[20] Law Insider. Data service provider definition (lawinsider.com), Nov 2022.

[21] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, Third Edition.* The MIT Press, 3rd edition, 2009.

[22] L. Wirzenius, J. Oja, S. Stafford, and A. Weeks. *The Linux System Administrator's Guide.* CreateSpace Independent Publishing Platform, 2007.

[23] Wikipedia. Github (wikipedia.com), Dec 2022.

[24] Wikipedia. Lambda Architecture (wikipedia.com), Nov 2022.

[25] Richard Wang. streaming data vs. real time data — what's the difference? (medium.com), Jan 2022.

[26] J Martin Bland and Douglas G Altman. Statistics notes: Measurement error. *BMJ*, 312(7047):1654, 1996.

[27] Amazon. What is streaming data? (aws.amazon.com), Nov 2022.

[28] SmartBear. What is unit testing? (smartbear.com), Dec 2022.

List of Figures

List of Tables

Acronyms

API Application Programming Interface. 4

CDA Confirmatory Data Analysis. 20

CRUD Create, Read, Update, Delete. 46

EDA Exploratory Data Analysis. 20, 21

HTTP Hypertext Transfer Protocol. 47

JSON JavaScript Object Notation. 6, 15

RAD Rapid Application Development. 3

RAM Random Access Memory. 47

REST Representational State Transfer. 4, 13, 14, 41, 46

RUP Rational Unified Process. 3

SDLC System Development Life Cycle. i, 2, 3, 12, 43

Glossary

REST API (also known as RESTful API) is an [A]pplication [P]rogramming [I]nterface that allows for interaction with RESTful web services, created by computer scientist Roy Fielding [15]. 13, 41

data platform A central repository and processing house for all of an organization's data. i, 2–5, 10, 12–14, 25, 39, 43

Agile An iterative approach to project management and software development [16]. 3, 41

big data Data that is distinguishable by specific attributes ("Big V's", volume, velocity, variety, veracity, value, ...) [17] . 13, 14

branch protection rules A feature available as a part of GitHub services that allows setting up security constraints on CRUD and other manipulation with the repository. 3

bug An error, flaw or fault in computer software. 2, 12

code repository File archive and web hosting facility where programmers, software developers, and designers store large amounts of source code for the software and/or web pages for safekeeping. 25

Data Analyst A person performing exploratory and statistical analysis on a dataset. 14, 16, 20

data availability A constant availability of time-grained data sourced from an extended time period. 13

data integrity The maintenance and the assurance data accuracy and consistency over its entire life-cycle[18]. 3

data pipeline Series of data processing steps where the output of one element is the input of the next one. 2, 4, 10, 11, 15, 25, 39

data preprocessing Data manipulation process used to ensure or enhance analysis performance [19]. 21

data service provider A natural or legal person who provides information and communication technology services to a declarant in relation to reporting obligations [20]. 4, 5

data structure A data organization, management, and storage format that is usually chosen for efficient access to data [21]. 47

data view A dataset that contains a subset of available information. 16, 20, 26

data visualization A pictorial representation of data. 21

enhancement Software enhancements and tweaks to existing features. 2, 12

event-based A triggering mechanism that starts on an event (usually an HTTP request method). 13

feature An options or functional capability available to the user; a distinguishing characteristic of a software item. 2, 12, 47

file system A method and data structure that the operating system uses to control how data is stored and retrieved [22]. 13

flat file A type of database that stores data in a plain-text format. 13, 47

GitHub An Internet hosting service for software development and version control using Git [23]. 25, 46, 47

GitHub Actions [19] A deployment automation service issued by GitHub. 25

in-memory Term used to describe information stored in a computer's Random Access Memory (RAM). 10

JavaScript Object Notation A lightweight data-interchange format; type of flat file. 45

Lambda Architecture Architecture designed to handle massive quantities of data by taking advantage of both batch and real-time processing methods [24]. 13, 42

