

Using Graph Algorithms for Unsupervised Learning

Exercise: Community Detection. Verify the communities of American College Football Conferences, by analyzing the graph of team-team games.

In this exercise, we will load a small graph with natural communities into the TigerGraph graph platform, install the Louvain Modularity algorithm from the graph algorithm library, run the algorithm, and visualize the result.

We will then repeat this for a much larger algorithm where it is not practical to visual the graph, but we can analyze the output data.

Data Set: network of American football games between Division IA colleges during regular season Fall 2000. From *M. Girvan and M. E. J. Newman, Proc. Natl. Acad. Sci. USA 99, 7821-7826 (2002)*.

Algorithm: Modularity-based Community Detection, using the Louvain method for computing modularity, with enhancements for parallelism. For a description, see <https://docs.tigergraph.com/graph-algorithm-library#louvain-method-with-parallelism-and-refinement>

Graph Schema: One vertex type, Football_Team, and one undirected edge type, Football_Edge. The edges have one FLOAT attribute called "weight", required by the algorithm (see below). The vertices have one INT attribute called "score" for storing the output.

Instructions:

1. In a Linux shell for your TigerGraph machine, change to the football folder:
`cd /home/tigergraph/usecases/football`
2. Create the graph schema for the Football graph:
`bash football_schema.sh`
3. Load the graph data into the Football graph:
`gsql football_load.gsql`
4. Let's list out the graph catalog so far, using the LS command. Since we now have at least one graph, GSQL requires that we specify which graph we want to work with:

```
gsq1 -g Football ls
```

5. Now we want to run the GSQL graph algorithm installer. The algorithm library's top level folder is at /home/tigergraph/gsq1_algorithms. The install.sh script is in the algorithms subfolder. To avoid having to change to that folder, we can create another small script to change to the right directory:

```
vim ~/alg-install.sh
```

Insert the following text into the file:

```
#!/bin/bash
cd /home/tigergraph/gsq1-graph-algorithms/algorithms
bash install.sh
```

6. Now run the installer. We will walk you through how to answer each of the prompts.

```
bash alg-install.sh
```

```
> Graph name?
```

```
Football
```

```
> Please enter the index of the algorithm you want to create or
EXIT:
```

```
(The installer will list each available algorithm and a number.
Enter the number for Louvain Method with Parallelism and Refinement)
```

```
6
```

```
(At this point the installer will show you the schemas of the
available vertex and edge types. Note not just their names but also
the attributes, because you might need to use some of the attributes
for reading or writing values used by the algorithm.)
```

```
> Vertex types:
```

```
Football_Team
```

```
> Edge types:
```

```
Football_Game
```

```
> Edge attribute that stores FLOAT weight:
```

```
weight
```

```
> Please choose query mode:
```

```
1
```

```
> Do you want the algorithm to update the graph [yn]?
```

y

Choose a way to show result:

4

> Vertex attribute to store INT result (e.g. component ID):

score

(It is now finished asking about one algorithm. It now goes back to see if you want to configure another algorithm. This time, select "1" for EXIT:

> Please enter the index of the algorithm you want to create or EXIT:

1

> Algorithm files have been created. Do you want to install them now [yn]?

y


It will take about a minute to install the algorithms.

7. Let's switch to GraphStudio. In a web browser, preferably Chrome, use the URL for your EC2 machine, and append ":14240". E.g. if your machine's URL is <http://ec2-1-2-30-400.us-east-2.compute.amazonaws.com> then use this URL: <http://ec2-1-2-30-400.us-east-2.compute.amazonaws.com:14240> Note: GraphStudio works with most web browsers, but Microsoft IE and Edge are not supported.
8. Before running the graph algorithm, let's take a quick look at the other tabs, so you become familiar with GraphStudio. In the upper left, make sure it says Football. If it doesn't, click the down arrow next to the graph name and select Football.
9. Go to the Design Schema tab. See the vertex types and edge types we created. Move the cursor over a vertex type or edge type to see its attributes.
10. Go to the Map Data to Graph tab. (GraphStudio may warn you about unsaved changes. This is because the schema was created from the console instead of through GraphStudio. You can click "Leave"). The Map tab corresponds to the loading job you created with the football_load.sql script.
11. Go to the Load Data tab. Here you run a loading job, and on the right, the table lists how many of each vertex type and edge type have been loaded. You can see that there are

115 vertices (football teams) and 613 edges (games).

12. Go to the Explore Graph tab. Here you search the neighbors of selected vertices, or look for shortest paths between vertices.

13. Go to the Write Queries. Here you can write queries for scratch, edit existing ones, or run queries. GraphStudio has already added the queries that you created with the GSQL Graph Algorithm Library installer.

14. Select `louvain_parallel_file`. Click the RUN arrow at top  and set the following parameter values:

`iter1: 10 (default)`

`iter2: 10 (default)`

`iter3: 10 (default)`

`split: 10 (default)`

`outputLevel: 1`

`fComm: /home/tigergraph/usecases/football/football.comm`

`fDist: /home/tigergraph/usecases/football/football.dist`

`display: true`

Scroll down the parameters all the way to see the Run Query button and click it.

15. When the algorithm has finished, you can see the output displayed as a graph. You notice some naturally groupings. This is NOT due to the algorithm. This is due to connectively pattern of the graph, and the graph layout algorithm ("Force-based") which is making decisions about where to draw each vertex and edge, based on the principle that vertices that are highly interconnected should be close together. In other words, What we need to do is verify that the Louvain algorithms cluster assignments are consistent with (1) the graph layout, but more importantly (2) their true football conference memberships.

16. The algorithm assigned each community an ID number = the least vertex ID number among its members. To see these community IDs, we can look at the end of the output for the query. Back in your shell console, display the contents of the community distribution output file:

```
cat ~/usecases/football/football.dist
```

You should see the following:

```
6,104857601
5,103809025
14,102760449
12,103809024 134217728
```

```
11,104857602
9,104857600 103809026 134217730
8,102760448
10,103809029 102760450
```

This means:

There is one community of size 6, whose ID is 104857601.

...

There are two communities of size 12, whose IDs are 103809024 and 134217728.

17. GraphStudio lets you color all the vertices that satisfy a given property.

...

18. We can also look at the output file with the Louvain-calculated community membership for each vertex, and compare those IDs to the actual football conference groupings.

View the contents of the file `football.comm`. Because the lines are in random order, it's hard to tell the community groupings. Moreover, we have no direct way to compare if the predictions match the actual conference members. We can fix that.

a. On line 287, you should see



```
fComm.println(s, s.@cid)
```

This says print the team id (`s`) and the team's predicted community id (`s.@cid`).

b. Edit the line to insert two additional fields before the current two:

```
fComm.println(s.conference, s.name, s, s.@cid)
```

We are putting conference ID first, so we can just sort the links later, which will accomplish our grouping.

19. Save  and Install  the modified query. Run the query again with the same parameter values as before. Sort the result:

```
sort football.comm > football.comm.sorted
```

20. View `football.comm.sorted`:

```
vim football.comm.sorted
```

BONUS: If time allows, we can load a larger social network (Facebook) and examine some additional graph properties.

Source: <https://snap.stanford.edu/data/ego-Facebook.html>

This page provides details about the source, the graph size, and several graph characteristics, which we can verify with the GSQL graph algorithm library.

21. In a Linux shell for your TigerGraph machine, change to the football folder:

```
cd /home/tigergraph/usecases/social
```

22. Create the graph schema and load the data:

```
gsql social_create.gsql
```