Introduction to Web Science

Assignment 9

Prof. Dr. Steffen Staab
staab@uni-koblenz.de

René Pickhardt
rpickhardt@uni-koblenz.de

Korok Sengupta
koroksengupta@uni-koblenz.de

Olga Zagovora
zagovora@uni-koblenz.de

Institute of Web Science and Technologies
Department of Computer Science
University of Koblenz-Landau

Submission until: January 18, 2016, 10:00 a.m.
Tutorial on: January 20, 2016, 12:00 p.m.

For all the assignment questions that require you to write scripts, make sure to **include the scripts in the answer sheet, along with a separate python file.** Where screen shots are required, please add them in the answers directly and not as separate files.

Team Name: Golf

Members: Atique Baig, Mtarji Adam, Deepak Garg

1 Generative models (abstract) (10 points)

In the lecture sessions you will learn about 6 potential parts you could find in research paper abstracts. Consider the following research paper abstract[1]

> Hit songs, books, and movies are many times more successful than average, suggesting that "the best" alternatives are qualitatively different from "the rest"; yet experts routinely fail to predict which products will succeed. We investigated this paradox experimentally, by creating an artificial "music market" in which 14,341 participants downloaded previously unknown songs either with or without knowledge of previous participants' choices. Increasing the strength of social influence increased both inequality and unpredictability of success. Success was also only partly determined by quality: The best songs rarely did poorly, and the worst rarely did well, but any other result was possible.

1. Name the 6 potential parts you could find in research paper abstracts.
   a) State the background and problem tackled with the research.
   b) Name the methodology used.
   c) Formulate 1 to 3 precise research question that are answered in the paper.
   d) Talk about the unique solution or idea.
   e) Demonstrate the results.
   f) Conclude with a point of impact.
2. Mark all parts you can find in the given abstract.
   a) **State the background and problem tackled with the research.**
   " Hit songs, books, and movies are many times more successful than average, suggesting that "the best" alternatives are qualitatively different from "the rest"; yet experts routinely fail to predict which products will succeed."
   b) **Name the methodology used.**
   "We investigated this paradox experimentally, by creating an artificial 'music market' "
   c) **Talk about the unique solution or idea.**
   "in which 14,341 participants downloaded previously unknown songs either with or without knowledge of previous participants' choices."
   d) **Demonstrate the results.**
   "Increasing the strength of social influence increased both inequality and unpredictability of success. Success was also only partly determined by quality:

[1] https://www.princeton.edu/~mjs3/salganik_dodds_watts06_full.pdf

The best songs rarely did poorly, and the worst rarely did well, but any other result was possible."

2 Meme spreading model (10 points)

We provide you with the following excerpt from the meme paper[2] which will be discussed at the lecture. This part of the paper contains and explanation of their basic model. Your task is to **list five model choices** that stay in conflict with reality and **discuss the conflict**.

> Our basic model assumes a frozen network of agents. An agent maintains a time-ordered list of posts, each about a specific meme. Multiple posts may be about the same meme. Users pay attention to these memes only. Asynchronously and with uniform probability, each agent can generate a post about a new meme or forward some of the posts from the list, transmitting the cor- responding memes to neighboring agents. Neighbors in turn pay attention to a newly received meme by placing it at the top of their lists. To account for the empirical observation that past behavior affects what memes the user will spread in the future, we include a memory mechanism that allows agents to develop endogenous interests and focus. Finally, we model limited attention by allowing posts to survive in an agent's list or memory only for a finite amount of time. When a post is forgotten, its associated meme become less represented. A meme is forgotten when the last post carrying that meme disappears from the user's list or memory. Note that list and memory work like first-in-first-out rather than priority queues, as proposed in models of bursty human activity. In the context of single-agent behavior, our memory mechanism is reminiscent of the classic Yule-Simon model.
>
> The retweet model we propose is illustrated in Fig. 5. Agents interact on a directed social network of friends/followers. Each user node is equipped with a screen where received memes are recorded, and a memory with records of posted memes. An edge from a friend to a follower indicates that the friend's memes can be read on the follower's screen (#x and #y in Fig. 5(a) appear on the screen in Fig. 5(b)). At each step, an agent is selected randomly to post memes to neighbors. The agent may post about a new meme with probability p_n (#z in Fig. 5(b)). The posted meme immediately appears at the top of the memory. Otherwise, the agent reads posts about existing memes from the screen. Each post may attract the user's attention with probability pr (the user pays attention to #x, #y in Fig. 5(c)). Then the agent either retweets the post (#x in Fig. 5(c)) with probability $1 - p_m$, or tweets about a meme chosen from memory (#v triggered by #y in Fig. 5(c)) with probability p_m. Any post in memory has equal opportunities to be selected, therefore memes that appear more frequently in memory are more likely to be propagated (the memory has two posts about #v in Fig. 5(d)). To model limited user attention, both screen and memory have a finite capacity, which is the time in which a post remains in an agent's screen or memory. For all agents, posts are removed

[2] http://www.nature.com/articles/srep00335

after one time unit, which simulates a unit of real time, corresponding to Nu steps where Nu is the number of agents. If people use the system once weekly on average, the time unit corresponds to a week.

