*Abstract*:

**NGSA Project**

**Experimental evaluation of community detection algorithms**

**for musical tastes predictions**  
  
Florian Bettini – Maxime Lutel – Gabriel Muller – Benjamin Pommier

Our project aims to evaluate multiple community detection algorithms when applied to predicting musical tastes. We will therefore compare two approaches: one using classical clustering community detection algorithms, the other combining these algorithms with node embeddings.

Performances of both approaches will be evaluated on data collected from Deezer in November 2017 in 3 different countries: Romania, Croatia and Hungary. These datasets contain individuals (nodes) who have relationship (edges) and musical tastes (node attributes). Our approach is motivated by the increasing need in performance within an industry driven by a strong growth and an overwhelming pressure to generate profits.

# Motivation and Problem Definition

## The Problem

Improving the identification of communities within a musical network could benefit to both platforms and customers by drastically improving the quality and relevance of musical suggestions. The difficulty of assigning individuals to musical communities is that a relationship may not reflect a potential taste for a given genre of music. Consequently, we want to implement a node embedding in addition to a community detection algorithm, and then compare the results with classical community detection algorithms.

## Applications

Algorithms of community detection can be used for both:

* Suggesting new genres to individuals. For example, if someone belongs to a ‘pop’ community, but is not yet interested in ‘pop’, it could be relevant to suggest him new pop songs.
* Suggesting new relationships to individuals, thus it could be relevant to suggest a new friend to a user when both are within a same community.

## Related work

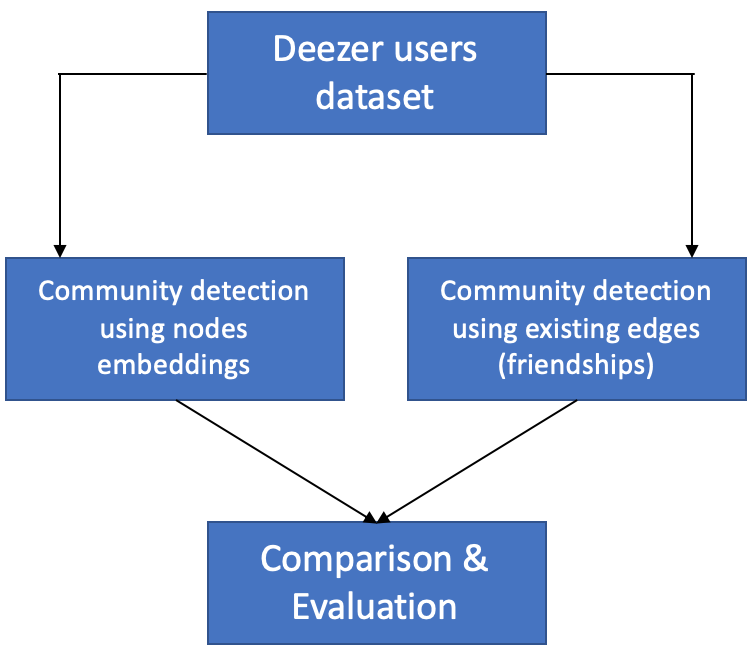
Community detection aims to discover groups of nodes on a graph such that the intra-group connections are denser than the intergroup ones. In social network graphs, communities are often overlapped. Hence classical algorithms such as Louvain or Girvan-Newman cannot address the issue of soft-clustering.

New algorithms were developed to address this topic, such as CPM (Clique Percolation Methods), Fuzzy Detection or NMF and PCA based methods, as mentioned in [6].

With the recent development of neural networks and deep learning, node embeddings are used for community detection. The general idea is to benefit from additionnal information from the nodes to generate a vector representation of the node and then learn the community by using unspervised machine learning techniques.

# Methodology

Our approach will be structured as in this diagram:



*Figure 1. Methodology*

As a first step, we will rely on classical clustering models to detect communities. This approach will be computed thanks to different kind of algorithms:

* Hierarchical clustering (FastGreedy)
* Spectral clustering
* Dynamic algorithms (Walktrap and Label propagation)
* Louvain method

The performance of these algorithms will be compared based on the modularity criteria.

As a second step, we will implement methodologies that learn node embeddings and use them to improve the clustering methods. The embeddings learning step will be performed with methods such as Node2Vec and Deep Walk, and the community detection will rely on a method combining these embeddings with clustering and regularization, which is called GEMSEC (Graph Embedding with Self Clustering). We will test variations of these methods to find the one that is the most suitable for community detection.

The comparison of these two approaches will allow us to understand whether embeddings improve community detection or not, and to quantify its impact on modularity and classification tasks.

# Evaluation

The Deezer dataset (available [*here*](http://snap.stanford.edu/data/gemsec-Deezer.html)) will be used for the entire project. On this dataset, each node has attributes (musical taste: between 1 and 84 per user) which can be predicted with a supervised method thanks to the embeddings learned for each node.

Our methodology results in two distinct outputs: clustering and embeddings. Each of them will be evaluated in a different way:

* Clustering quality will be evaluated thanks to modularity, which will be high if there are dense connections within communities and sparse ones between communities.
* Embeddings quality will be evaluated by using them to perform a multi-label classification task to predict the attributes of each node. We will compare the F1 score obtained when training machine learning models with these embeddings as features in order to determine which method gives the best embedded representation of nodes.

