Twitter Data Analysis

Applied Data Science in Python - Complete Report

Project 5

Sebastian Mohr

Student-ID: 3188691

Wintersemester 2022

1 Project Description

The goal of this project is to give users the ability to get information about a specific events hashtag and its top conversations on **Twitter**. The tweets for an event should be analyzed and the sentiment displayed for the user of the application. Additionaly the application shows the Top 10 hashtags and users associated with the chosen event. The user can further investigate the displayed data and look at the users, as well as their followers and the tweets.

1.1 Modules, data structure, tools

In this section I will explain how I accessed the data I needed for my application and how I structured the data. Also I will define which Tools, Application, Modules and IDE I used in the course of this project.

1.1.1 Getting the data

The data that will be shown in the application will be gathered through the Twitter developer API. To achive this I signed up to Twitter with a new account that registered for the developer tools. Twitter already provides libraries for Python to easiely access its API.

First the app has to authenticate itself, to make sure that it is connected to the already created Twitter account. After the authentication the data can be gathered through a search url, that contains all the query parameters. Here the search url tells the API which data the application wants to get. The query parameters specify different metrics for the tweets, like the tweet author, date and time of its creation or used hashtags.

For this problem the API call will first retrieve all tweets that contain the hashtag of the searched for event. After analyzing the tweets, the the Top 10 contributors account data will also be retrieved from the API.

1.1.2 Data structuring

The data provided by the Twitter API has to be processed somewhere in memory. To achive this goal, the data will be saved into objects of the **pandas** type `DataFrame`. `DataFrame`s have the advante of being able to store and filter large amounts of data in memory. Additionaly it can be easily exported to CSV. When saving the data on the hard drive it can be accessed every time when opening the application and only be updated when it is requested. This will boost the performance of the application, as it's not dependent on an internet connection for basic features and doesn't have to gather new data on every startup. The data can then still be updated in the background when clicking an update button for example. The application handles tweets and users, which will be saved in seperate `DataFrame`s.

1.1.3 Tools and Application

The backend of the application will be written in `Python`, as it provides good data analysing tools for the different problems of this project. To access the Twitter API I will use the `Tweepy` library. `Tweepy` provides several features to use all of the methods the Twitter API offers. For the **sentiment analysis** I will use the python library `TextBlob`. It's a text processing library which helps to analyze written text, which is perfect for analyzing many written tweets about a specific topic like a big event.

For the frontend I will use a web application, which will use several different `TypeScript` files. In these files I will build a User Interface with the library `React`. With `React` interactive web pages can be build that behave like a native running program would. To power `React` I will use `MUI` as the component library.

1.1.4 Modules and Algorithm

The application will be split between front- and backend. In the frontend, only the user interface and basic computation, while the backend will handle all the Twitter API calls, as well as the data processing and persistence.

The application will first gather a specific number of tweets for the event from the Twitter API. After saving and analyzing the data, the Top 10 users by interactions will be figured out. Their info will also be retrieved from the Twitter API. After one of the Twitter users is clicked, its profile will be retrieved from Twitter and its followers and

information will be displayed. The followers als can be viewed and their tweets will also be retrieved from the Twitter API.

1.1.5 IDE

To develop the application I used two IDEs from JetBrains. For the Python backend I used the PyCharm IDE, as it provides good features like automatic imports and code completion. The same can be said for WebStorm, which I used to develop the frontend. Both of the IDEs have the same keybindings and plugins, which makes it very easy to work with both of them at the same time.

