

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.

Delta Group Members

Oana Dumitrasc

odumitrasc@uni-koblenz.de

Alisa Becker

alisabecker@uni-koblenz.de

Omar Aly

oaly@uni-koblenz.de

1 Generative models (abstract) (10 points)

In the lecture you will learn about 6 potential parts you could find in research paper abstracts. Consider the following research paper abstract¹

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

Solution 1:

1. State the Background and Problem you tackle with your research.
2. Name the methodology you have used.
3. Formulate 1 to 3 precise research question that are answered in your paper.
4. Talk about your unique solution or idea.
5. Demonstrate the results.
6. Conclude with a point of impact.

Solution 2:

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.
2. We investigated this paradox experimentally
3. Is Increasing the social influence affect the product success?
Is the product quality affect its success?
4. Creating an artificial “music market” in which 14,341 participants downloaded previously unknown songs either with or without knowledge of previous participants’ choices.

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

5. Increasing the strength of social influence increased both inequality and unpredictability of success.
Success was also only partly determined by quality.
6. The best songs rarely did poorly, and the worst rarely did well.

2 Meme spreading model (10 points)

We provide you with the following excerpt from the meme paper² which will be discussed at the lecture. This part of the paper contains an 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 corresponding 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 becomes 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 p_r (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.*

In the real world that is less likely. Users are paying attention to all sorts of memes coming from different sources. Maybe paying attention is not the right phrasing, what is paying attention? If some users get in contact with a meme, is that paying attention to it, or does it mean nothing if the user just sees it, but decides to ignore it.

2. *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 corresponding memes to neighboring agents..*

This modeling choice also does not coincide with reality, the uniform probability is just a simplification, as in the real world some people re-tweet more and more often than others.

3. *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.*

In the real world we consider that memory is more likely based on priority than on first in first out pattern. One cannot assume that something is simply forgotten after the last post carrying that information disappears. Human brain is much more complex than that, memories can be revived based on all kinds of sensors. If at some point one user sees a similar meme his mind will immediately connect those two together.

4. *Any post in memory has equal opportunities to be selected, therefore memes that appear more frequently in memory are more likely to be propagated.*

Even though it is a good simulation of how popular memes have a higher priority of being re-tweeted, in the real world people don't store the same piece of information twice in their memory. It is true that if we see something multiple times it is very likely that we will remember that thing better than something which we only saw once, but the way we are prioritizing is also governed by the human factor.

5. *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..*

It is true that the attention of users is limited, but has attention a direct relation to the finite capacity of memory. We can agree that the screen memory can be limited to something finite, but a person's memory is not something that can easily be

modeled like that. The point is that a person's memories cannot simply disappear after a fixed amount of time.

3 Graph and its properties (10 points)

Last week we provided you with a graph of out-links³ 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.

Answer:

The Diameter of the Graph is: 9 ... as shown in figure 1

```
1: import pandas as pd
2: import time
3: import networkx as nx
4:
5: store = pd.HDFStore('store2.h5')
6: df2 = store['df2']
7: store.close()
8:
9: numOfVertcies = len(df2['out_links'])
10:
11: graph = dict()
12:
13: for i in range(numOfVertcies):
14:     graph[df2['name'][i]] = df2['out_links'][i]
15:
16: def diameter(graphs):
17:     diameters = list()
18:     for graph in graphs:
19:         diameters.append(nx.diameter(graph))
20:     return max(diameters)
21:
22: G = nx.Graph(graph)
23:
24: start = time.time()
25: connectedComponentsGraphs = list(nx.connected_component_subgraphs(G))
26: print("Diameter of the Graph is: " + str(diameter(connectedComponentsGraphs)))
27: end = time.time()
```

³<http://141.26.208.82/store.zip>

```
28:  
29: totalTime = end - start  
30:  
31: print(totalTime)
```

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".

LA_TE_X

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 **L**A_TE_Xengine to **LuaLaTeX**.