Algorithms in Data science
Final Project:
STUDY OF COMMUNITIES IN GITHUB WITH A MEMBERSHIP NETWORK

Manuelle Ndamtang, 66732000

December 2021

Introduction

Graphs occupy a very important place in machine learning. Easy to implement and especially to represent, they can be applied in all domains. A particular type of graph attracts our attention: it is indeed a bipartite graph. The fact that it gathers at the same time two types of nodes makes it rich in approach perspectives. In this project, the graph on which we will work is a large-scale membership network.

The main objective here is to proceed to the detection of the communities in the bipartite graph in order to determine the different communities existing in a graph, to calculate the different proportions of them in order to determine the number of nodes of each type for a given community. The second step will be to set up a model that could predict the community to which a node would belong.

1 Presentation of the dataset

The chosen dataset comes from the Github Contest of 2009. It is made of data allowing to have a bipartite graph made of 177 336 nodes with users and projects. The edges represent the fact that a user is a member of a project and there are 440 237 edges. The number of users is 56 519 and the number of projects is 120 867. We are dealing with a disconnected graph, with 15 067 subgraphs, On the users' side as well as on the projects' side, we notice that the graph is not so dense. Indeed, the density of user nodes is 0.000064 and for the projects nodes is 0.000064.

When we study particularly the subgraphs, we notice that they are indeed not uniformly distributed. The largest subgraph has a size of 139 852 nodes . As for the rest of subgraphs, their sizes vary between 2 and 45 nodes(figure 1).

2 Communities Detection

To determine the community in a bipartite network, there is two approaches:

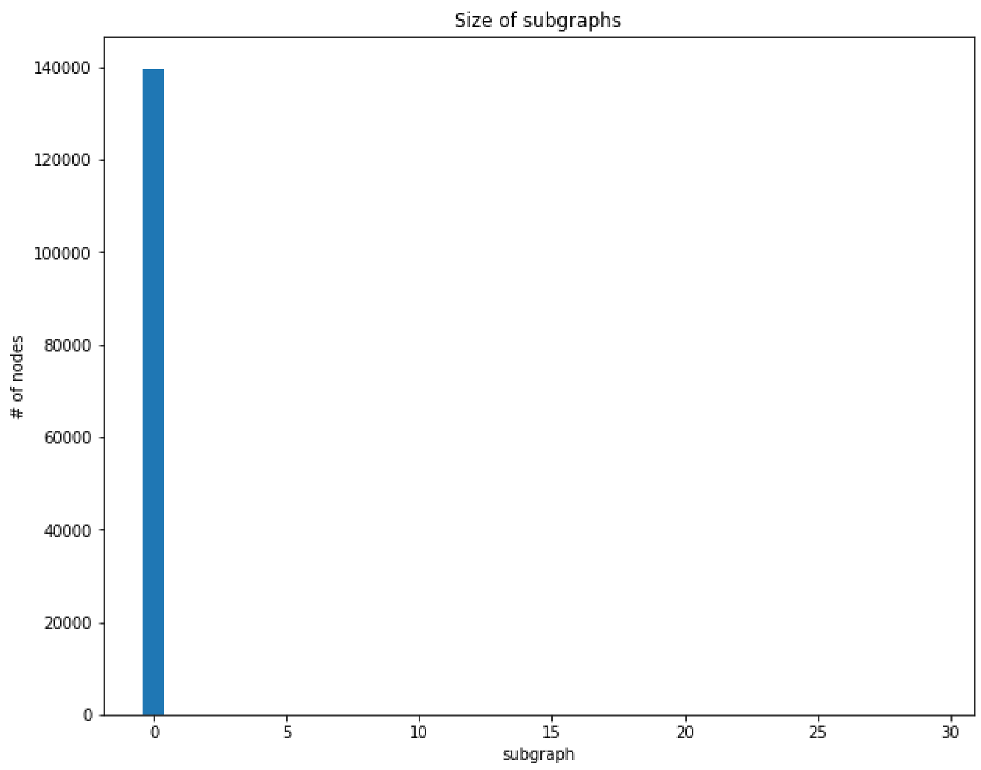

Figure 1: Size of 30 biggest subgraphs ordered

- The `one-to-one communities detection`: Here a community is a set of nodes that are densely connected. To process for the detection of communities, we have to compress the bipartite network into a one-mode projection and apply the community detection algorithm for unipartite graph. The advantage is the easiness to apply it. But the main drawback is that the projection might bring loss of information.;
- The `many-to-many communities detection`: Here a community is a set of node that have the same type and have similar link patterns. The advantage of this correspondence community algorithm is that it keeps the the initial structure of the graph while extending unipartite algorithms.

.

2.1 Community detection in the graph using the bipartite Louvain algorithm

It's a one-to-one correspondence community algorithm. The library used comes from the paper "Improving performances of Top-N recommendations with co-clustering method". The idea is to apply Louvain algorithm on a projection of the graph in order to detect cluster.

For each subgraph, the algorithm was applied and in total, we got approximately 37 349. We noticed that the algorithm is not really stable because each time we run the algorithm, the number of communities is not the same.

And if we look at the statistics of the biggest subgraph, we ended up some big communities as presented in the Figure 2. This figure represented the first 30 biggest communities in the largest subgraph.

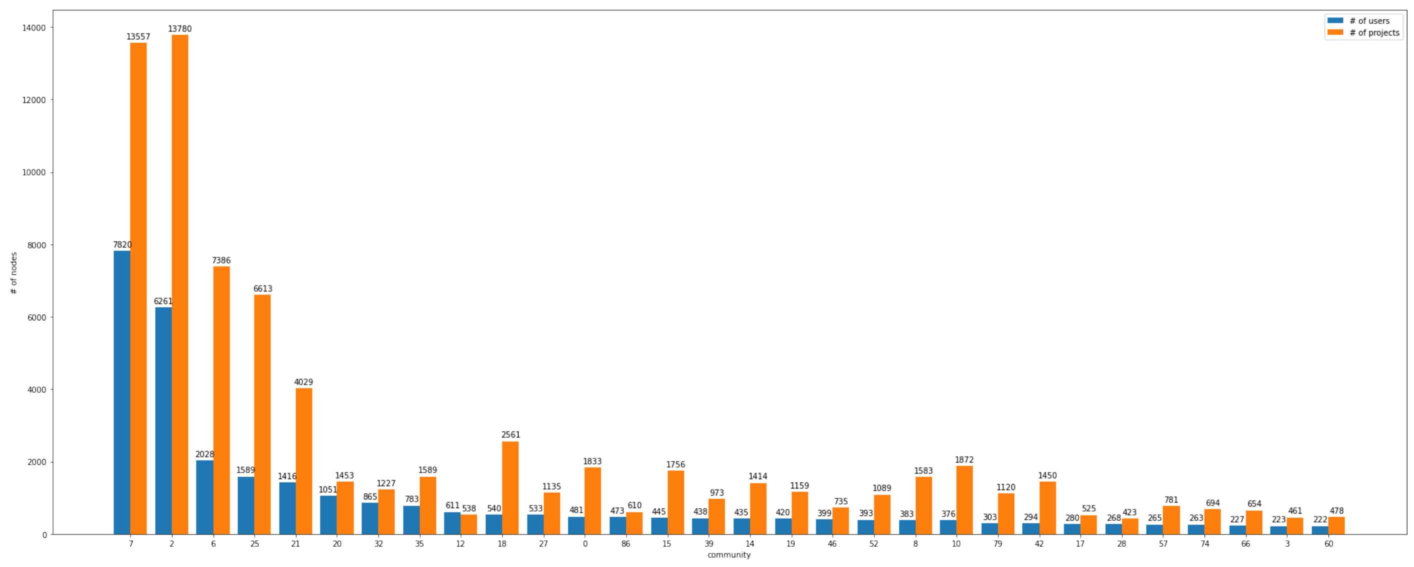

Figure 2: The 30 biggest communities in the largest subgraph using bipartite Louvain algorithm

This subgraph has approximately 2 668 communities.