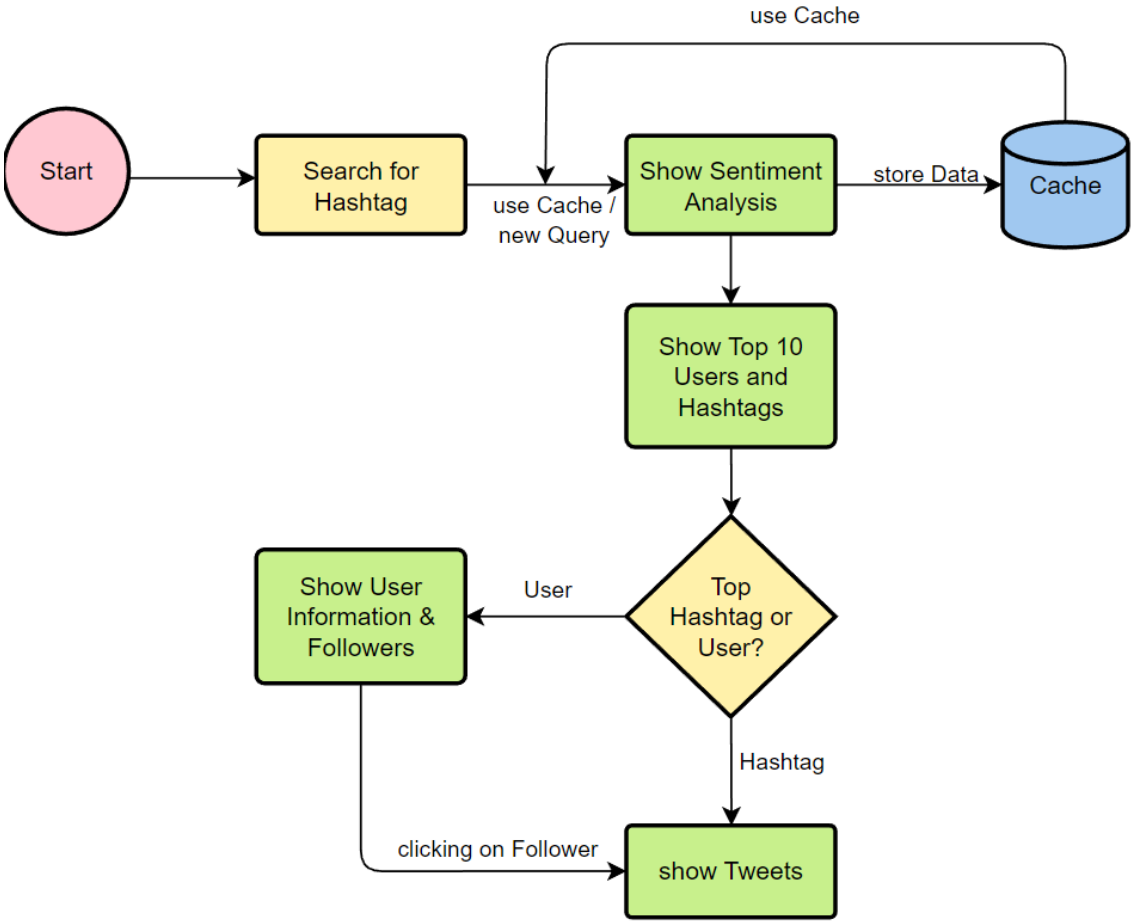
2 High Level Design

The application is split between front- and backend. The backend handles most of the data loading, transformation and persistence. To get the data from the backend, it runs on a simple Web-Api, that provides a few endpoints where data can be accessed. The frontend then calls these endpoints and retrieves the data through JSON objects, which can be easily deserialized in the frontend application and then displayed to the user.

The backend provides 6 different endpoints:

- **/api/sentiment**: The most important endpoints. When this endpoint is called, the backend retrieves tweets for a specific hashtag (given as query parameter), then analyzes them and sends them back to the frontend.
- **/api/top**: This frontend automatically gets called after the sentiment analysis has been retrieved. The backend analyzes the recently retrieved tweet-list and searches for the most used hashtags and the users with the most tweets in the data set.
- **/api/user**: This endpoint is used to get information about one user. The backend receives information of the user with the given user-id from the twitter API. Metrics like follower-count, following-count and so on are contained in the retrieved data set.
- **/api/user/followers**: This endpoint returns a list of up to 100 followers of the given user.
- **/api/user/tweets**: This endpoint returns a list of the most recent tweets of the given user.
- **/api/hashtag**: With this endpoint, all the tweets that contain both the main-hashtag and one of the top hashtag can be retrieved from the backend.

