# Visual Twitter

## Problem Presentation

The aim is to collect data from Twitter, according to several filters such as: language, domain, location and so on. The collected information will be processed and then clustered together, using certain means to figure out the similarity of tweets. Different clusters may be created for different subjects.

At the end of the clusterization process, the tweets will be processed again in order to display, for the user, what information could be extracted regarding a certain topic from the cluster it belongs to.

## State-of-the-art

When we previously talked about the State-of-the-art, we did so by splitting our project's requirements into 3 tasks: gathering data, the clustering process and evaluating the results. The first one, gathering data is done, in our case, by making use of the API offered by Twitter. While others may choose to implement or use their own or a third-party software for this, we decided that since Twitter's own API offers a large number of endpoints and different ways to manipulate the information received from them, there was no need for us to sidetrack into a different problem.

The clustering process usually implies grouping objects together, based on how similar they are. When performed over text, the most commonly encountered method is to use TF-IDF, which stands for term frequency-inverse document frequency. This looks at how relevant a word is to the document it is part of and then how relevant it is to the whole collection which the document belongs to.

TweetMotif[1] presents another solution. They try and characterize a topic by a 1-to-3 word textual label and a set of messages, whose texts must all contain the label. They do this while keeping track of a number of things: frequency contrast, topic diversity, topic size and the number of topics. Having said this, for them the clustering process can be seen as the scoring and filtering of topic phrase candidates, which is done in a similar manner to TF/IDF.

As for the data evaluation part, after conducting research on the aspects mentioned in the initial state-of-the-art file, we came to the conclusion that treating all the problems that might surface when referring to data authenticity and grammar and abbreviations checking would be a

1 TweetMotif: Exploratory Search and Topic Summarization for Twitter

difficult problem on its own, so for the sake of prioritizing the problem we first decided to tackle, we chose to go with a pragmatic approach. This approach consists of considering only a few whitelisted Twitter users when gathering tweets for the initial centroids (for authenticity and relevance purposes) and outlining the events based on predetermined patterns (for relevance purposes).

## Our solution

The proposed solution is a web application which can be used to view information about basketball matches that took place in a certain period of time, primarily those in the NBA. As an overview of how this works, a dataset is created by sampling random tweets related to our subject. When a user searches using the names of the teams that participated in the match, the clustering process over our dataset starts. Once the clustering ends, we look into one of the clusters and take the relevant information from the tweets that are there.

The user interface and the use case remain quite simple, as most of the focus has been put into data gathering and into the clustering algorithm. The clustering algorithm is obvious, as it was the main focus of this project and the requirement that we have given ourselves was to end with an algorithm that is able to group together tweets that have a similar message or are related to the given topic.

Data gathering on the other hand proved to be more difficult due to the approach we chose to take at first. The initial proposed solution would have considered topics and events from a more general range (e.g sports, political events, artistic events, etc.), but since the complexity of such a solution would be too big we decided to follow a niched approach that only focuses on sports, more precisely basketball. By doing this, we would increase our odds of having relevant information added to our dataset and it would also allow us to better see how well our algorithm works, since most tweets would already be part of the same domain.

## Results

### Gathering the data

For our first attempts, we tried to gather tweets regardless of whether they were related to our subject or not. Compared to later attempts, this did prove to be faster, adding much more tweets to the dataset than the other methods, in the same time interval. On the other hand, when taking into consideration the number of posts that are made everyday, the fact that our dataset would be bigger by a few thousand tweets would not matter that much, when put into perspective. Also because of this, the chances of tweets related to basketball being added to the dataset were also quite slim. In our earlier attempts, we had cases where out of a 10 000 tweets sample, none contained words or tags that would tie them to basketball. Hence, the idea that we could build a pertinent solution in this way was too optimistic.

What we settled on was to use the Filtered Stream endpoint. This allowed us to only get tweets for our dataset that are directly related to the subject that we chose to focus on. This works similarly to what we did on our first attempts, pulling random tweets in order to create a sample, but in addition we are able to add a set of rules which dictate how filtering is done on the API's side, deciding which tweets are returned. In our case, the rules are quite simple, requiring the tweet to contain one of several relevant hashtags such as #basketball or #nba. In addition to this, we also require the tweets to be in English, to help us in the processing that takes place before the actual clustering.

