Game of Thrones: character mentions and audience sentiments

Gabriel Dintrans, Sebastián Ferrada, Paula Ríos

19 de junio de 2017

1. Objective

The objective of this work is to track the number of tweet mentions of certain characters of Game of Thrones TV Series, during the broadcast of an episode premiere.

Additionally, to link the appearance and/or mention of the characters with certain emotions analysing the emojis that are present in the tweets.

2. Data

We used a data stream from Twitter. This was acquired during the broadcast of the premiere of the sixth season of Game of Thrones in 2016. The acquisition was made using the Twitter API, which allows to track keywords, hashtags or account mentions. We selected the following keywords to track tweets: `got`, `game of thrones`, `#got`, `#gameofthrones`, `#gotfinale`, `@gameofthrones`. Tha data consists in 10 GB of tweets in `json` format, containing the text and other metadata such as the author, the timestamp, amount of retweets, etc. This data was previously parsed: we extracted only the text and timestamp of the tweets and output a `csv` file for each episode of the series.

3. Methods

Data must be preprocessed in order to run map-reduce operations with Hadoop. Tweets are originally extracted as `json` files from the Twitter API. We run a script in Python to load the `json` objects and extract only the relevant attributes for the project, which are the text of the tweet and its timestamp. We produced 9 `csv` files, one for each episode.

In Parallel we scrapped the list of characters from the series Season 6 Wikipedia article, containing 124 credited characters. We manually added nicknames for the characters that are best known for them; for example, Lord Petyr Baelish

is also known as “Little finger” and will be most likely mentioned like that in the tweets.

Since we want to know the number of mentions that a character has and how it varies during the broadcasting of an episode, we decided to obtain the number of mentions of each character each minute of the transmisions. Todo so, we run a first map task that reads the text of the tweet, maps the timestamp to a minute and matches the characters names and nicknames against the text. If a match is found it outputs a tuple `(character_name+minute, 1)`. Afterwards we perform a reduce function that sums the output of the mapping, delivering the number of mentions of each character on each minute of the episode. Finally, we load the count on R to draw the charts, showing the trends of the characters mentions through the episode.

We also attempted to perform an exploratory sentiment analysis, using the emojis that were present on the tweets. To do so, we use the `emoji-java`[1] library, which matches the texts and extraxts all the emojis that are present. Then, we matched the text as before, obtaining a full list of emojis used when referring to a particular character.

4. Results

Applying our methods we found that, in general, there is a correlation between character mentions on tweets and their appearance on the show. However, there were certain character names that were mismatched, for example, Samwell Tarly, who is also known as ”Sam”matched with words like ”same”. Something similar happened with Eddard Stark, also known as ”Ned”.

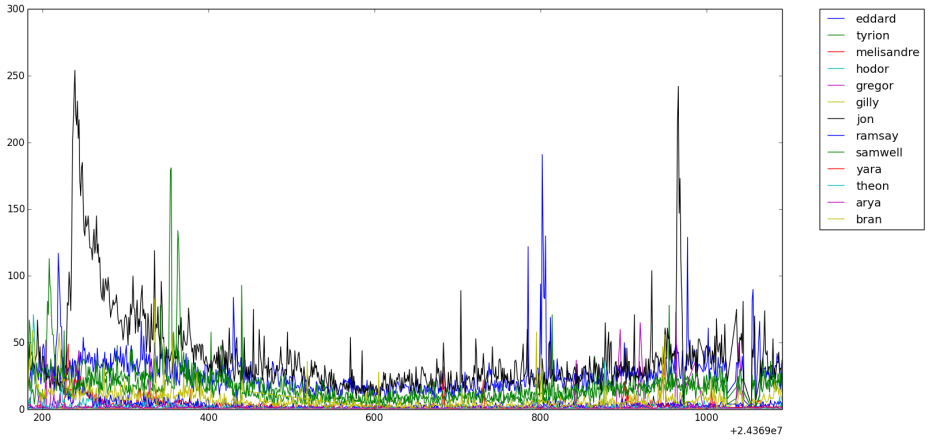

Figura 1: Graph with results of character mentions during episode 2.

[1] `https://github.com/vdurmont/emoji-java`

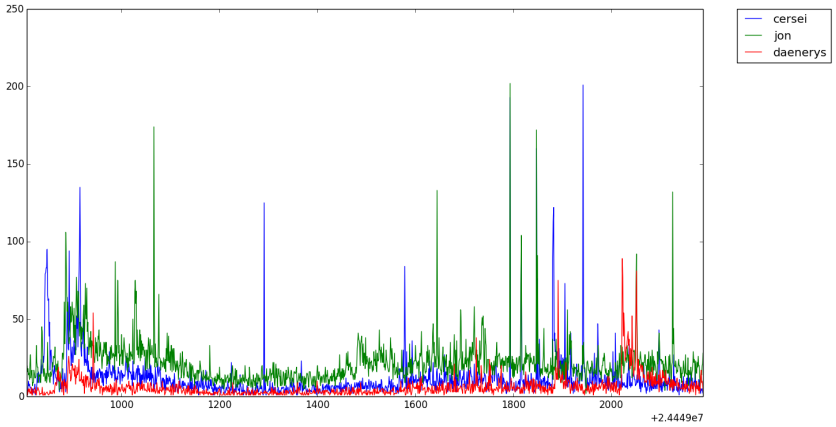

Figura 2: Graph with results of character mentions during episode 10.

5. Conclusion

The main lesson we learnt is how to deal with parallel algorithms on distributed file systems. Initially we didn't think having a static global variable containing the characters names (in this case, a hashmap) would be a problem, until we realised other machines didn't have access to it, and therefore the MapReduce we had created was useless. The solution to the problem was to add a cache file with character names information to the MapReduce job, which would then be accessed during map set up and create the hashmap we originally had.

In regards to character mention analysis, there were issues with the matching of certain character names, as previously mentioned, which could be fixed by performing a better matching of words to character names, avoiding common language words. In regards to the sentiment analysis using emojis, we were unable to get results due to difficulties with the library used to extract emojis from tweets.