COMP 598 Homework 9 – Network Modeling

30 pts
Assigned Nov 18, 2021
Due Nov 26, 2021 @ 11:59 PM

Non-standard (i.e., not built-in) python libraries you can use:

- Pandas
- networkx
- numpy

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 (15 pts)

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.
- 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 (i.e. we will call them with our own files, which can be in any location). Indentation won't matter in this exercise; you can have a one-line JSON file or a pretty-printed one.

Examples

1. *applejack*

   All

   fluttershy

   Applejack has no follow-on speaker, since "All" should be removed. Fluttershy starts a new chain.

2. *applejack*

   random pony not in top 101

   fluttershy

   Applejack has no follow-on speaker, since "random pony" is not among the 101 most frequent speakers and should not be considered. Fluttershy starts a new chain.

3. *applejack*

random pony not in top 101

fluttershy

fluttershy

all

applejack

We'll have applejack (no follow-on), then fluttershy (no follow-on, since the next line it's the same character, followed by an "all"). Then lastly, applejack. Applejack starts a new chain.

You must submit the results of `build_interaction_network.py` in a json file named **interaction_network.json**, which should be placed on the root of the submission folder. Please check the Github template for further details.

Task 2: Compute interaction network statistics (15 pts)

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 # of edges.
- The top three most connected characters by sum of the weight of edges.
- The top three most central characters by betweenness.

Your output file should have structure:

```
{
    “most_connected_by_num”: [c1, c2, c3],
    “most_connected_by_weight”: [c1, c2, c3],
    “most_central_by_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.

You must submit the results of `compute_network_stats.py` in a json file named **stats.json**, which should be placed on the root of the submission folder. Please check the Github template for further details.
Please check hw9 submission template for more information. Please read the instructions carefully - https://github.com/druths/comp598-2021/tree/main/hw9