Figure 2.1: Flowchart of the application design



2.1 Necessary Functions

The application has some necessary functions for each problem.

- **TweepyClient**: I created a class named `TweepyClient` to have all the necessary functions used for communicating with the Twitter API in one class. Here you can find the functions `GetTweetsByHashtag(hashtag, max_results)`, `GetFollowersByUserId(user_id)`, `GetUserMetricsByUserId(user_id)` and `GetTweetsFromUserId(user_id)` that include the basic Twitter API requests for each endpoint.
- **AnalyseSentimentOfTweetList()**: This function is part of the `SentimentAnalyzer` class, that includes a tweet-list as parameter. The analysis method takes this list

and analyzes every tweet in the list with the help of the `TextBlob`-library. Then the Tweet-objects get assigned their sentiment. The tweet-list afterwards get returned to the caller.

- **AnalyzeTweetList()**: This method is part of the `TopHashtagsAndUsersAnalyzer` class, which includes a tweet-list and the main hashtag as attributes. The function is used to find the most used hashtags and the users with the most tweets in the analyzed data set. It returns a tuple of two lists, one containing the top hashtags and the other containing the top users.

- **StringValidator**: This class is used to validate the strings before making any data transformations. As the API can possibly also be accessed from outside of the frontend, there could be malicious requests, which can be prevented when validating query parameters before accessing data.

- **CsvHelper** and **TweetDataframeHelper**: These two classes were used to unify and ease the work with DataFrames and cached CSV-Data. Whenever the cache has to be accessed, the CsvHelper class was used, so that there is only one place where data is written and read in the application. The TweetDataframeHelper was used to write the Tweet-list into a dataframe or read it frome a given Dataframe and return it to the caller as list.

3 User Interface

The user interface was programmed in React and thus is a single page application. The necessary fragments of the application always get shown/hidden when needed.