Additionally, along with this sample of tweets that we create, we also have a list of whitelisted users, from which we pull the last 30 tweets from their timeline. While smaller than the entire corpus of tweets, tweets related to basketball only are still not small in number. To these we also add the tweets that qualify as spam and were able to get past the filters we set on our side and we still do not have the certainty that our dataset would contain tweets related to match results. To this extent, these users were handpicked after making sure that they mostly post about basketball related events, increasing the chances to collect the data we need.

### The clustering process

We split this process into two parts: loading the data and the actual clustering. Loading the data presumes pulling our tweets from the database into memory and pre-processing them for clustering. This consists of tokenizing tweets and calculating the TF/IDF value for each token. These values are the basis on which the similarity between tweets will be later calculated.

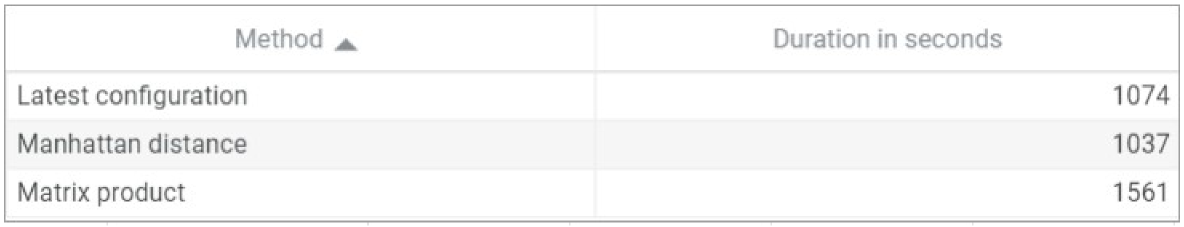
In regard to TF/IDF, our approach presents a difference compared to the traditional one. Normally, the same word, but part of different documents would have different frequencies. In our case however, once the frequency for a word has been calculated once, it will be saved and then, the next time the same word will be encountered, it will be assigned to it also. This is mainly because our documents are tweets, which do not average a high count of words and consequently the term frequency for a word would usually be 1. Of course, the possibility of the same word having a higher frequency for another tweet does exist and our approach ignores the existence of such variations, while only taking in account the first encounter, but due to the improvement in execution duration that we get and due to the fact that the frequencies will keep changing during the clusterization process, we deemed this trade off to be beneficial.

The first step in the clustering process is creating the centroids for our clusters. Usually this is done by taking random tweets from our dataset and then, through the process, the tweets representing the centroid would constantly change. We do the same with the exception that one centroid is specifically created in order to build a cluster that contains the tweets related to our topic. Our approach here, in order to make sure that we would have at the end a cluster that is related to our subject, was to have one of the centroids specifically built around our topic. We did this by taking the tweets belonging to the whitelisted users, we search for those that contain the teams' names searched by the user, using the first **n** results as the centroid.

Once we have our initial centroids, the clusterization consists of applying the same operations over our tweets again and again, until the ending condition has been met, in our case, the maximum number of iterations has been reached. The main points here are calculating the similarity between tweets and the centroids, in order to find their membership and also updating those values as the process advances.

For the first part, we use cosine similarity. The values we calculate in pre-processing are used to create the term-frequency vectors and by calculating the cosine of the angle between two vectors we can determine if they point in the same direction, this translating to the degree of similarity between them. This cosine similarity is calculated between each tweet and the centroids and these values are compared against one another in order to find the apartenency of the tweet to a cluster. The second part, updating the values, is done so that the process does not become stale from one iteration to the other, meaning that we will have changes in our clusters' structure.

As far as calculating similarity is concerned, we have tried different approaches and the solution we arrived at is the result of a longer process of trial and error. Of these, however, we will go over a different method of calculating two vectors' product and also the use of Manhattan distance instead of cosine similarity and compare their results.



| Method ▲ | Duration in seconds |
|---|---|
| Latest configuration | 1074 |
| Manhattan distance | 1037 |
| Matrix product | 1561 |

*Figure 1: Average durations for different clustering configurations. Tests run on 5000 iterations*

Let us first talk about the results when transforming the vectors' product into a matrix one. Along with the increase in duration that is observable, this method presented one additional drawback. If we are to say that the Manhattan distance method for calculating similarity produces worse results, meaning that the end cluster size would be greater, most of the tweets added not having much correlation with those inside the centroid, that is, its accuracy is too low, then for the matrix product method, the accuracy would be too high. For most cases, the end result would consist of only the centroids. That being said, both methods perform worse in this regard, when compared to the method we settled on.

