COMP 370 Homework 10 – Network Modeling

Assigned Nov 28, 2023
Due Dec 4, 2023 @ 11:59 PM

In this assignment, we're going to practice the fundamentals of network modeling by analyzing the conversation network of the My Little Ponies.

Task 1: Build the MLP interaction network

In this assignment, we are modeling the interaction network as who speaks to who. There is an (undirected) edge between two characters that speak to one another. The weight of the edge is how many times they speak to one another.

To keep our life simple, we're going to use a very simple proxy for "speaking to one another". We will say that character X speaks to character Y whenever character Y has a line IMMEDIATELY after character X in an episode. So, for the following excerpt of dialog from an episode…

> Twilight Sparkle: Hey Pinkie. Sorry I accused you of being nosy.
> Pinkie Pie: It's okay, Twilight – we all have our rough days.
> Applejack: Come on, everypony! We need to get a move-on.
> Spike: Hurray!

In this excerpt, we have the following interactions:

> Twilight Sparkles speaks to Pinkie Pie
> Pinkie Pie speaks to Apple Jack
> Applejack speaks to Spike

Of course, from the text, we can see that Pinkie Pie's comment wasn't *really* addressed to Apple Jack… highlighting how proxies can be wrong. But for this assignment, we're going to assume it's a good proxy.

Keep the following in mind:

- A character can't talk to itself
- We're considering the top **101 most frequent** characters, not just the ponies. Frequency is defined by # of speaking lines.
- Don't include characters that contain the following words in their names: **"others", "ponies", "and", "all"**. For example, "Fluttershy and other ponies" should not be considered.
- When skipping these records, the speaker that comes before won't have **ANY** follow on speaker (i.e., a plural character like "Fluttershy and other ponies" nulls out the interaction it's involved in)
- Respect episode boundaries – interactions shouldn't carry over.
- We are counting UNDIRECTED interactions – this means that if Spike speaks to Applejack and later Applejack speaks to Spike … then the number of interactions between them is 2.
- Match characters by exact name. For example, do not bother to build an equivalence between "Twilight" and "Twilight Sparkle" if you encounter both types of references in the dialog. Same applies to "Young Applejack" and "Applejack"; we know they are the same, but your code should treat them differently.

Your script, **build_interaction_network.py**, should work as follows

```
python build_interaction_network.py -i /path/to/<script_input.csv> -o
/path/to/<interaction_network.json>
```

where the interaction network json file should have structure…

```
{
      “twilight sparkle”: {
            “spike”: <# of interactions between twilight sparkle and spike>,
            “applejack”: <# of interactions between ts and aj>,
            “pinkie pie”: <# of interactions between ts and pp>,
            …
      },
      “pinkie pie”: {
            “twilight sparkle”: <# of interactions between ts and pp>
            ...
      }
      ...
}
```

For consistency, lowercase all character names in the output JSON.

As usual, the arguments -i and -o should take any path. Indentation won't matter in this exercise; you can have a one-line JSON file or a pretty-printed one.

Task 2: Compute interaction network statistics

Write the script **compute_network_stats.py** which is run as follows:

```
python compute_network_stats.py -i /path/to/<interaction_network.json> -o
/path/to/<stats.json>
```

Using the networkx library, compute

- The top three most connected characters by degree centrality.
- The top three most connected characters by *weighted* degree centrality (each edge contributes its weight = # of interactions)
- The top three most connected characters by closeness centrality.
- The top three most central characters by betweenness centrality.

Your output file should have structure:

```
{
      “degree”: [c1, c2, c3],
      “weighted_degree”: [c1, c2, c3],
      “closeness”: [c1, c2, c3],
      “betweenness”: [c1, c2, c3]
}
```

As usual, the -i and -o arguments refer to any path (and not just a file name). Indentation won't matter in this exercise; you can have a one-line JSON file or a pretty-printed one.

Submission instructions

Submit the following in hw10.zip

- build_interaction_network.py
- compute_network_stats.py
- stats.json (the output of Task 2, run on the interaction network derived from all the MLP data)