Figure 3.1: Starting Screen with parameter selection

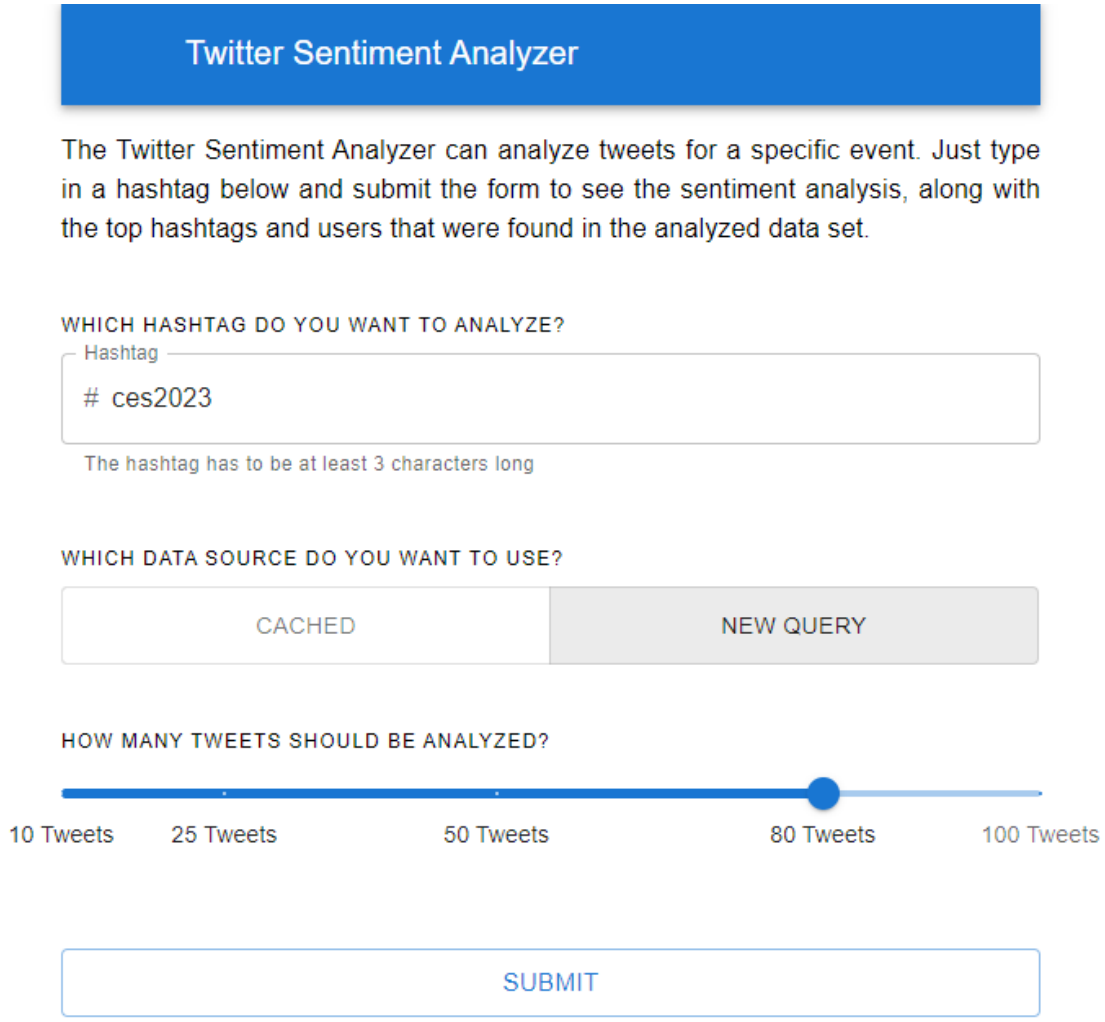


Figure 3.2: Sentiment Screen with sentiment analysis and Top Hashtags / Users

