# Introduction to Web Science

## Assignment 9

Prof. Dr. Steffen Staab            René Pickhardt

staab@uni-koblenz.de            rpickhardt@uni-koblenz.de

Korok Sengupta            Olga Zagovora

koroksengupta@uni-koblenz.de            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: Oscar

# 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.

   The 6 potential parts that can be found in a research paper abstract are:

   a) State the Background and Problem you tackle with your research.

   b) Name the methodology you have used.

   c) Formulate 1 to 3 precise research question that are answered in your paper.

   d) Talk about your 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.

- **Background** - Hit songs, books, and movies are many times more successful than average

- **Problem** - yet experts routinely fail to predict which products will succeed.

- **Methodology** - We investigated this paradox experimentally

- **Research questions** - suggesting that "the best" alternatives are qualitatively different from "the rest"

- **Unique solution/idea** - by creating an artificial "music market" in which 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. Success was also only partly determined by quality: The

---

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

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

- **Point of impact** - ?

# 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 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 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.

Model choices that are in conflict with reality are:

1. "Users pay attention to these memes only." - This assumption is in conflict with reality because people nowadays can get information from different sources like: social networks, news site, news papers, blogs, etc. The limitation that the users pay attention only to the memes of the model's network may alter the behavior of the users.

2. FIFO queue - In reality the memory doesn't work as a first-in-first-out rule, but more like a priority queue. People give more importance to special memories and pay less attention to other information or memories. In reality the first memory out should be the one that is less significant for the person.

3. Time unit to clear memory - People's memory doesn't work the same. The fact that the model has the same time unit for each of the users is a conflict with reality. Different people have different "types" of memory and also different capability to pay attention to information.

4. The attention of people in reality is not limited only by time but also from the number of other people that he/she knows and the amount of information exchange that he/she gets from these interactions.

5. In the model agents are supposed have the same characteristics and do the same operations. In reality people are not the same and based on their popularity or profession they have different influence on people. So a post tweeted by a celebrity or person with high professional reputation is more probable to become "famous" than a post from an ordinary person.The number of followers of this person is not the only factor that plays role in the spread of memes.

# 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: import re
4: #function to deal with the given data,
5: #deleting unexpected symbols in article's outlinks
6: def deal(row):
7:     r=set()
8:     for x in row:
9:         x=re.findall(r'[0-9a-zA-Z]+', x)
10:        x=''.join(map(str, x))
11:        r.add(x)
12:    return r
13:
14: store = pd.HDFStore('store.h5')
15: df2=store['df2']
16: df2.name=df2.name.apply(lambda x: re.findall(r'[0-9a-zA-Z]+', x))
17: df2.name=df2.name.apply(lambda x: ''.join(map(str, x)))
18: df2["out_links"]=df2["out_links"].apply(deal)
19: df2["out_links"]=df2["out_links"].apply(set)
20: df2.set_index('name')['out_links'].to_dict()
21: my_graph= dict(zip(df2.name, df2.out_links))
22:
23: def bfs_shortest_path(graph, start_node, end_node):
24:     queue = [[start_node]]
25:     visited = set()
26:     while queue:
27:         #first path in the queue
28:         path = queue.pop(0)
29:         #last node in the path
30:         vertex = path[-1]
31:         #Checks if we got to the end
```

---

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

```
32:             if vertex == end_node:
33:                 return len(path)-1
34:             elif vertex not in visited:
35:                 #enumerate all adjacent nodes, construct a new path
36:                 #and push it into the queue
37:                 for current_neighbour in graph.get(vertex, []):
38:                     new_path = list(path)
39:                     new_path.append(current_neighbour)
40:                     queue.append(new_path)
41:                 #Mark the vertex as visited
42:                 visited.add(vertex)
43:
44: def diameter(graph):
45:     #shortest path for each note to another
46:     shortest_path_node=[]
47:     #eccentricity of each node,which is the max of shortest paths
48:     eccentricity_of_node =[]
49:     for key in my_graph.keys():
50:         shortest_path_node=[bfs_shortest_path(my_graph, key, key1)
51:                             for key1 in my_graph.keys()]
52:         ecc=max(shortest_path_node)
53:         eccentricity_of_node.append(ecc)
54:      #diameter is the max of eccentricity of each node
55:     diam = max(eccentricity_of_node)
56:     return diam
57:
58: #call the function 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`.