# Experiments & Results

As explained above, various experiments will be realized in order to attain an optimal clustering of the users:

* Multiple community detections algorithms will be evaluated (Girvan-Newman, Louvain, spectral clustering, …).
* Overlapping and non-overlapping community detection methods will be tested. Only the most fitted to the dataset will be conserved.
* With GEMSEC, new graphs will be generated based on the users’ music tastes. Continuous edge creation (each edge has a weighted corresponding to a similarity score) and discrete edge creation (an edge exists only if a similarity score is higher to a threshold (typically 0.5)) methods will be tested and compared.

## GEMSEC: Theory

## DeepWalk

xxx

## Clustering

xxx

## Regularization

xxx

## Impact of each layer

### Impact on modularity

This part aims to evaluate the performance of the GEMSEC algorithm in terms of classification. To do so, a modularity score is computed for each algorithm among (from the simplest to most sophisticated):

1) DeepWalk

2) DeepWalk with Regularization

3) GEMSEC

4) GEMSEC with Regularization

**Modularity formal definition**

However, we need to know if a node belongs to a specific community in order to compute the modularity of a graph (later expressed as ). Indeed, the definition of the modularity score Q is:

with:

* the adjacency matrix
* the degree of node i
* the number of edges in the graph
* the community of node i
* the Kronecker function

**Impact of each layer**

Fortunately, GEMSEC algorithms (3 and 4) provide the coordinates of the means of each cluster. Therefore, when computing the modularity, we assign each node to its closer cluster (eg the one with the closest cluster’s mean).

On the other hand, for evaluation purpose, we use a KMeans algorithm for DeepWalk algorithms (1 and 2) to provide communities to each node.

Therefore, the modularity score we compute is a great indicator of the clustering performance: the closer it is to 1, the stronger the clusters (and embeddings) are.

We performed this modularity analysis on 3 datasets: Deezer Croatia, Deezer Hungary, and Deezer Romania. Results are shown in Tables 1, 2 and 3.

|  |  |  |
| --- | --- | --- |
| **Algorithm** | Dataset | Modularity |
| **DeepWalk** | Deezer Croatia | 0.235 |
| **DeepWalk with Regularization** | Deezer Croatia | 0.287 |
| **GEMSEC** | Deezer Croatia | 0.354 |
| **GEMSEC with Regularization** | Deezer Croatia | **0.407** |

Table 1 – Modularity analysis – Deezer Croatia

|  |  |  |
| --- | --- | --- |
| **Algorithm** | Dataset | Modularity |
| **DeepWalk** | Deezer Hungary | 0.275 |
| **DeepWalk with Regularization** | Deezer Hungary | 0.292 |
| **GEMSEC** | Deezer Hungary | 0.351 |
| **GEMSEC with Regularization** | Deezer Hungary | **0.409** |

Table 2 – Modularity analysis – Deezer Hungary

|  |  |  |
| --- | --- | --- |
| **Algorithm** | Dataset | Modularity |
| **DeepWalk** | Deezer Romania | 0.468 |
| **DeepWalk with Regularization** | Deezer Romania | 0.484 |
| **GEMSEC** | Deezer Romania | 0.543 |
| **GEMSEC with Regularization** | Deezer Romania | **0.549** |

Table 3 – Modularity analysis – Deezer Romania

Comment on the difficulty to achieve great score due to the structure of Deezer’s graph. (lots of nodes, few edges).

**Impact of random walk’s order**

Definition of random walk (order 1 or 2)

To go even further, we decide to validate, or not, the results in [1] with regards to the order of the random walk. Indeed, results are shown to be better for the second order.

Based on the Deezer’s dataset in Romania, we get the following results:

|  |  |  |
| --- | --- | --- |
| **Algorithm** | Dataset | Modularity |
| **DeepWalk** | Deezer Croatia | 0.235 |
| **DeepWalk with Regularization** | Deezer Croatia | 0.287 |
| **GEMSEC** | Deezer Croatia | 0.354 |
| **GEMSEC with Regularization** | Deezer Croatia | **0.407** |

Table 1 – Modularity analysis – Deezer Croatia

### Impact on multi-label node classification

GEMSEC algorithm aims at improving the embedding for further clustering tasks. As we focused on the Deezer dataset, another use of these embeddings may be to predict the music that a user may learn, by analyzing only its relationships. As each of the user (node) likes one or several genres (Pop, Rock, etc.)), we face a multilabel classification problem. We adopted the same methodology as [1] to compare the different embeddings.

First, we start by embedding all the graphs are embedded (using GEMSEC, DeepWalk, Node2Vec, etc.) in . The embeddings are then used separately as features for the multi-label classification problem. To be able to compare the various embeddings, we choose to use a OneVsAll approach with a LogisticRegression (C=1). Hence, with this method, a classifier will be trained to determine whether a user likes the genre or not by genres. As the computation was high and the number of genres were high, we reduced the number of genres to 40 (instead of 88). This approach is enough to estimate the performance of the embeddings because we have all the most represented classes. After 40 the percentage of “genre liked” is below 1%.

