Instagram #Coronavirus

Posts, Sentiment, Likes and #FakeNews Relation over time

Tobias Drebert
ITMO University

05.06.2020

1. Data gathering

For the collection of Instagram[1] Data I was looking for a suitable api interface. I searched for all posts of one given hashtag. I had already made the experience that official api[2] are often limited and only disclose the necessary data against payment. For my task I needed all posts to be able to make a valid statement. My first idea was to get data from the website[3]. But I don't find a good solution for pagination with the tokens. Then I found on Stackoverflow a unofficial api interface to get posts in hashtag with pagination[4].

First I searched on Instagram to find the most popular hashtag for corona with the most posts. It was #coronavirus with 25 million Posts.
https://www.instagram.com/explore/tags/coronavirus/?__a=1 gives me the first 60 posts (graphql.hashtag.edge_hashtag_to_media.edges) and a next token (graphql.hashtag.edge_hashtag_to_media.page_info.end_cursor). With the token it is possible to get more posts of the hashtag:
https://www.instagram.com/explore/tags/coronavirus/?__a=1&max_id=[token]. This call is repeated with the new token until all posts are loaded.

However, this call did not return a few old posts to the latest posts. Thus, the analysis could also look at a longer period of time. Furthermore, 25 million posts are very many posts that the calls would take days. So I decided to download about 1 million data (950,054 posts; 4.25 GB), which took about 1 day by the many calls. These were the posts of 7 days (26.05-02.06) and some posts (old posts) from 01.01.2020-02.06.2020. My script is written in python (api/main.py) and because there were crashes in the connection, I had to write another script, where I could download from the next token (api/main_next.py). My scripts save every 10,000 posts the posts and tokens into files (data/instagram).

In the json only the posts are saved and top posts, related hashtags, hashtag picture are ignored. The json of each posts contains text, images, date, id, shortcode, likes … (see structure in appendix).

[1] https://www.instagram.com/
[2] https://developers.facebook.com/docs/instagram-api/guides/hashtag-search
[3] https://www.instagram.com/explore/tags/coronavirus/
[4] https://stackoverflow.com/a/51724707/10457835,
https://stackoverflow.com/questions/49265339/instagram-a-1-url-not-working-anymore-problems-with-graphql-query-to-get-da

2. Selection and Implementation

For an analysis over time all data is needed but I splited the data in posts from 26.05-02.06 (all posts) and 01.01-02.06 (not all posts, only selection). Instagram posts have many emojis so I used for sentiment analysis a emoji dataset (data/EmojiSentiment) with sentiment score[5]. With Spark I go through all posts and calculate the average of the sentimental score, which can be calculated from the Emoji dataset (positive-negative)/total. The score shows in the interval +1 (positive), 0 (neutral), -1 (negative) the sentiment, measured by the used emojis. Furthermore, I was interested in hashtag relations especially #fakenews. In Instagram posts the hashtags are in the text. So I checked if #fakenews occurs in the text (lowercase). Finally, I calculated the average number of likes of the posts. My implementation is written in Scala (Main.scala).

3. Processing and Analysis

The result was saved as csv files[6] and then graphically processed with Excel[7].

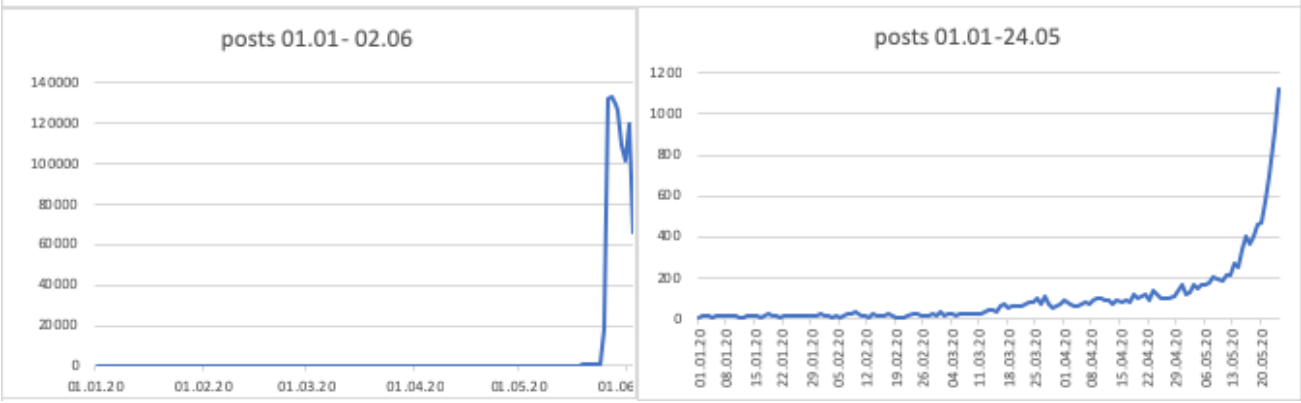


Figure 1: Posts 01.01-02.06

As stated in section 1, only the most recent data is included in the crawled data set. However, there is some data before the 24.05. which allows a small analysis.