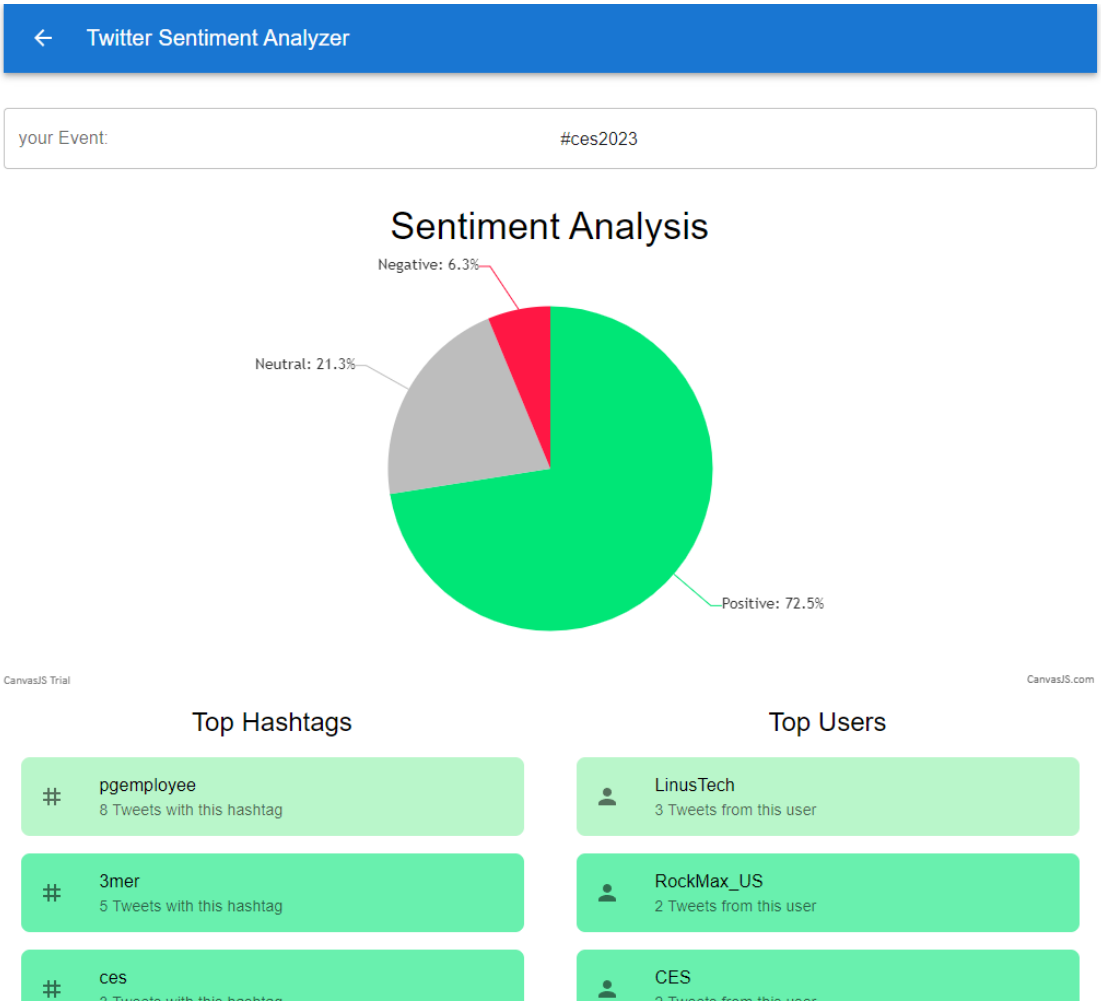


Figure 3.3: User Dialog with basic information

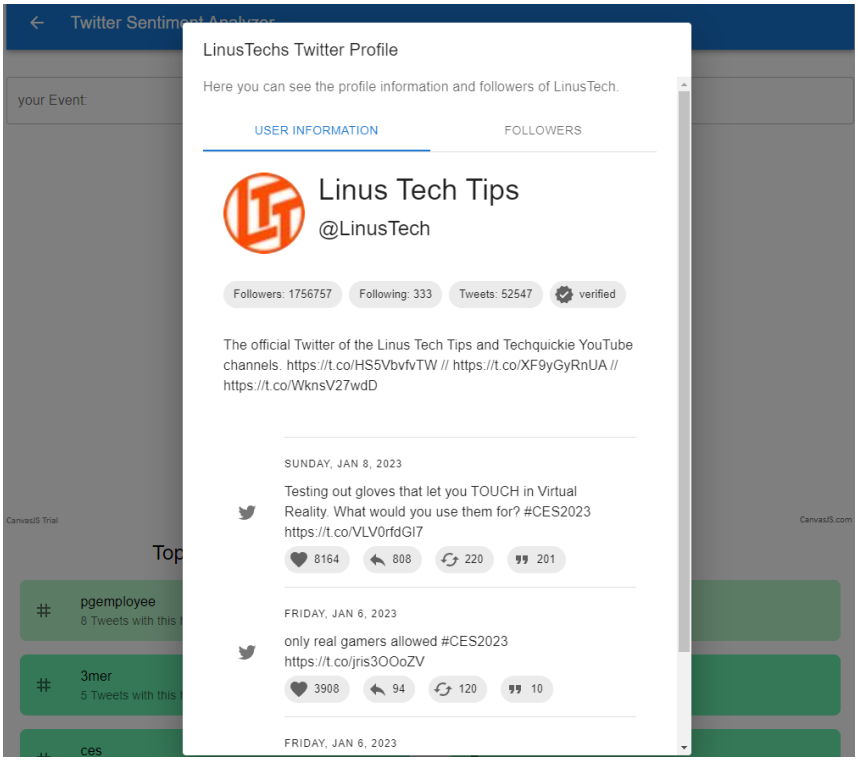


Figure 3.4: User Dialog with followers

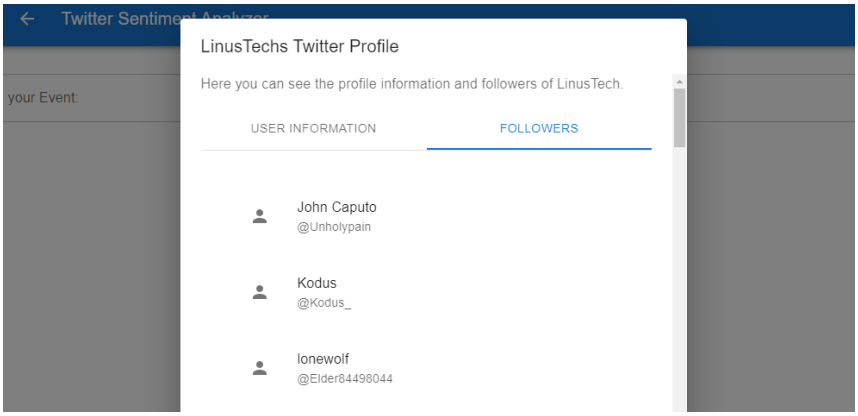