[19] https://github.com/features/actions

loosely coupled An approach to system design where individual components depend on each other to the least extent practicable. 12

matplotlib Comprehensive library for creating static, animated, and interactive visualizations in Python. 10

operating system (OS) System software that manages computer hardware, software resources, and provides common services for computer programs. 13, 47

pandas Flexible and easy to use open source data analysis and manipulation tool, built on top of the Python programming language. 10

Python Programming language, convenient for data processing. 2

real time data Data that is generated within specific time constraints [25]. 13, 14, 42

schedule-based A triggering mechanism that starts at a given point in time. 13

semantic release [20] A process that creates a new (incremental) software version by determining if a new version is needed by reading commit messages that follow a set of formalized conventions. 25

standard deviation A measure of the amount of variation or dispersion of a set of values [26]. 21

streaming data Data that is generated continuously by thousands of data sources, which typically send in the data records simultaneously, and in small sizes (order of Kilobytes) [27]. 13, 42

temporal network A network whose links are only active at certain points in time. 34

Tweet object [21] Twitter Tweet object model. 5

[20] https://github.com/semantic-release/semantic-release

[21] https://developer.twitter.com/en/docs/twitter-api/v1/data-dictionary/object-model/tweet

unit testing A validation technique that asserts accuracy on the smallest piece of code that can be logically isolated in a system [28]. 25

User object [22]Twitter User object model. 5

[22] https://developer.twitter.com/en/docs/twitter-api/v1/data-dictionary/object-model/user

A Appendix

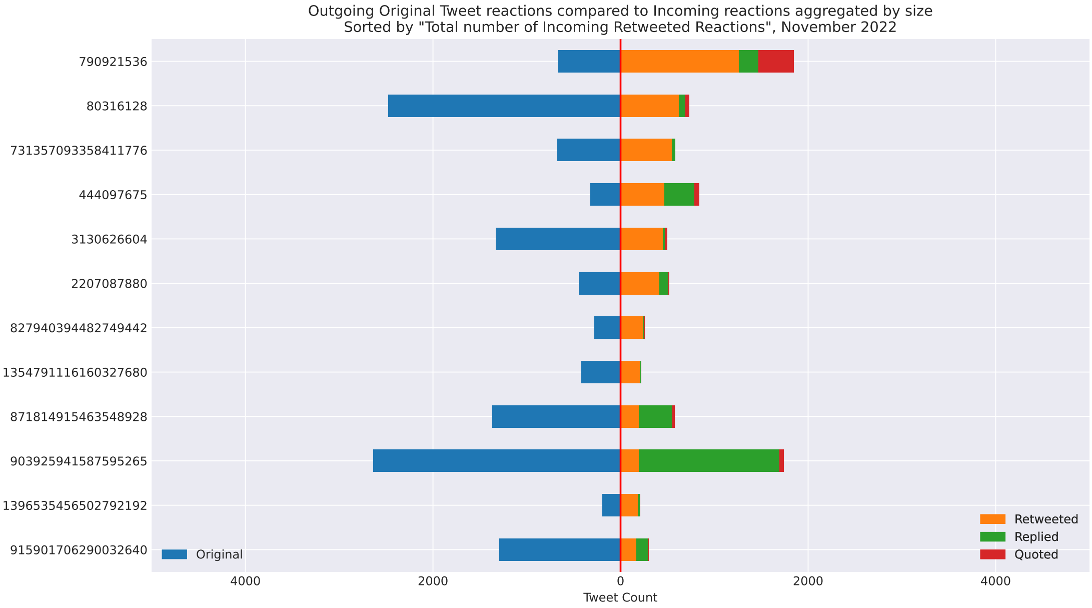

Figure 13: Outgoing Original Tweet reactions compared to Incoming reactions aggregated by size. Referenced by: 3.2