[5] http://kt.ijs.si/data/Emoji_sentiment_ranking/
[6] result/20200605_instagram_950054_sentiment.csv; result/20200605_instagram_sentiment_20200602-20200525.csv
[7] result/20200604_Instagram_Visualisation_v2.xlsx

date	posts	Sentiment Sum	Senti ment Score	fake news Sum	Faken ews Score	Like Score
02.06.20	65924	17577.3185	0.2666	206	0.0031	63
01.06.20	119463	32022.0565	0.2680	340	0.0028	96
31.05.20	101987	27414.8982	0.2688	285	0.0028	106
30.05.20	109541	29226.7544	0.2668	359	0.0033	101
29.05.20	127016	33809.3298	0.2662	337	0.0027	99
28.05.20	128980	34078.2995	0.2642	366	0.0028	122
27.05.20	132925	35100.9327	0.2641	363	0.0027	104
26.05.20	132057	34977.6965	0.2649	416	0.0032	108
25.05.20	18587	4977.1075	0.2678	145	0.0078	109
24.05.20	1122	308.2145	0.2747	3	0.0027	77
23.05.20	921	241.2610	0.2620	1	0.0011	60

The number of Likes is about 100 Likes on average for #Coronavirus posts with a slight increase. Looking at the last week, it's noticeable that the number of Likes decreased slightly to 02.06. (Tuesday).