1. **Users pay attention to these memes only.**
   A normal user behavior will span over all kinds of posts other then memes, which means that in reality, less attention is given to memes then what the model simulates.

2. **Each agent can generate a post about a new meme or forward some of the posts from the list, transmitting the cor- responding memes to neighboring agents.**
   The agent does not account for the probability that a user sees the meme but does not react (no forwarding) and does not produce any new memes.

3. **That list and memory work like first-in-first-out rather than priority queues.**
   The human memory is more reliant on similarities and hooks, meaning that memories stored together are more likely to be remembered if at least one is recalled (hook) and that memories that are similar are more likely to be remembered subsequently, a first-in-first-out approach to mimic that memory might produce a much different behavior then reality.

4. **Any post in memory has equal opportunities to be selected.**
   In reality, the users choice is much more personal, some users might only gravitate to posts that have topics (or tags) that they relate to. There may be a number of other reasons to pay attention to a certain post other then the frequency, and in most cases, the background of each user should be taken into consideration before calculating the probability.

5. **For all agents, posts are removed after one time unit, which simulates a unit of real time, corresponding to Nu steps where Nu is the number of agents. If people use the system once weekly on average, the time unit corresponds to a week.**
   This means that in this model, we cannot see a meme been reborn if enough time passes, while in reality this happened many times due to all kinds of factors (major events, getting attention from a new generation of users, which in this case is not simulated since its a frozen network, etc). Having the meme been able to disappear without possible way to reaper other then been a new meme (which will reset all previous data) means that this special case, while it can be rare, is not taken into account.

3 Graph and its properties (10 points)

Last week we provided you with a graph of out-links[3] of Simple English Wikipedia which should be reused this week.

Write a function that returns the diameter of the given directed network. The diameter of a graph is the longest shortest path in the graph.

3.1 Hints

1. You can first write a function that returns the shortest path between nodes and then find the diameter.
2. Do not forget to use proper data structures to avoid a memory shortage.

Solution (pyhton code)
We could not solve the memory issue in time, the code still works on smaller graphs. Perhaps if we consider other algorithms like Dijkstra instead of getting all the possible paths and taking the length of the shortest path.

```
import pandas as pd

store = pd.HDFStore('store2.h5')
df2=store['df2']
# we transform the list of outlinks into a set
df2['out_links_unique']=df2.out_links.apply(lambda x:set(x))
graph=df2.set_index("name")["out_links_unique"].to_dict()

# Using breath first search algorithm to get all paths
def bfs_paths( start, goal):
    queue = [(start, [start])]
    while queue:
        (vertex, path) = queue.pop(0)
        try:
            for next in graph[vertex] - set(path):
                if next == goal:
                    yield path + [next]
                else:
                    queue.append((next, path + [next]))
        except KeyError:
            pass
# the shortest path is the first element returned by bfs_path
def shortest_path_len( start, goal):
```

[3] http://141.26.208.82/store.zip

```
    try:
        return len(next(bfs_paths( start, goal)))
    except StopIteration:
        return None

def find_diameter():
    # we get the vertices
    v = list(graph.keys())
    smallest_path_lens = []
    # looping on all possible pairs
    for i in range(len(v)-1) :
        for j in range(i+1, len(v)):
            # we store the lengths
            len_shortest_path =shortest_path_len(v[i],v[j])
            smallest_path_lens.append(len_shortest_path)
    # the maximum length of all shortest paths between every possible pair
    # of vertices is the diameter
    diameter =max(smallest_path_lens)
    return diameter

# We still have some memory issues, this is working on smaller graphs
# but we run into memory erros on the store.h5 df2 graph
print(find_diameter())
```

Important Notes

Submission

- Solutions have to be checked into the github repository. Use the directory name `groupname/assignment9/` in your group's repository.
- The name of the group and the names of all participating students must be listed on each submission.
- Solution format: all solutions as *one* PDF document. Programming code has to be submitted as Python code to the github repository. Upload *all* `.py` files of your program! Use `UTF-8` as the file encoding. *Other encodings will not be taken into account!*
- Check that your code compiles without errors.
- Make sure your code is formatted to be easy to read.
  - Make sure you code has consistent indentation.
  - Make sure you comment and document your code adequately in English.
  - Choose consistent and intuitive names for your identifiers.
- Do *not* use any accents, spaces or special characters in your filenames.

Acknowledgment

This latex template was created by Lukas Schmelzeisen for the tutorials of "Web Information Retrieval".

LaTeX

Currently the code can only be build using LuaLaTeX, so make sure you have that installed. If on Overleaf, there's an error, go to settings and change the LaTeXengine to `LuaLaTeX`.