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.

Group Name : Yankee
Candidates :

1. Sabin Bhattarai - 216203590
   sbhattarai@uni-koblenz.de
2. Biplov K.C. - 216203865
   biplov@uni-koblenz.de
3. Syed Salman Ali. - 216203923
   salmanali@uni-koblenz.de

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.

Answer

The 6 potential parts we could find in research paper abstracts are as follows:

- Background and problem that is to be tackled.
- Methodology that is used.
- 1 to 3 precise precise research question that is answered in the paper.
- Description of unique solution or idea from the author.
- Demonstration of results
- Conclusion: Likely with point of impact.

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

2. Mark all parts you can find in the given abstract.

Answer

• Background and problem : 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.

• Methodology used : We investigated this paradox experimentally, by creating an artificial ”music market”

• Research question :

• Description of solution or idea : 14,341 participants downloaded previously unknown songs either with or without knowledge of previous participants’ choices.

• Results : Increasing the strength of social influence increased both inequality and unpredictability of success.

• Conclusion : 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

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.

Answer

Five model choices that stay in conflict with reality are as follows:

- Each meme are considered as hashtag, which is not true in reality
- Memes cannot be posted again once it disappears from memory. But it reality memes can appear again depending upon context of post.
- Assumption of frozen network of agents. But this might not be the case in realty as agents may not be in frozen networks.
- User only pay attention to memes that has been posted multiple times. In reality user pays attention to the memes which he finds useful. Saying user 'only' pays attention to memes that has been posted multiple times ignores the fact user might pay attention to other memes.
- Memes that appear frequently in memory are more likely to be propagated. Actually, propagation of meme can depend upon number of other factors. For example, number of followers, etc.

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.

```
1:
2: import pandas as pd
3:
4: # we get all paths from a given start node to the end node , it returns
5: # the lists of path
6: def get_all_path_from_start_end(start_node, end_node, graph_dict , path=[]):
7:     path = path + [start_node]
8:     try:
9:         if start_node == end_node:
10:             return [path]
11:     except:
12:         return [path]
13:     if start_node not in graph_dict:
14:         return []
15:
16:     paths = []
17:     for outlink in graph_dict[start_node]:
18:         if outlink not in path:
19:             related_paths = get_all_path_from_start_end(outlink,
20:                                                         end_node, graph_dict,
21:                                                         path)
22:             for p in related_paths:
23:                 paths.append(p)
24:     return paths
25:
26: # calculates the diameter of a graph i.e. the longest shortest path in the graph.
27: def diameter_for_our_graph(graph_dict):
28:     v = list(graph_dict.keys())
29:     pairs = {i+j :(v[i],v[j]) for i in range(len(v)-1) \
30:                                                 for j in range(i+1, len(v))}
```

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

```
31:
32:     smallest_paths = []
33:
34:     # This part is just to check for one pair i.e. for start and end vertex pair.
35:     #singleExtractdict = {}
36:     #key , value = pairs.popitem()
37:     #singleExtractdict[key] = value
38:     # Testing
39:
40:     for (start,end) in pairs.values(): #to be changed to pairs for singleExtractdi
41:         paths = get_all_path_from_start_end(start,end , graph_dict)
42:
43:         if (len(paths) > 0):
44:             smallest = sorted(paths, key=len)[0]
45:             smallest_paths.append(smallest)
46:
47:     if len(smallest_paths) > 0:
48:         smallest_paths.sort(key=len)
49:         diameter = len(smallest_paths[-1]) - 1
50:         return diameter
51:     return 0
52:
53: # Helper function that changes particular list into numeric starting from 1.
54: def _changeToNumbers(toChangeListToNumber):
55:         changedList = []
56:         for value in toChangeListToNumber[0]:
57:             if value in newG:
58:                 changedList.append(newG[value])
59:         return changedList
60:
61: if __name__ == "__main__":
62:     store = pd.HDFStore("store.h5")#read .h5 file
63:     df2=store['df2']
64:
65:     # Dictionary of article names and its associated article text in list form
66:     dict_df2 = df2.set_index('name').T.to_dict('list')
67:     articles_list_ofoutlinks = dict_df2
68:
69:     # Here we convert each unique string into unique numbers and create the
70:     # dictionary accordingly as numeric comaprisons are each computations.
71:     newG = {}
72:     for index, key in enumerate(articles_list_ofoutlinks):
73:         newG[key] = index
74:
75:     updatedGraph = {}
76:     for key, value in articles_list_ofoutlinks.items():
77:         newKey = newG[key]
78:         newValue = _changeToNumbers(value)
79:         updatedGraph[newKey] = newValue
```

```
80:
81:     print("The diameter of our graph is := " , \
82:                                             diameter_for_our_graph(updatedGraph))
```

```
The diameter of our graph is :=  9
```

Figure 1: Obtained result for our program above.

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`.