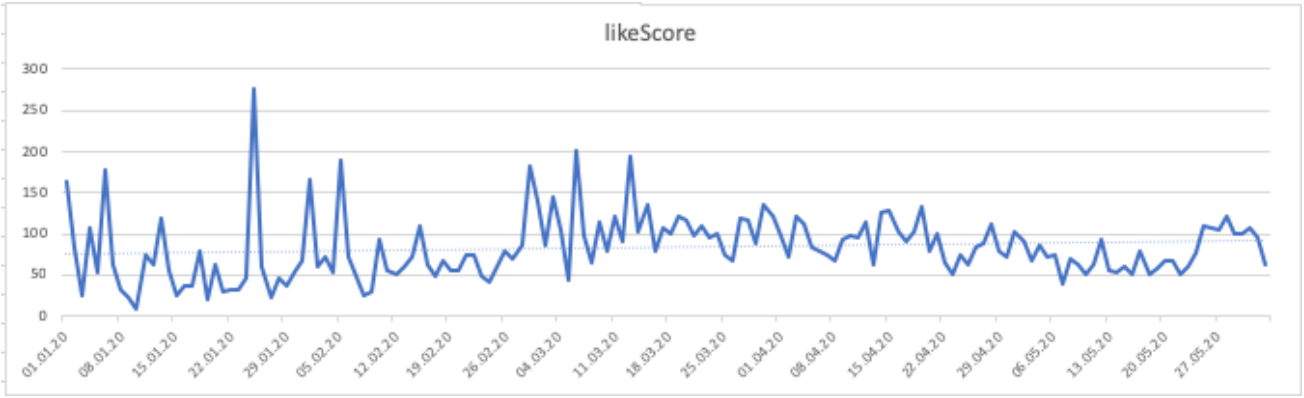


Figure 2: Like Score 01.01-02.06

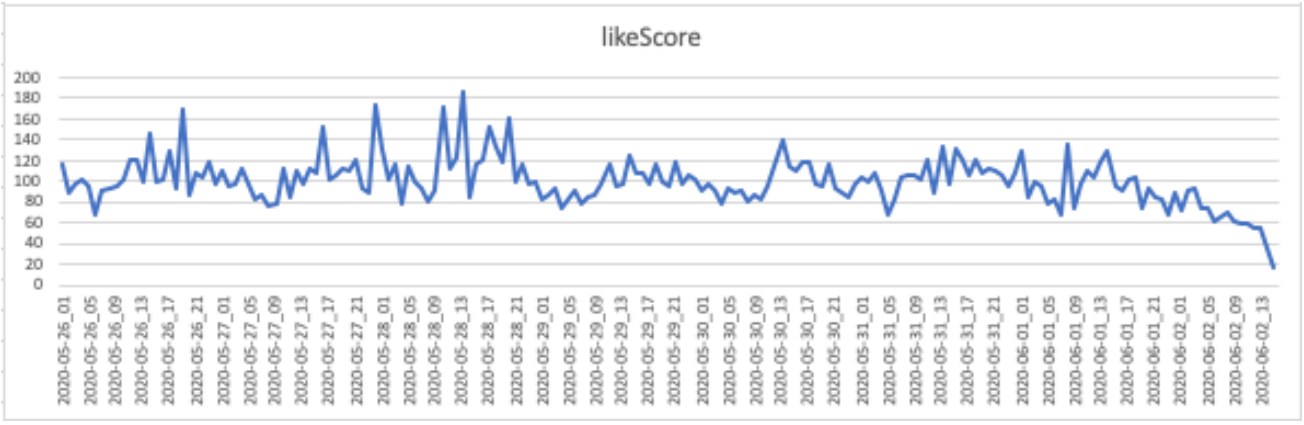


Figure 3: Like Score 26.05-02.06

The posts about Corona are positive on average (Sentiment Score 0.26), which is slightly more positive in the data set than the emoji 😂. The unsettled curve can be attributed to the low number of posts, because in the week with all posts, there are almost no deviations from the score.

