In this assignment, you’ll build and visualize a social network of movie actors: the graph nodes will represent individual actors, and two actors will be connected by an edge in the graph if they co-starred at least one movie together from an [IMDB top 100 movies (Links to an external site.)](http://www.imdb.com/chart/top) list - with no edge between them otherwise.  To do this, you’ll apply your new knowledge of HTML parsing with BeautifulSoup and JSON processing, and get experience with a widely-used graph visualization package, GraphViz.

First, you’ll need to install GraphViz  ([http://www.graphviz.org (Links to an external site.)](http://www.graphviz.org/" \t "_blank)) to get the gvedit application that draws the actor social network (parts of pydot also use graphviz).

You’ll be using these modules for this homework:  urllib2, BeautifulSoup, re, json, pydot, itertools, time.  Instructions for installing BeautifulSoup and pydot are in Lecture 3 slides.

We covered these modules and GraphViz in class (except for the time package, which is mentioned below).  Usingitertools isn’t required but it makes generating the graph easier.

**Step 1. (10 points)**

Fetch the IMDB top 100 movies (by number of votes) page using this URL:

[http://www.imdb.com/search/title?at=0&sort=num\_votes&count=100 (Links to an external site.)](http://www.imdb.com/search/title?at=0&sort=num_votes&count=100)

and save it in a HTML file named step1.html. The saved HTML file should look similar to step1\_desired\_output.html.  Note that a few movies have titles or actors with, e.g. accented characters. **Make sure you use the utf8 encoding to write out the HTML to use Unicode and preserve any non-English characters.**

**Step 2. (25 points)**

Parse the HTML page above with BeautifulSoup, extract movie information as described below, and save the result in a tab-delimited file named step2.txt.  Your step2.txt file should have 3 columns and 100 rows. The 3 columns should be:

IMDB\_ID

Rank

Title

The IMDB\_ID is the part that sits between last two slashes in the movie URL in the table.

For example, if the URL is http://www.imdb.com/title/tt0111161/, the IMDB ID is tt0111161

Your tab-delimited step2.txt file should look like step2\_desired\_output.txt. Here’s a sample of the first four lines:

tt0111161   1     The Shawshank Redemption (1994)

tt0468569   2     The Dark Knight (2008)

tt1375666   3     Inception (2010)

tt0110912   4     Pulp Fiction (1994)

**Step 3. (20 points)**

Use the Web service http://omdbapi.com/ to get movie metadata for each of the top 100 movies using the IMDB ID you collected in Step 2.  The API with sample requests is documented on the homepage.

For example, this URL fetches JSON for the movie “The Social Network”, which has IMDB ID tt1285016:[http://www.omdbapi.com/?i=tt1285016 (Links to an external site.)](http://www.omdbapi.com/?i=tt1285016)

You should see this JSON response:

{"Title":"The Social Network","Year":"2010","Rated":"PG-13","Released":"01 Oct 2010","Runtime":"120 min","Genre":"Biography, Drama","Director":"David Fincher","Writer":"Aaron Sorkin (screenplay), Ben Mezrich (book)","Actors":"Jesse Eisenberg, Rooney Mara, Bryan Barter, Dustin Fitzsimons","Plot":"Harvard student Mark Zuckerberg creates the social networking site that would become known as Facebook, but is later sued by two brothers who claimed he stole their idea, and the cofounder who was later squeezed out of the business.","Language":"English, French","Country":"USA","Awards":"Won 3 Oscars. Another 102 wins & 86 nominations.","Poster":"http://ia.media-imdb.com/images/M/MV5BMTM2ODk0NDAwMF5BMl5BanBnXkFtZTcwNTM1MDc2Mw@@.\_V1\_SX300.jpg","Metascore":"95","imdbRating":"7.8","imdbVotes":"326,376","imdbID":"tt1285016","Type":"movie","Response":"True"}

**IMPORTANT!  You MUST pause 5 seconds between EVERY HTTP request to omdbapi.com. If you don’t do this, and send requests omdbapi.com continuously in a loop with no delay, the server may reject your requests AND MAY EVEN SHUT DOWN.  (Yes, this has happened before.)**

**HINT:  Use the sleep(x) function in the time module after each HTTP request to pause x seconds.**

Save your results in a text file named step3.txt that contains a JSON string for each movie on each line. The file should look like step3\_desired\_output.txt

**Step 4. (20 points)**

After you verify that your step 3 output is correct, **you can comment out your URL fetching code for step 3 to avoid running that time-consuming step from now on**. Now open the file you saved in step 3, load the JSON string on each line into a variable, extract just the movie title and actors list, and save the results in a tab-delimited file named step4.txt.  There should be two columns in your tab-delimited step4.txt file:

1.     Movie name

2.     A JSON string containing the first 5 actors in the actors list.

The file should look like step4\_desired\_output.txt

**Step 5. (20 points)**

In this step you’ll generate the DOT file containing the actor graph, using the pydot module as described in class. After downloading and install GraphViz from [http://www.graphviz.org/ (Links to an external site.)](http://www.graphviz.org/), you should install the pydot package in your usual way, e.g.pip install pydot or sudo pip install pydot. Read and try out the examples at [http://pythonhaven.wordpress.com/tag/pydot/ (Links to an external site.)](http://pythonhaven.wordpress.com/tag/pydot/)

Now load the file you saved in step 4 and generate a graph using the actor lists. Each actor will be a graph node, and if two actors are in the actors list (of the first five actors, that is) for the same movie, then there will be an edge between them in the graph. Save the resulting graph in a .dot file, which is a plain text file in the DOT language.  Note that we don’t want to save a PNG file: we want the DOT file instead. The pydot manual at [http://code.google.com/p/pydot/downloads/list (Links to an external site.)](http://code.google.com/p/pydot/downloads/list) explains how to do this.

Save your .dot file to a file called actors\_graph\_output.dot.  It should look like the fileactors\_graph\_desired\_output.dot supplied in the homework ZIP file.

HINT:  Suppose A, B, C, D, E is the actors list for a movie. You’ll need to add edges for

every possible pairs of actors in this list, e.g. (A, B)  (A, C) etc. This is where the optional itertools module I described in class will come in useful, if you choose to use it.

**Step 6. (5 points)**

Open the saved actors\_graph\_output.dot file you created in step 5 using the gvedit application that comes with GraphViz. Using the Graph/Settings dialog, save the graph visualization in a PNG image file named actors\_graph.png, which should look like actors\_graph\_desired\_output.png provided in the homework ZIP file.

**What to submit**

As usual put all of your code in a single file named si-601-hw3\_**youruniquename**.py. Submit a zip file named si-601-hw3\_**youruniquename**.zip containing your Python source code file and all output files from the 6 steps above, including the final PNG image file of the graph.