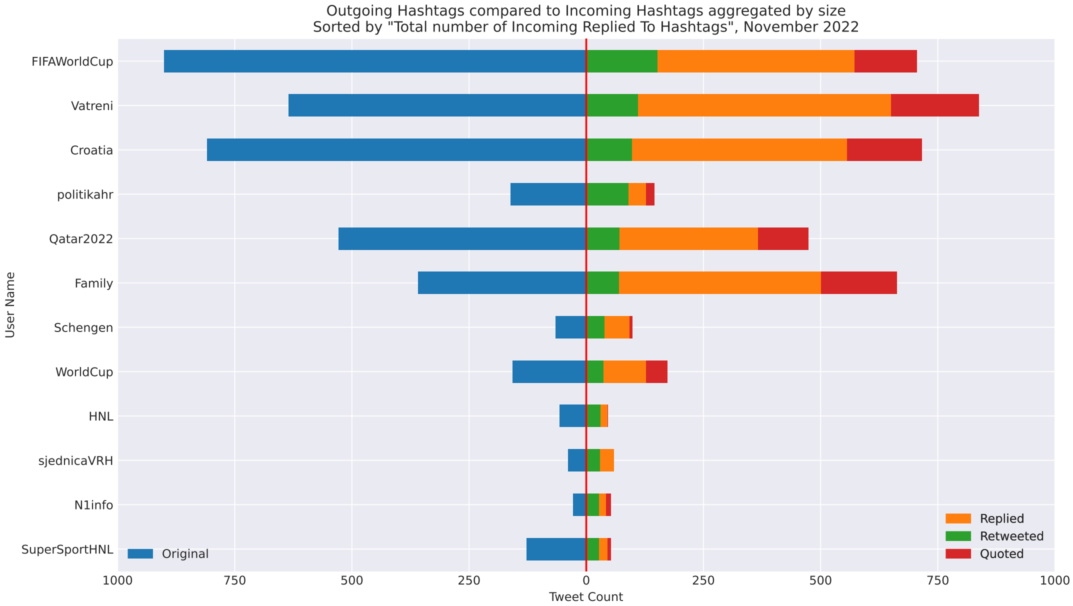

Figure 14: Outgoing Hashtags compared to Incoming Hashtags aggregated by size. Referenced by: 3.4

Table 3: Correlations between measures describing selected subgraphs that represent "popular" or "not popular" content (1/2)

	avg_degree	avg_in_degree	avg_out_degree	avg_betweenness_centrality	**popular**
avg_degree	1.00	-0.0127	0.8033	0.3901	0.5936
avg_in_degree	-0.0127	1.00	-0.4716	0.0727	-0.7343
avg_out_degree	0.8033	-0.4716	1.00	0.1195	0.8535
avg_betweenness_centrality	0.3901	0.0727	0.1195	1.00	-0.1093
avg_clustering_coeff	-0.1040	-0.8858	0.4877	-0.3781	0.6508
betweenness_centrality	0.4882	-0.3122	0.4943	0.8708	0.2703
best_pagerank	-0.2160	0.3393	-0.4805	0.7365	-0.5966
density	0.2799	0.2963	-0.0845	0.9650	-0.3435
popular	0.5936	-0.7343	0.8535	-0.1093	1.00

Table 4: Correlations between measures describing selected subgraphs that represent "popular" or "not popular" content (2/2)

	avg_clustering_coeff	betweenness_centrality	best_pagerank	density	**popular**
avg_degree	-0.1040	0.4882	-0.2160	0.2799	0.5936
avg_in_degree	-0.8858	-0.3122	0.3393	0.2963	-0.7343
avg_out_degree	0.4877	0.4943	-0.4805	-0.0845	0.8535
avg_betweenness_centrality	-0.3781	0.8708	0.7365	0.9650	-0.1093
avg_clustering_coeff	1.00	0.0857	-0.5542	-0.5694	0.6508
betweenness_centrality	0.0857	1.00	0.4665	0.7257	0.2703
best_pagerank	-0.5542	0.4665	1.00	0.8024	-0.5966
density	-0.5694	0.7257	0.8024	1.00	-0.3435
popular	0.6508	0.2703	-0.5966	-0.3435	1.00