Figure 3.5: Tweet Dialog of followers tweets

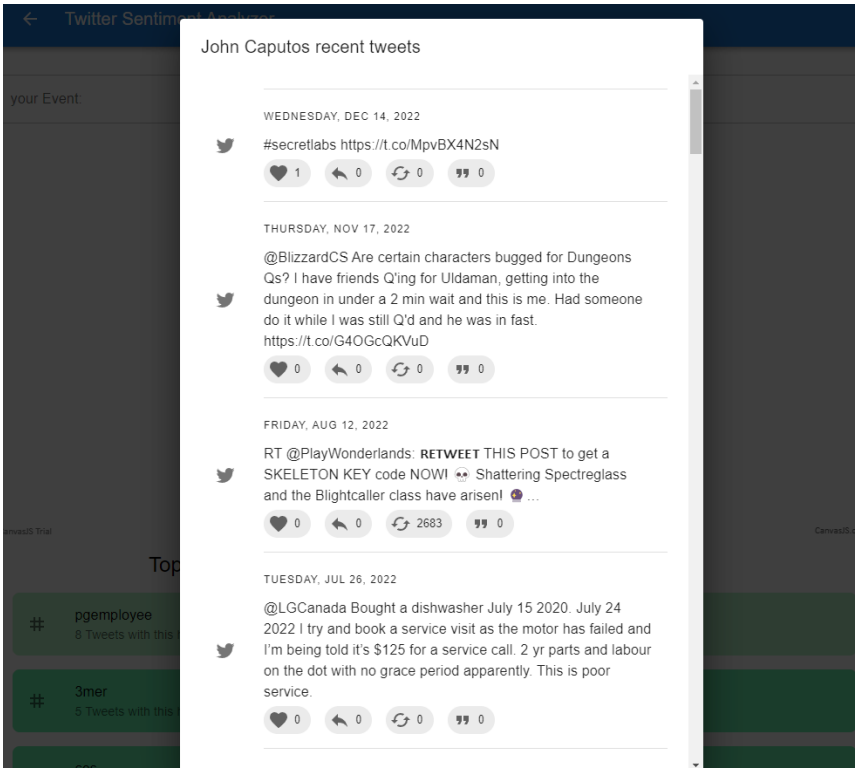
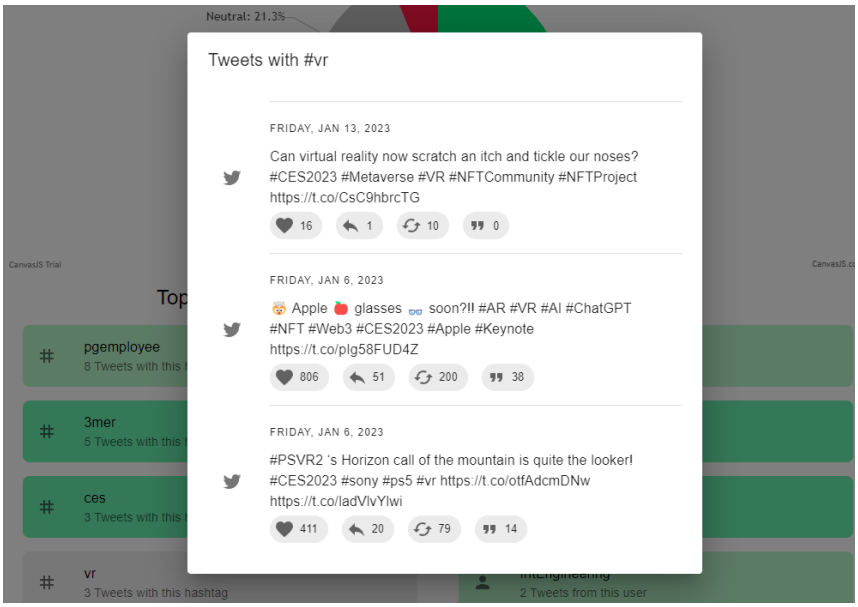
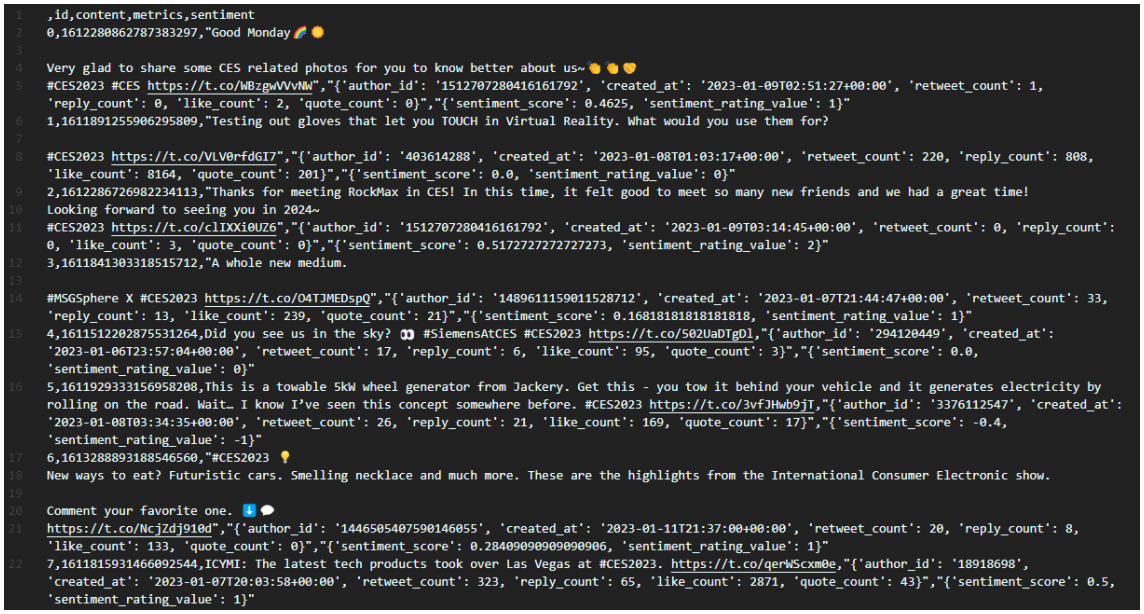


Figure 3.6: Tweet Dialog of tweets that include the selected hashtag



The cached data was saved in a CSV-file, which includes all the analyzed Tweets Data, but no data from the users except the author_id:

Figure 3.7: Example data for #CES2023



```
,id,content,metrics,sentiment
0,1612280862787383297,"Good Monday🌈🌞

Very glad to share some CES related photos for you to know better about us~👏👏🤝
#CES2023 #CES https://t.co/WBzgwVVvNW","{'author_id': '1512707280416161792', 'created_at': '2023-01-09T02:51:27+00:00', 'retweet_count': 1,
'reply_count': 0, 'like_count': 2, 'quote_count': 0}","{'sentiment_score': 0.4625, 'sentiment_rating_value': 1}"
1,1611891255906295809,"Testing out gloves that let you TOUCH in Virtual Reality. What would you use them for?

#CES2023 https://t.co/VLV0rfdGI7","{'author_id': '403614288', 'created_at': '2023-01-08T01:03:17+00:00', 'retweet_count': 220, 'reply_count': 808,
'like_count': 8164, 'quote_count': 201}","{'sentiment_score': 0.0, 'sentiment_rating_value': 0}"
2,1612286726982234113,"Thanks for meeting RockMax in CES! In this time, it felt good to meet so many new friends and we had a great time!
Looking forward to seeing you in 2024~
#CES2023 https://t.co/clIXXi0UZ6","{'author_id': '1512707280416161792', 'created_at': '2023-01-09T03:14:45+00:00', 'retweet_count': 0, 'reply_count':
0, 'like_count': 3, 'quote_count': 0}","{'sentiment_score': 0.5172727272727273, 'sentiment_rating_value': 2}"
3,1611841303318515712,"A whole new medium.

#MSGSphere X #CES2023 https://t.co/O4TJMEDspQ","{'author_id': '1489611159011528712', 'created_at': '2023-01-07T21:44:47+00:00', 'retweet_count': 33,
'reply_count': 13, 'like_count': 239, 'quote_count': 21}","{'sentiment_score': 0.16818181818181818, 'sentiment_rating_value': 1}"
4,1611512202875531264,Did you see us in the sky? 👀 #SiemensAtCES #CES2023 https://t.co/502UaDTgDl,"{'author_id': '294120449', 'created_at':
'2023-01-06T23:57:04+00:00', 'retweet_count': 17, 'reply_count': 6, 'like_count': 95, 'quote_count': 3}","{'sentiment_score': 0.0,
'sentiment_rating_value': 0}"
5,1611929333156958208,This is a towable 5kW wheel generator from Jackery. Get this - you tow it behind your vehicle and it generates electricity by
rolling on the road. Wait… I know I've seen this concept somewhere before. #CES2023 https://t.co/3vfJHwb9jT,"{'author_id': '3376112547', 'created_at':
'2023-01-08T03:34:35+00:00', 'retweet_count': 26, 'reply_count': 21, 'like_count': 169, 'quote_count': 17}","{'sentiment_score': -0.4,
'sentiment_rating_value': -1}"
6,1613288893188546560,"#CES2023 💡
New ways to eat? Futuristic cars. Smelling necklace and much more. These are the highlights from the International Consumer Electronic show.

Comment your favorite one. ⬇️💬
https://t.co/NcjZdj910d","{'author_id': '1446505407590146055', 'created_at': '2023-01-11T21:37:00+00:00', 'retweet_count': 20, 'reply_count': 8,
'like_count': 133, 'quote_count': 0}","{'sentiment_score': 0.28409090909090906, 'sentiment_rating_value': 1}"
7,1611815931466092544,ICYMI: The latest tech products took over Las Vegas at #CES2023. https://t.co/qerWScxm0e,"{'author_id': '18918698',
'created_at': '2023-01-07T20:03:58+00:00', 'retweet_count': 323, 'reply_count': 65, 'like_count': 2871, 'quote_count': 43}","{'sentiment_score': 0.5,
'sentiment_rating_value': 1}"
```

4 Conclusion

After working on this project for the past couple weeks, I can say that data science is not as easy as I have thought. Often the data doesn't behave like I would like it to. I often ran into problems where my backend provided the data not like my frontend requested it and so there where a few exceptions that I had to overcome. All in all I'm proud of the application I developed and it was fun working on my own project. I learned a few things about Python, Pandas, API-requests and Frontend development. Sadly I wasn't able to use machine learning in this project, but its a topic I want to work on in the future and am in anticipation of learning.