Paulina Toro Isaza
CSCI 233
Project Report
May 25, 2016

Overview

Marvelist displays the Marvel comic universe's social network information in an appealing and comprehensive manner. The idea for this project came from my interest in social networks and having played around the Marvel Social Universe Graphs data.

The program fulfills two main tasks:

1. Showcasing information about an individual character and their network
2. Finding the common appearances between two characters

The program only covers characters and issues up until 2001.

Data

- **Marvel Universe Social Graphs** by Kai Chang, Tom Turner, and Jefferson Braswell
  http://exposedata.com/marvel/
  Comic and hero network data: Table that lists each character that appears in each issue. Only contains issues until 2001.
- **Marvel Developer Database (Official)**
  http://developer.marvel.com/
  Character thumbnail image
  Full name (secret identity)
  First appearance (issue and year of publication)
- **Marvel Chronology Project (Fan-run)**
  http://www.chronologyproject.com/key.php
  Key for issue codenames used in Marvel Social Universe Graph data.
- **Marvel Wikia (Fan-run)**
  http://marvel.wikia.com/wiki/Marvel_Database
  Publication year for each issue

Goals

- Have two characters met?
  - ☑ List of shared appearances in chronological order
  - ☒ Series with most shared appearances
    This was more of a stretch goal and not as interesting as the others. The code would be rather trivial to add but for the sake of time I did not work on this.
  - ☑ Images of the two characters
  - ☑

Year each character was created and first year of meeting

The first year of meeting is the first issue on the list and has the accompanying year of publication.

☑ Link to individual networks

User can click on the character's name to open a window of the individual's network.

- Individual's network
  - ☑ Images and names of the five characters with which the individual shares the most appearances
    i. Links to the common appearances
  - ☑ Character thumbnail image
  - ☑ Size of network (Degrees)
  - ☑ Year created (and issue of first appearance)
  - ☑ Network measurements
    ii. Closeness centrality
    iii. Betweenness centrality
  - ☒ Location in macro network visualization

    I was unable to get matplotlib to work in a virtual environment and so could not draw the graph.

Tools

1. Python program that separates name from superhero code name in the hero_in_comics table from Marvel Universe Social Network.

   Example entry: `IRON MAN/TONY STARK` →

   ```
   Name: Tony Stark
   Codename: Iron Man
   ```
2. Python program to populate SQL table with all Marvel API entries.

   See Section: Problems for more details.
3. Python program to separate issue in issue table by name, volume, and issue.
4. Webscrapper using Marvel API that scraps the first appearance of the character from the Marvel Developer Wiki
5. Python program to grab relevant character info from Marvel API
6. Python program to get issues from SQL database and turn into a usable URL for Marvel Wikia.
7. Webscrapper that scraps Marvel Wikia issue page to find the year of publication for an issue.
8. Python program that creates new table based on matches between API table and network table. The code searches for matches by code name or real name.
9. Function that finds all issues in which both heroes appear. Uses the Wikia Issue scrapper to get the issue's year of publication, assigns it as a key, and sorts all common appearances by year.
10. Modification of Mitja Martini's autocomplete Tkinter widget http://tkinter.unpythonic.net/wiki/AutocompleteEntry
11. Single character object class that holds identity and network information of the character
12. GUI for individual character's network screen
13. GUI for common appearances screen
14. GUI for start screen that allows user to input character names

Problems Encountered

- Data needed cleaning
  hero_in_comics table had more than one comma per line, so had to convert relevant delimiter to semi-colon.
  Source code for Marvel Chronology Project keys had HTML typos that were interfering with the scrapper. I had to fix the HTML typos manually and save the new code in an html file.
- Connecting to API's or scrapping websites without getting kicked off or exceeding call limits.
  To get a list of all Marvel API characters, I had to manually search for each character in 100 increments. For example, first I would call characters 0 – 99, then 100 – 199, etc.
  I was unable to get the publication year of each issue in the database without getting kicked off by the Marvel Wikia. I decided to scrap for the publication years for only the issues currently being used in the Common Appearances screen. This means that the program runs a lot slower.
- Dealing with inconsistencies in data
  It was hard to generalize fixes for such inconsistencies. Many of them resulted from characters in the **heroes_in_comics** table not being matched to the Marvel API entry even if the match did exist. For example, the character Vision had a type that included a space at the end of his name in the **heroes_in_comics** table but not in the Marvel API table. Other inconsistencies dealt more with characters going by more than one name. An example would be Jean Grey, who is referred to as Phoenix in the **heroes_in_comics** table but as Marvel Girl in the Marvel API.
- Some characters have different errors when trying to create the individual character network screen or common appearance screen.
  See Section: Bugs for more details.
- Having complete GUI on one window
  Instead I used TopLevel to open up a new window every time a user clicks on a link. A better solution would have been to use tabs.
- I had a lot of problems installing python modules on my laptop because pip did not work.
  Switched to my sister's computer and had no issues.
- Using matplotlib in virtual environment
  Was unable to solve this and so could not draw graph of the network.
- The program takes a long time to load
  The Individual Character Network screen calculates the betweenness and closeness centrality measures for every character in the network. The solution would be to have a separate mini-program that calculates this information and inserts into the Character table.
  The Common Appearances screen has to scrap the web page for each issue in order to get the year of publication. For characters with a large number of common appearances, this can take a while. One solution would be to have the scrapper insert the years into the relevant table so that subsequent comparisons between those two characters will load automatically.

Project Expansions

- Include characters and issues after 2001
  This will require using Marvel API's issue information as opposed to the Marvel Universe Social Network data.
- Include more of the Marvel API characters
  Create a small program that asks the user to input the **heroes_in_comics** name and the Marvel API name to create a new match and update all relevant tables with the match.
- Network visualization

Find a way to use matplotlib or find a different graphing module.

- Screen with list of characters in each issue

  Use Sqlite3 query to select and then output each character that appears in a selected issue. On the Common Appearance screen, user will be able to click on the issue to open new window with this information.
- Overall network statistics and rankings

  On the Start Screen, users will be given the option to look at a summary of the overall network information. This can include the average, minimum, and maximum of number of degrees, closeness centrality, and betweenness centrality. It can also include the top five characters in each category.
- Scrap websites only once and insert into table to lower loading times
- Add network statistics columns to Character table to cut down on loading times

Bugs

- Social network statistics don't appear
  1. J. JONAH JAMESON
  2. SHARON CARTER
  3. GWEN STACY
- IndexError: list out of range
  1. MORLUN
  2. COLOSSUS
  3. SILVER FOX
- HTTPError: HTTP Error 404: Not found
  1. KAREN PAGE
  2. ROUGHHOUSE
- Some "Top 5 Encounters" pairs show 0 common appearances.

Lessons Learned

This is the largest programming project I have worked on and so the most vital lesson was how to tackle such an involved job. I broke down the project into smaller tasks and tried to stick to a deadline where I would have at least one task done each week.

While I had used SQL to work with data before in my databases class, this was my first time working with such a large data set and with multiple, clear goals in mind. I learned how to clean, filter, match, and manipulate the data for my own purposes – beyond what the data was originally used for.

I worked on websites while I was younger, but I had not written any web scrappers before or worked with APIs. It was nice to be able to combine the programming I'm learning now with my past knowledge of markup languages.

After this project, I will also take care to use classes from the start. Most of my programming courses have covered classes very quickly towards the end. Thus, I'm used to writing code without classes first and then implementing classes later on. It would be far more efficient to just use an object-oriented approach from the start.