For the rest of subgraphs, the number of communities vary between 2 and 30. We didn't display the size of communities as chart because it might be less informative given the number of communities in the biggest subgraph.

2.2 Community Detection in Bipartite Networks by Multi-Label Propagation

This algorithm is a many-to-many community detection algorithm. It's based on the label propagation procedure specific for a bipartite networks. The library used in the project to apply it comes from the paper "BiMLPA: Community Detection in Bipartite Networks by Multi-Label Propagation". The advantages of this algorithm is that it has and near-linear time complexity, it's stable and can automatically detect communities without any a knowledge of the number of communities within the graph. BIMPLA uses multi-label propagation algorithm on bipartite network in order to detect communities.

After applying this algorithms for each subgraph with some parameters such as the number of iterations (we keep the default one: 100), the threshold (0.3) and the maximum number of labels (7) , we finally got in for the biggest sugbraph 13 365 communities for the users nodes and 12 110 communities for the projects nodes. The figure 3 illustrated the 30 biggest communities for this subgraph. As you can see, it seems like the algorithm has converged for the largest subgraph even though we apply 100 iterations. The time spent to get communities in the entire graph is approximately 30min using GPU in Colab application. Maybe to improve the result, set the number of iterations for the propagation of labels to a big number might improve the number of communities detected.