In regards to duration, an argument may be made in the favor of the Manhattan distance, but we decided that giving up on accuracy just for a slight decrease in runtime would not be beneficial for us.

## Evaluating the results

If we are to talk about how good our solution is in regards to the result of the clusterization algorithm, that is what our designated cluster contains when we've reached the given number of iterations, then we can consider that we have reached our goals.

After running multiple tests in different configurations, we can say that for the most part, the content of the cluster is related to the topic that was given as input, meaning that most of the tweets inside it talk about either one of the teams or the match itself. We do say most of them because some unrelated tweets do still manage to sneak through. For the most part however, these are tweets related to other teams, but which do have a similar structure to those found in our centroid.

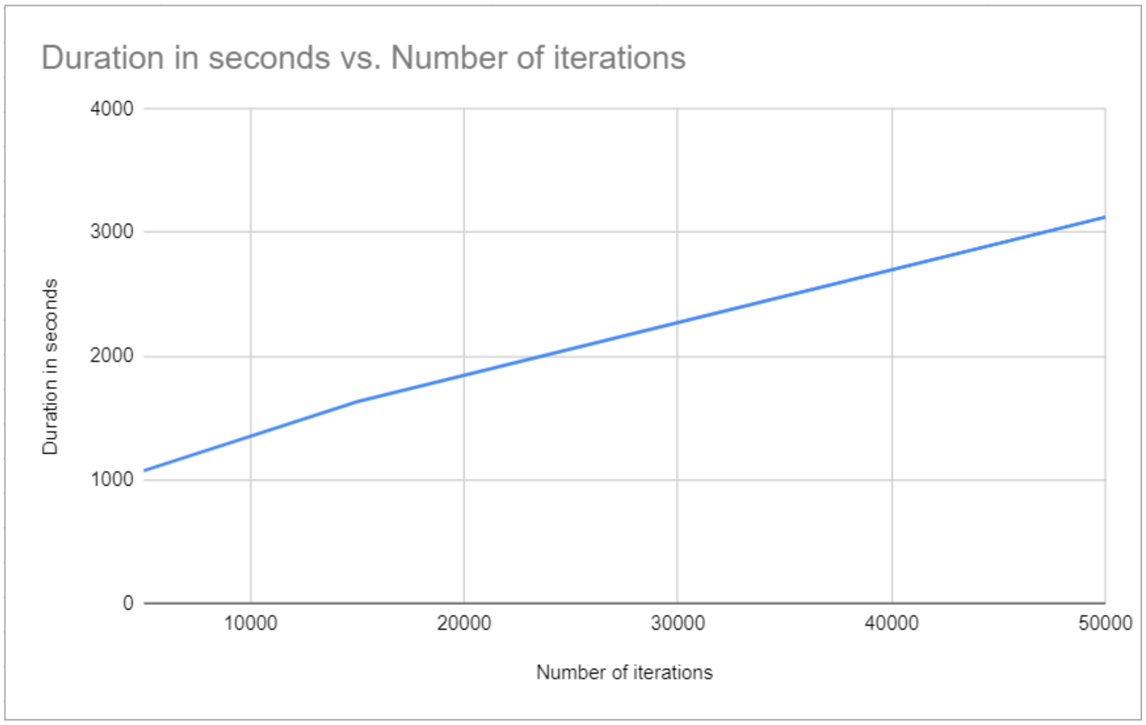


*Figure 2: Correspondance bettwen number of iterations and duration*

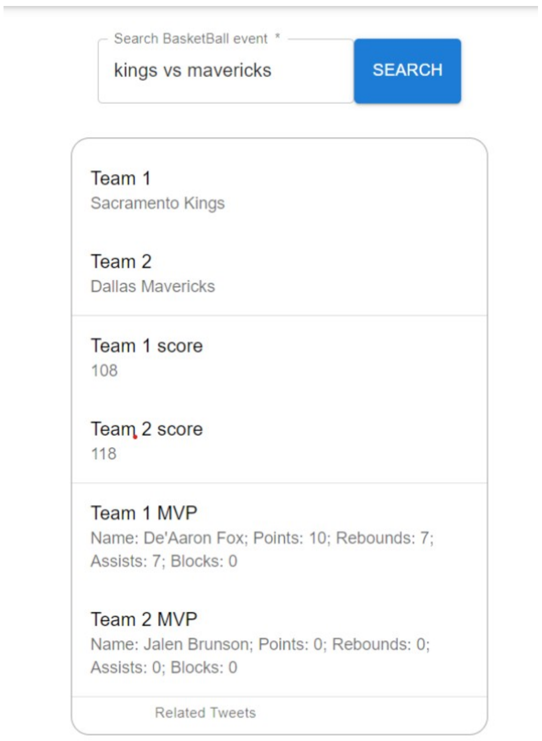
When talking about the end result, what is displayed for the user to see, however, there is still some work left. Our intent here was to create a few templates for different events such as matches, transfers, accidents and so on, but due to how difficult it proved to make only one such template and other difficulties imposed by the clustering process, we opted to stick with only the first of the previously enumerated templates. While for the most part it does what it is supposed to do, it remains highly dependent on the content of the cluster and the order of the tweets inside it. More improvements could be made here, given time, but we consider that interpreting the results of such a process would prove to be a difficult problem to tackle in itself.