During the training process, we might also have included a gridsearch. It was tried to optimize the parameters, but we didn’t include in the final results because:

* Optimal parameters were not the same for all models, hence introducing a bias in the comparison
* Running a real gridsearch with enough parameters to significantly observe an increase in performance was too computationally expensive

The objective function we use to analyze the performance of the algorithm was the F1-Score. Micro, macro and weighted F1-Score were used.

Finally, as we can observe in the table below, GEMSEC outperforms the other embeddings for the classification task. Also, a small positive influence is observed for the regularization process.

INSERT TABLE WITH FIGURES FOR ALL THE EMBEDDINGS

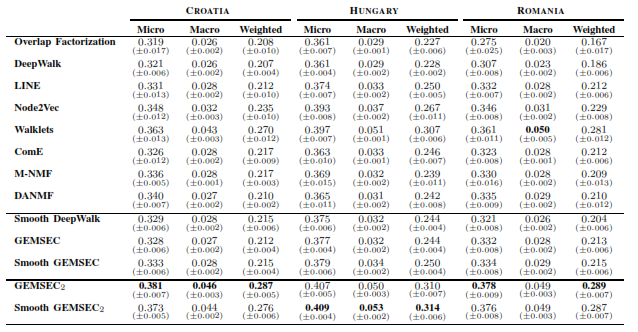


Figure 3 – Classification task

## Going further

### Modifications in Pytorch

xxx

### Modifying clusters’ means at each epoch

xxx

## Performance vs. other algorithms

xxx

### Performance on modularity

xxx

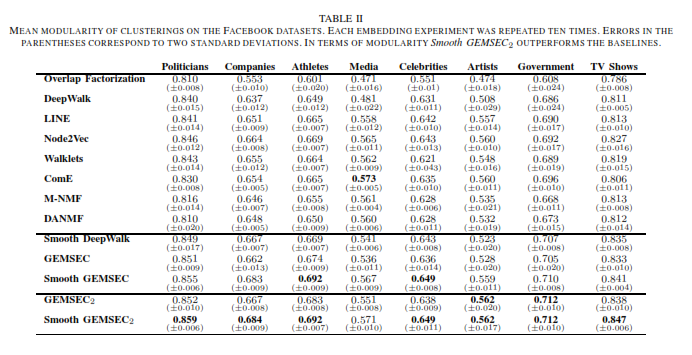


Figure 4 – Modularities

- on Facebook dataset

- on Deezer dataset

- Partie de Gab sur les différentes modularités (Louvain, ...)

### Performance on the classification task

xxx

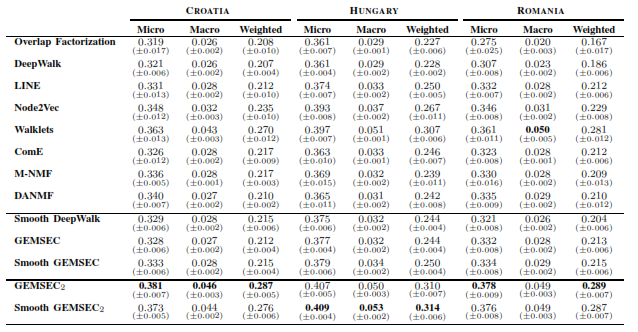


Figure 3 – Classification task

# Reference

Some references were selected in order to support our approach. As our focus will mainly be on the comparison of algorithms on a specific dataset, we will mainly rely on [5] to implement the algorithms and compare our results.

# References

|  |  |
| --- | --- |
| [1] | B. Rozemberczki, R. Davies, R. Sarkar and C. Sutton, "GEMSEC: Graph Embedding with Self Clustering," *ASONAM,* 2019. |
| [2] | M. Aqib Javed , M. Shahzad Younis, L. Siddique , Q. Junaid and B. Adeel, "Community detection in networks: A multidisciplinary review," *Journal of Network and Computer Applications,* vol. 108, pp. 87-111, 2018. |
| [3] | L. Ye, S. Chaofeng , H. Xin and Z. Yanchun, "Community Detection in Attributed Graphs: An Embedding Approach," in *AAAI-18*, 2018. |
| [4] | C. Sandro , V. W. Zheng, H. Cai, K. Chen-Chuan Chang and E. Cambria, "Learning Community Embedding with Community Detection and Node Embedding on Graphs," in *CIKM*, Singapore, 2017. |
| [5] | J. Chitra Devi and E. Poovammal, "An Analysis of Overlapping Community Detection Algorithms in Social Networks," in *IMCIP*, 2016. |
| [6] | T. Cunchao, Z. Xiangkai and W. Hao , "A Uniﬁed Framework for Community Detection," *Journal Of Latex Class Files,* vol. 14, no. 8, 2015. |
| [7] | N. Schlitter and T. Falkowski, "Mining the Dynamics of Music Preferences from a Social Networking Site," in *International Conference on Advances in Social Network Analysis and Mining*, Athens, 2009. |