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: hotel

Andrea Mildes - mildes@uni-koblenz.de

Sebastian Blei - sblei@uni-koblenz.de

Johannes Kirchner - jkirchner@uni-koblenz.de

Abdul Afghan - abdul.afghan@outlook.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.
2. Mark all parts you can find in the given abstract.

Solution:

1. Name the 6 potential parts you could find in research paper abstracts:
   1. Background and problem that is tackled in this paper
   2. The methodology used
   3. Research questions that are answered in the paper
   4. A unique solution or idea
   5. Demonstration of the results
   6. A point of impact
2. Mark all parts that can be found in the given abstract:
   1. Background and problem that is tackled in this paper
      "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. The methodology used
      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.

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

3. Demonstration of the results
   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.

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.

Solution: Five model choices that stay in conflict with reality:

1. *"Our basic model assumes a frozen network of agents."*
   In reality, the network is not frozen but changes over time.
2. *"Users pay attention to these memes only":*
   In reality, users pay attention to numerous other things and not only to these memes.
3. *"Note that list and memory work like first-in-first-out rather than priority queues, as proposed in models of bursty human activity."*
   In reality, it is not always the case that we forget memes in chronological order. For how long we still remember memes depends on how funny or interesting the corresponding meme was.
4. *"At each step, an agent is selected randomly to post memes to neighbors."*
   This implicates, that all agents roughly post the same amount of memes. This does not apply to reality, science some people post way more than others.
5. *"Each post may attract the user's attention with probability pr (the user pays attention to #x, #y in Fig. 5(c))."*
   This does not correspond with reality, since some memes attract more attention by using very catchy colors or bold fonts etc., which is not represented in this model.

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:

```
# assignment 9
# Andrea Mildes - mildes@uni-koblenz.de
# Sebastian Blei - sblei@uni-koblenz.de
# Johannes Kirchner - jkirchner@uni-koblenz.de
# Abdul Afghan - abdul.afghan@outlook.de

import sys
import pandas as pd
import math
from queue import *

#this creates a dictionary with the article name as the node of
# the graph and its outgoing links as the connection between the nodes.
#Note: the connections between the nodes are one-directional,
# because an outgoing link may appear in one article
#but there may not be an outgoing link to the previous article.
#The nodes are the keys and the lists of connections are the values
# in our dictionary.

def create_graph():
    graph = {}
    if(len(df2.name) == len(df2.out_links)):

        for i in range(0,len(df2.name)):
            graph.update({df2.name[i]:df2.out_links[i]})
    else:
        graph = 'unequal length'

    return graph

```

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

```
31: # This is the actual calculation of the shortest path from a given
32: # article to every other article
33: # using all possible combinations of outgoing links
34: # it returns a dictionary with the name of the destination article
35: # and the length of its path
36:
37: def shortest_path(start, edges):
38:     q = Queue()
39:     distance = {}
40:     previous = {}
41:     visited = [start]
42:     distance[start] = 0
43:     q.put(start)
44:
45:     while (q.empty() is False):
46:         x = q.get()
47:         try:
48:             for edge in graph.get(x):
49:                 if (edge not in visited):
50:                     distance[edge] = distance[x] + 1
51:                     previous[edge] = x
52:                     q.put(edge)
53:                     visited.append(edge)
54:         except TypeError:
55:             pass
56:     return distance
57:
58: # This method gets the dictionary with all the shortest paths
59: #  as an argument and simply has to iterate over it
60: # in order to find the largest value. That largest value is
61: # the diameter of the graph
62:
63: def get_diameter(dict):
64:     start = ''
65:     end = ''
66:     diameter = 0
67:     for (key, path) in dict.items():
68:         for(link, val) in path.items():
69:             if (val > diameter):
70:                 diameter = val
71:                 start = key
72:                 end = link
73:     return (diameter, start, end)
74: #This method simply stores every possible edge
75: # (starting and end point) in the whole graph into a single list
76: def get_edges(name):
77:     edges = []
78:     for edge in graph.get(name):
79:         edges.append((name,edge))
```

```
80:     return edges
81:
82: #Note: This program takes very long to terminate, since it has a
83: # complexity of O(V * (V + E)), where V is the number of Articles and E is
84: # the total number of outgoing links. The algorithm for calculating the
85: # shortest
86: #  path uses the Breadth-First Search Algorithm. It may take several
87: # hours to calculate the diameter.
88: # That being said, since the longest shortest path possible can only be as
89: # large as the amount of articles - 1, the diameter can not exceed that value
90: def main():
91:     global df1, df2, graph
92:     if(len(sys.argv) > 1):
93:         fileName = sys.argv[1]
94:     else:
95:         fileName = 'store2.h5'
96:     store = pd.HDFStore(fileName)
97:     df1 = store['df1']
98:     df2 = store['df2']
99:
100:     graph = create_graph()
101:
102:     edges = []
103:     distance = {}
104:     for name in df2.name:
105:         edges += get_edges(name)
106:     for name in df2.name:
107:         distance[name] = shortest_path(name, edges)
108:     (diam, start, end) = get_diameter(distance)
109:     print('The diameter is: \nFrom: ' + start + '\nTo: ' +
110:           end +'\nValue: ' + str(diam))
111:
112:
113:
114: if __name__ == '__main__':
115:     main()
```

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`.