## Use case

On the 31st of December we collected tweets in order to build a smaller dataset, to test the last version of the algorithm. We ended up with a dataset of 5000 tweets. After analyzing some of the tweets, more specifically, those of the whitelisted users, we noticed tweets about the match between Sacramento Kings and Dallas Mavericks. Consequently, we chose to run our tests on this match.

After multiple runs, we noticed that, regardless of the number of iterations, the number of tweets that are part of the cluster that we are interested in average around 30, with few exceptions, in the case of the higher number of iterations, averaging around 40 tweets. These clusters were then post-processed, meaning that we would parse through the tweets that were in these clusters in order to gather the information that would later be displayed for the user to see.

For this part, we try to look for certain structures inside the tweets that are most often used in order to describe the outcome of a match. We chose this approach because in order to gather the data needed, no matter the structure of the tweet, would be another problem in the direction of data mining. Glossing over this issue, our format still provides satisfactory results. This being said, the accuracy of the results depends on the order that the tweets were added to the cluster.


Search BasketBall event *
kings vs mavericks
SEARCH

Team 1
Sacramento Kings

Team 2
Dallas Mavericks

Team 1 score
108

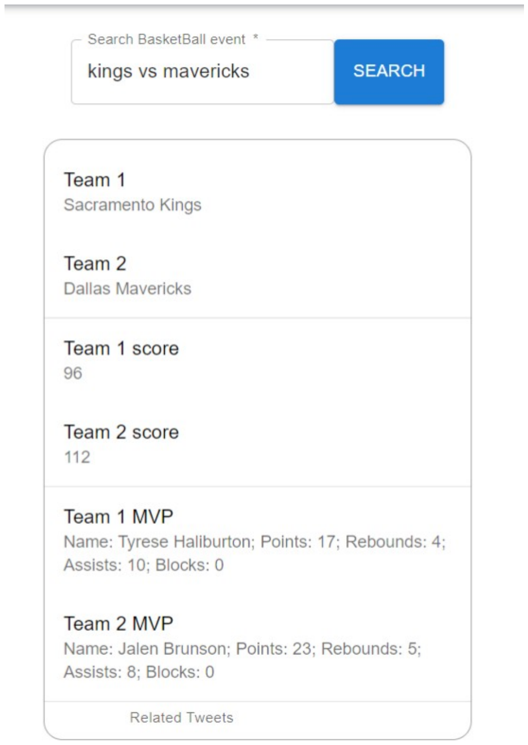
Team 2 score
118

Team 1 MVP
Name: De'Aaron Fox; Points: 10; Rebounds: 7; Assists: 7; Blocks: 0

Team 2 MVP
Name: Jalen Brunson; Points: 0; Rebounds: 0; Assists: 0; Blocks: 0

Related Tweets

*Figure 3: 31st December test results 1*


Search BasketBall event *
kings vs mavericks
SEARCH

Team 1
Sacramento Kings

Team 2
Dallas Mavericks

Team 1 score
96

Team 2 score
112

Team 1 MVP
Name: Tyrese Haliburton; Points: 17; Rebounds: 4; Assists: 10; Blocks: 0

