TP: Mining of Twitter data

Oana Balalau (TA), Mauro Sozio
name.lastname@telecom-paristech.fr

In this lab session we are going to analyze data collected from Twitter with the objective of finding interesting information and patterns. The lab consists of three main tasks:

1. data cleaning and preprocessing. You will be given a collection of raw data from Twitter in the JSON format. This data is usually quite noisy, for example there are many copies of a same tweet, while each tweet might contain text which is not relevant for our purposes. After cleaning the data, you will build a graph representing the co-occurrences of relevant "entities" in tweets.

2. extracting dense subgraphs from the latter graph. To this end, you should adapt the sequential greedy algorithm for finding dense subgraphs (which we presented during our class on Wednesday) so as to 1) deal with weighted graphs, 2) find "small" subgraphs 3) finding more than one dense subgraph, 4) deal with large graph, in particular try to give a linear implementation (in the size of the input) of the algorithm.

3. analyze the dense subgraphs that you found so as to find "interesting" information.

You should work in a group of two students. You should send us the code, as well as a short report (around 5 pages) explaining your findings and how you adapted the algorithm presented in class. More info on each of the tasks follow.

1 Task 1: Data Cleaning and Processing

1.1 Corpus

We have collected data from Twitter using their streaming APIs. You can download it from here `http://tinyurl.com/athensweekdatalocal`.

In the folder **data** you can find 4 datasets:

- the subfolder **Oscars** contains some of the tweets that have been posted by Twitter users during the Oscars this year (between Monday February 23rd and Tuesday February 26th of 2015). The tweets have been filtered using the hashtag #Oscars2015. [1]

- the subfolder **NewYork** contains tweets that have been collected using the hashtag #NewYork or by retrieving all tweets geotagged in New York (between Monday 23rd of February and Tuesday 26th of February 2015).

[1] hashtags in Twitter are not case sensitive

- the subfolder **ParisJanuary** contains tweets geotagged in the region of Paris for the entire month of January 2015.
- the subfolder **ParisFebruary** contains tweets geotagged in the region of Paris for the entire month of February 2015.

Each subfolder contains several files, with each file containing tweets from the corresponding day. Each file might contain several tweets separated by a newline character.

The tweets are in JSON format. For a complete list of the attributes of a tweet, visit this link: `https://dev.twitter.com/overview/api/tweets`. The attributes that we are interested are **text** and **entities.hashtags**.

1.2 Filtering

Data from Twitter is usually quite noisy for several reasons. One of the reasons is the presence of spammers, that is, users (possibly robots) that post repeatedly using trending hashtags in order to catch attention or users that post intensively using very similar text (or identical), without providing any new information. One basic way to limit this issue to some extent is to remove copies of same tweets. For our purposes, we say that two tweets are identical if they contain exactly the same set of hashtags. In this filtering step, remove the text from the tweets and keep only the hashtags. Then, remove duplicates. Example:

t1: "Terrorists at #CharlieHebdo #terrorists"
t2: "Attack at #ChalieHebdo #terrorists"
t3: "Terrorist attack at #CharlieHebdo #terrorists #Paris"

After the filtering we would have:

t1: "#CharlieHebdo #terrorists"
t3: "#ChalieHebdo #terrorists #Paris"

1.3 Build the graph

We shall build a graph representing hashtags co-occurrences as follows. For each hashtag occurring somewhere in the tweets there is a node, while there is an edge between two nodes if the corresponding hashtags co-occur at least once. Moreover, each edge is associated with a weight measuring the number of co-occurrences of the corresponding hashtags in tweets. The graphs is undirected.

2 Task 2: Finding Dense Subgraphs

2.1 Extract dense subgraphs

We will present an algorithm for the densest subgraph later during our class. For now just implement the algorithm presented in the slides 8 and 9 of the slides on finding densest subgraph posted on the website of the professor of this class.

After a good understanding of the algorithm, work on the following requirements:

- give a linear implementation (in the size of the input) of the algorithm presented in class (optional). Then give a definition of density for weighted graphs. In this case the weighted degree of a node is defined as the sum of the weights of the edges incident to such a node. Is there still a linear algorithm for the weighted case? If not, try still to provide a clever implementation of such an algorithm. Does the $2-$approximation guarantee still holds for weighted graphs?

- one practical problem of the algorithm presented in class is that it might produce very large graphs which are hard to analyze and might not be relevant for our task. Modify the algorithm so as to find a dense subgraph with number of nodes in the range $[2, 10]$. Do not worry about approximation guarantees for this task.

- our algorithm finds an (approximate) densest subgraph. Modify such an algorithm so as to find several dense subgraphs.

Output the results to a file **datasetname.subgr**. Compute the following extra information for each subgraph obtained:

1. density;

2. number of distinct users who contributed to the creation of the subgraph;

3. number of distinct tweets from which the subgraph emerged.

3 Task 3: Data Analysis

Run the dense-subgraph algorithm to find around twenty dense subgraphs for each of the four subfolders (NewYork, ParisJanuary, ParisFebruary, Oscars). Analyze those subgraphs, see what you observe and report your findings. You do not need to check manually all 80 subgraphs, interesting information can be found by checking only a few of them. Describe what you observe. Here are some examples of the analysis we expect to see in the report, however, you are free to report anything you find interesting (provided that you also motivate why you found it interesting). Be aware that this is still a quite noisy dataset, so not all subgraphs are relevant for our task or some of them might contain unrelated hashtags.

In January we observe some tweets related to an important event that happened in Paris. Which event? Also consider the subgraph obtained by the following hashtags: (#common, #selma, #johnlegend, #glory). Why are they occurring together? Glory is a song by John Legend and Common, won the Oscars for Best Song. Selma is an historical drama describing the events around Martin Luther King campaign for equal voting rights. In their acceptance speeches, the two musicians suggested that there is still a lot to do for ensuring equality in our society. By looking at the tweets we can see that people are discussing about that and we can also measure how popular is such a topic among Twitter users.

3.1 Optional: more accurate data analysis

So far we remove all text from tweets and retain only hashtags. For a more accurate analysis you might take into account the full text. This presents more opportunities but also more

challenges. You can use the following software from Stanford, the Stanford Log-linear Part-Of-Speech Tagger, `http://nlp.stanford.edu/software/tagger.shtml` to extract relevant nouns (such as city names, persons etc) which are not necessarily hashtags. Then build a new co-occurrence graph with the new nodes and find new subgraphs.

Other possible suggestions to improve the results. Merge misspelled tags of a same tag or merge singular and plural forms of a same noun.

4 Miscellanous

The following Java libraries can be useful for the project:

- **JSON.simple** to process files containing JSONs. The first step is to read the files line by line (each line corresponds to a tweet), parse the tweet and then extract the information needed. Pay attention to the fact that the JSONObject **entities** contains attributes with values of the type JSONArray.

  For more information, visit `http://code.google.com/p/json-simple/`.

- in order to easily construct and manipulate a graph you can use **JGraphT**, `http://jgrapht.org`. You can create an Object of the type **SimpleWeightedGraph** and gradually add nodes and increase the weights of edges.