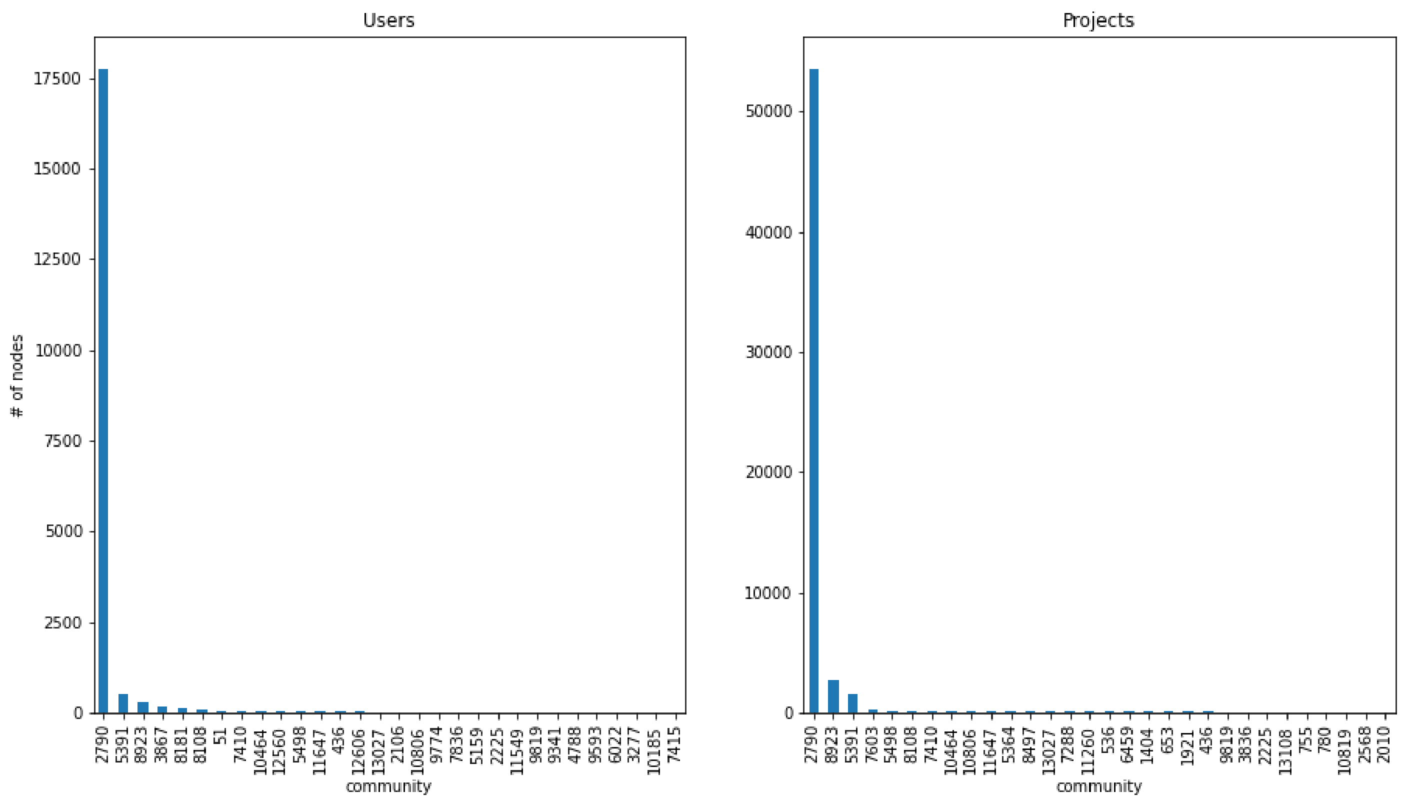

Figure 3: The 30 biggest communities in the largest subgraph using BIMPLA

3 Unsupervised Machine Learning

After trying some algorithms to detect communities using a one-to-one and a many-to-many correspondence, one question we can ask is: is it possible to create a machine learning model in order to predict if a particular node belong to a precise community? To answer to this question, we have to deal with the graph size. That's why we decide for this case to select two not big communities.The main idea here is to show how we can create such model for two communities and apply it in the future for all the communities if we wish.

3.1 Data preprocessing

To create the model, we select the 5th and the 11th largest community subgraph using the Bipartite Louvain algorithm and join them in order to create a graph. The tools that we will use in this part of the project is StellarGraph, Word2Vec and Scikit-learn.

3.1.1 The Graph embedding

To create a machine learning model for the graph, we have to transform the nodes to vectors. That is what is about the graph embedding. The idea is to extract network structure an use it in machine learning algorithm.

The process of translating relationships and network structure into a set of vector is call node embedding. So a node embedding algorithm encode each node into an embedding space while preserving the network structure.

In the project, the Node2vec algorithm is used with a second-order biased random walks to generate sentences from a graph. A second-order biased random walks take in account both the current state at a particular node as well as the previous state. A sentence is a list of node ids and the set of all sentences gives a corpus.The corpus we get is then used to learn an embedding vector for each. node in the graph.

The StellarGraph library is employed here to provide an implementation for second-order random walks, with a fixed maximum length and controlled parameters p (the return parameter, in others words the likelihood of backtracking the walk immediately revisiting a node in the walk) and q (The inOut parameter, allowing the traversal calculation to differentiate between inward and outward nodes).

Then we use the Word2Vec library in order to learn representation for each node in the graph. The dimensionality of the learning embedding vectors is set to 128.

3.1.2 The model creation

For the creation of the model, we decide to used an unsupervised learning algorithm: Kmeans. Here, the purpose is to find possible clusters given the fact that we known, there is already 2 communities. To make the clusters visualization much easy, we decide first to reduce the dimension space of the features space (from 128 to 3) using PCA and then apply the Kmeans algorithm with the number of components 2. The result of the algorithm is represented in Figure 4 for the dataset.