Team 2 MVP
Name: Jalen Brunson; Points: 23; Rebounds: 5; Assists: 8; Blocks: 0

Related Tweets

*Figure 4: 31st December test results 2*

Those two are results of tests run on the same dataset and here we can see why the order of the tweets inside the cluster matters. For Figure 3, we can identify 2 problems: the teams' scores and the fact that for Mavericks, although Jalen Brunson was identified as the MVP, his points are not identified and they remain 0. Although the MPV for Kings differs from the one in Figure 4, after looking through other tweets related to this match, he was the player with the second best performance for their team.

In Figure 4, the correct scores for the match were identified and they were assigned accordingly. Moreover, for both teams, the players' with the best performance were properly identified and their contributions also.

## Comparison with other solutions

While we did not find any similar existing applications that extract information about specific events (such as a basketball match between two given teams) from Twitter, there are solutions that offer somewhat similar functionalities, such as TweetMotif, which takes a word or phrase as input and finds tweets in which that phrase is used, as well as trending topics and related themes relevant for that specific phrase. A big difference between TweetMotif and our solution is that the first provides a more general overview of data related to the searched input, while our solution focuses on extracting information about a specific topic from tweets related to it.

There are also scientific articles that emphasize different processes of clustering and filtering tweets, along with their  pros and cons, but we did not manage to find any actual web application related to them.

## Future work

As for future functionalities that would make our tool more useful, a few good mentions would be:

- History functionality, so that the user can review and access old search queries;
- Whitelisted users from different areas corresponding to different topics (other sports, such as football or tennis, or different topics such as political or artistic) for wider data range for the initial dataset;
- More data patterns for extraction of relevant information regarding the possible additional topics mentioned above;
- Statistical overview on the tweets related to a search;
- A friendly way of viewing those tweets individually;
- A more aesthetic interface for the application;
- Authenticity check for all the tweets that make up the results clusters;

## Conclusions

After presenting the problem tackled, as well as the solution we proposed and comparing it with similar existing solutions, while considering the hurdles encountered and the complexity of the process, we can conclude that the final product we developed serves as a good pragmatic solution for the most important aspects outlined in the initial state-of-the-art.

By the end of the process we managed to come across and use advanced techniques such as clustering and natural language processing, which were also the most difficult steps in the development of the application.

## Bibliography

1. Meysam Asgari-Chenaghlu, Mohammad-Reza Feizi-Derakhshi, Leili Farzinvash, Mohammad-Ali Balafar, Cina Motamed. Topic Detection and Tracking Techniques on Twitter: A Systematic Review

2. Kevin Dela Rosa, Rushin Shah, Bo Lin, Anatole Gershman and Robert Frederking. Topical Clustering of Tweets

3. Rajaraman, A.; Ullman, J.D. (2011). "Data Mining" (PDF). *Mining of Massive Datasets*. pp. 1–17. Ref3

4. Kevin Dela Rosa, Rushin Shah, Bo Lin, Anatole Gershman and Robert Frederking. Topical Clustering of Tweets

5. Gromann, Dagmar & Declerck, Thierry. (2017). Hashtag Processing for Enhanced Clustering of Tweets.

6. Cody Buntain,Jennifer Golbeck. (2018). Automatically Identifying Fake News in Popular Twitter Threads

## Links

1. TweetMotif research paper sample: FIXME Formatting Instructions for Authors
2. Twitter API documentation: Documentation Home | Docs | Twitter Developer Platform
3. Topic Detection and Tracking Techniques on Twitter: A Systematic Review: Topic Detection and Tracking Techniques on Twitter: A Systematic Review
4. Hashtag Processing for Enhanced Clustering of Tweets: (PDF) Hashtag Processing for Enhanced Clustering of Tweets
5. Analysis of Twitter Data Using a Multiple-Level Clustering Strategy: Analysis of Twitter Data Using a Multiple-level Clustering Strategy
6. Topical Clustering of Tweets: Proceedings Template - WORD
7. Clustering of twitter technology tweets and the impact of stopwords on clusters: Clustering of twitter technology tweets and the impact of stopwords on clusters
8. Automatically Identifying Fake News in Popular Twitter Threads: https://arxiv.org/pdf/1705.01613.pdf