Graph mining - WS 2016 - Project Report

Maxi Fischer, Sören Tietböhl

Due date: 24 January 2017

Summary

Analysis of the Spotify Artist Network using an ego-based community detection algorithm.

1 Summary of the data

Spotify is a music streaming platform, hosting over 30 million songs and more than 2 million artists. The artists are associated with each other via the 'related artists' feature. This relation induces a graph, which is the subject of this project.

Because the related artists have to be queried one-by-one from the Spotify API we are only using a subset of the complete graph. Our subgraph is constructed by starting at an arbitrary artist (we chose the red hot chili peppers) and then using breadth first traversal. The total number of nodes in the subgraph is 50733, the number of edges is X. The 'related artist' relation goes both ways, so its an undirected graph. We can assume that in reality the graph is dynamic, because new songs and thus artists will be added all the time.

Degree Distribution: ¡insert graph here¿

TODO: diameter, number of triangles

The artist nodes contain various information, such as genre and popularity.

2 Description of the algorithm

Ego-things here

Briefly describe in your own words the algorithm or the algorithm(s) you used in terms of input and output, which kind of network it applies to, and what is the main idea behind it (three sentences). Reference the algorithm (e.g., [**?**])

3 Preprocessing and storing

In order to obtain the graph we queried the Spotify API at [**?**]. The related artist query returns 20 artists that are related to the given input artist. From there we continued with breadth first traversal and queried the API for all the new artists. The graph is stored directly in main memory using python with networkx. The nodes are named by the artists id on spotify, which is returned by the API.

Report all the preprocessing steps to obtain the final network. How do you store the data (graph database, main memory, disk)? In which format? Did you have to remove some information in order to clean the data?

4 Network analysis

In this section, explain thoroughly the steps of the analysis you performed and give a motivation for such an analysis (e.g. discovering the most influential node, finding interesting patterns, analyzing communities). Elaborate on why you find it challenging and useful. Report possible applications.

4.1 Qualitative Results

Comment the results of the analysis. You should also mention all the relevant details of your analysis, for instance specific values for parameters you used. It is preferable that you show some graphics and present some examples. If you use a visualization method to depict the nodes of the graph, make sure that you can explain at least a part of the network in the plot.

For instance for node classification, show some of the classified vertices. If you have a ground truth, compare it to your results. You can also show a use case by running the algorithm on a smaller network (like the Star Wars network) and show what you find.

For a graph query, show how the algorithm behaves with particular patterns (stars, triangles, cores ...). Is the algorithm producing useful and interpretable results? Are the results unexpected or is it something you can find by yourself? Is there some anomaly?

4.2 Time and scalability

Evaluate the used method regarding its run time and scalability. Report the time you need to preprocess the data (if there is such a phase, like an index construction) and to produce the results. On the same network and algorithm, consider incremental graph sizes (e.g. 10% of the network, 30%, etc.).

Alternatively, generate some random network with `http://www.cse.ust.hk/graphgen/` or any other graph generator and compare the results of the algorithm between the two graphs (the original one and the random one). Are there any important differences in the outcome? Specify the parameters of the graph generator of choice.

4.3 [Optional] Memory usage

Report how much memory is consumed by the algorithm.

5 Limitations and interesting findings

Did you notice any limitation in the approach? Does it work as expected regarding the quality of the results? In terms of performance, is it too slow when the network is too big? Is it domain dependent or applicable to various types of graphs? To which kind of users are the results of the analysis useful? Describe what you have learned and propose ideas on how the analysis could improved.