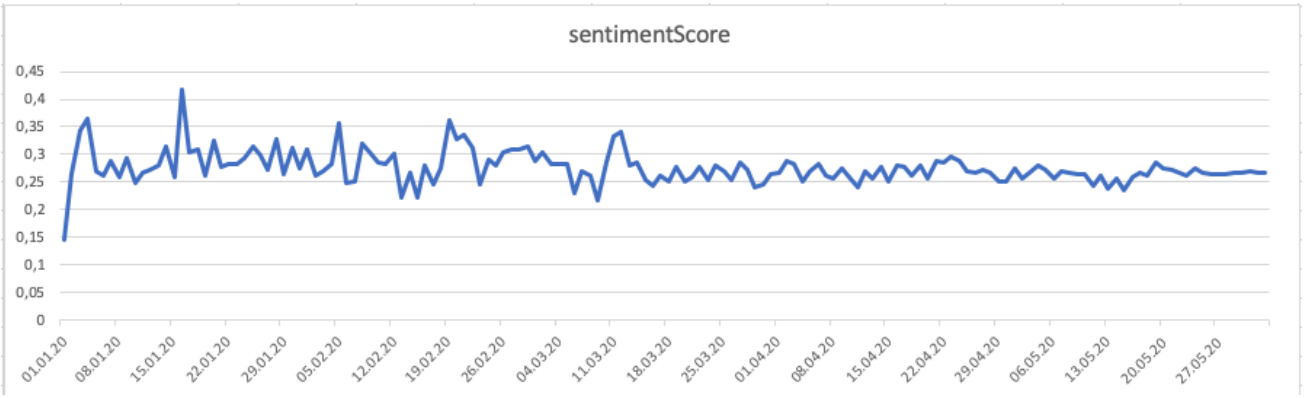


Figure 4: Sentiment Score 01.01-02.06

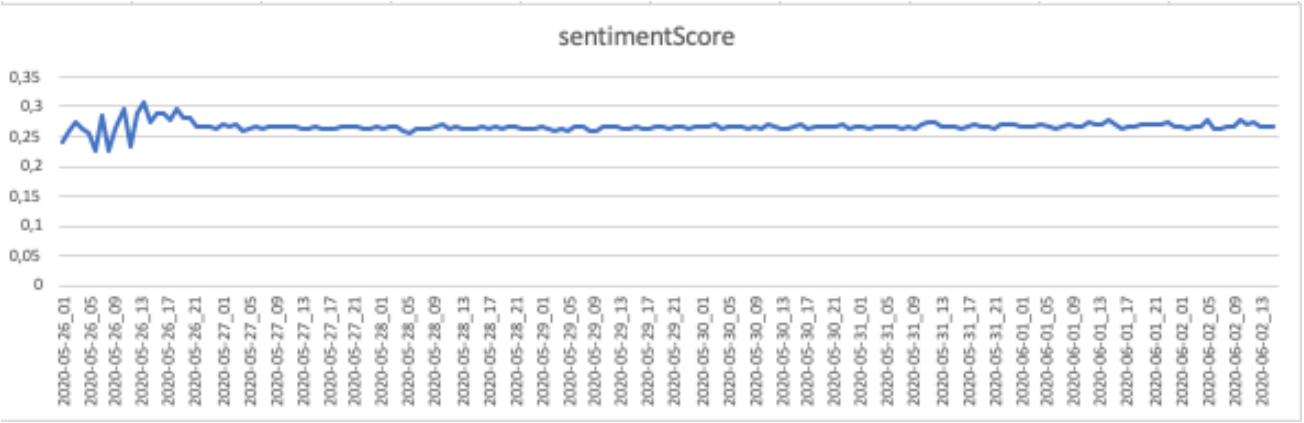
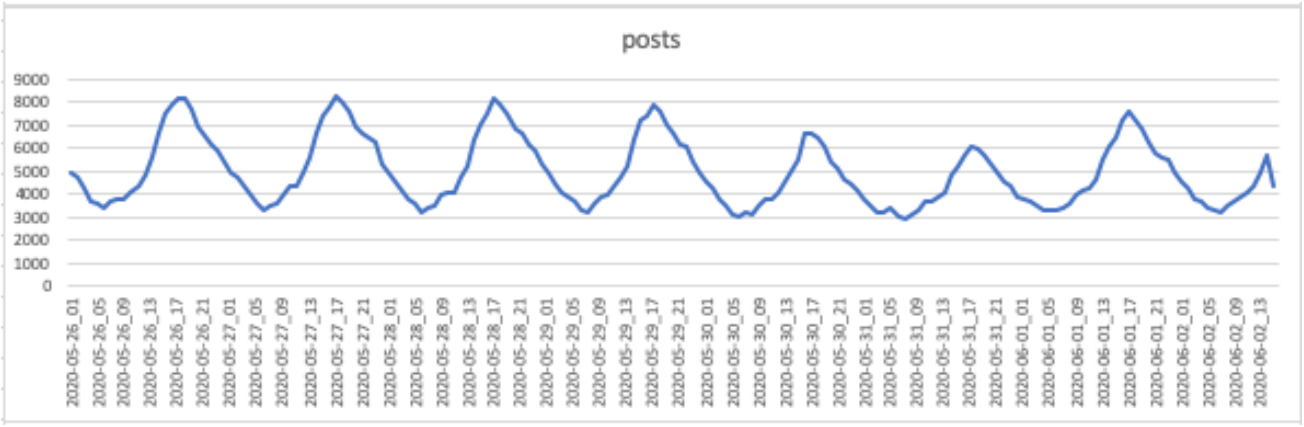


Figure 5: Sentiment Score 26.05-02.06

It is remarkable that the number of posts depends on the time of day. Thus, it can be stated that most of the posts to Corona are created every day around 6 p.m. and that worldwide. Furthermore, the fewest posts to Corona are created between 6 and 9 a.m. Furthermore, on weekends about 20 % (Saturday, 30.05) - 25 % (Sunday, 31.05) less posts are created for Corona than during the week. Whether this phenomenon only occurs with #Coronavirus cannot be said, because other hashtags were not considered.



There is a connection from #Coronavirus to #fakenews. This is between 150-400 posts per day. The percentage of 0.04% is relatively small, because fake news is more of a political issue and is less used by influencers and private persons who post a lot.

4. Appendix

Stucture of post json

```
root
 |-- __typename: string (nullable = true)
 |-- accessibility_caption: string (nullable = true)
 |-- comments_disabled: boolean (nullable = true)
 |-- dimensions: struct (nullable = true)
 |    |-- height: long (nullable = true)
 |    |-- width: long (nullable = true)
 |-- display_url: string (nullable = true)
 |-- edge_liked_by: struct (nullable = true)
 |    |-- count: long (nullable = true)
 |-- edge_media_preview_like: struct (nullable = true)
 |    |-- count: long (nullable = true)
 |-- edge_media_to_caption: struct (nullable = true)
 |    |-- edges: array (nullable = true)
 |    |    |-- element: struct (containsNull = true)
 |    |    |    |-- node: struct (nullable = true)
 |    |    |    |    |-- text: string (nullable = true)
 |-- edge_media_to_comment: struct (nullable = true)
 |    |-- count: long (nullable = true)
 |-- id: string (nullable = true)
 |-- is_video: boolean (nullable = true)
 |-- owner: struct (nullable = true)
 |    |-- id: string (nullable = true)
 |-- product_type: string (nullable = true)
 |-- shortcode: string (nullable = true)
 |-- taken_at_timestamp: long (nullable = true)
 |-- text: string (nullable = true)
 |-- thumbnail_resources: array (nullable = true)
 |    |-- element: struct (containsNull = true)
 |    |    |-- config_height: long (nullable = true)
 |    |    |-- config_width: long (nullable = true)
 |    |    |-- src: string (nullable = true)
 |-- thumbnail_src: string (nullable = true)
 |-- video_view_count: long (nullable = true)
```