4 Things to improve

Coming to the end of the project, we think that there a many things to improve in the project, but given the deadline, it wasn't possible to make it. These improvements are:

- the number of iterations for the detection of communities using BIMPLA algorithm
- the model training: instead of proceeding to the unsupervised learning, we can proceed to a supervised learning model using a target which are the community Id. We can split the model to a training and the test set a evaluate the accuracy of the model for the prediction of community.
- Usage of manifold algorithm for visualisation of node embedding: PCA helps us to reduce the dimensionality of the feature space. And when we reduce the feature space, we also lose lot of information. That is shown by the explained variation per principal component: `[0.10161072 0.08472438 0.05536379]` which is not really significant. So instead of proceeding to dimension reduction, we could keep the feature space and create the model with it and use a manifold tool such as TSNE (T-distributed Stochastic Neighbor Embedding) in order to visualize the hight-dimensional data.

Conclusion

At the end of the project, we can say that the bipartite graph is a graph with huge study possibilities. Another possibility would be to project ourselves into a unipartite graph of users and projects and then

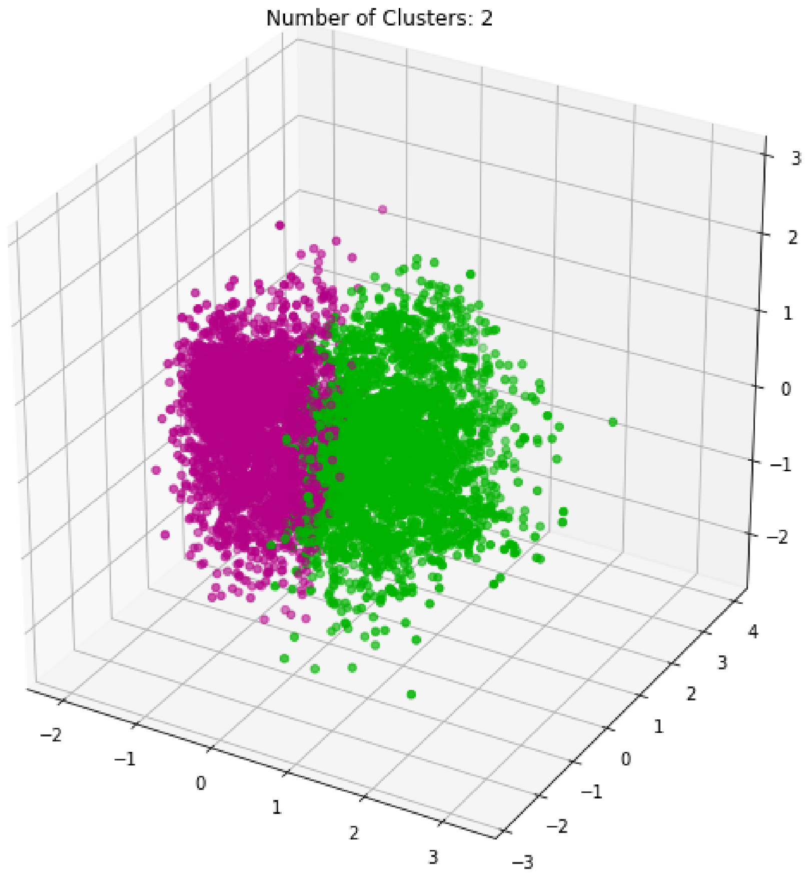

Figure 4: The community detection using Kmeans

try to detect communities; but the biggest concern with this approach is that the size of the graph does not allow us to do such a process.

For example the projection of user nodes would certainly give millions of edges and with NetworkX as a library, it is difficult to work with the large-scale graph. It would be interesting to have libraries able to handle large-scale graphs more efficiently to have more consistent results.

Links

- The dataset: `http://konect.cc/networks/github/`
- the library of the bipartite community detection with bipartite louvain algorithm: `https://github.com/THUfl12/bipartite-louvain`
- library for of the bipartite community detection with BiMLPA: `https://github.com/marblet/BiMLPA`
- Paper: ”BiMLPA: Community Detection in Bipartite Networks by Multi-Label Propagation”, Authors: Hibiki Taguchi, Tsuyoshi Murata, Xin Liu
- Paper: ”Improving Performances of Top-N Recommendations with Co-clustering Method”, Authors: Liang Feng, Qianchuan Zhao, Cangqi Zhou