Universität Hamburg
Department Informatik
Wissenschaftliches Rechnen
Dr. J. Kunkel, H. Hatzel

Exercise Sheet 8
Big Data Analytics
WiSe 2016/2017
Due Date: 2016-12-16, 390 P

Please document your code, without sufficent documentation you won't recive any points.

1 Cypher Query Language (60 P)

We have already imported a large dataset into Neo4j, you will all work on the same Neo4j instance. Write access is deactivated to avoid accidental deletions. You can access the Neo4j web interface on port 7475 on abu1. Forward the port to your local machine for easy access: `ssh -L 7475:abu1:7475 cluster`.
Write queries for the tasks below and execute them using the web interface.

- Print the names of all articles that link to the article *Artificial_intelligence*.
- Find all shortest paths that can lead you from the article *Alan_Turing* to the article *Gene_Amdahl* by following Wikipedia page-links.
- How many articles exist in the dataset?
- Which article is most strongly connected, as characterized by the sum of outgoing and incoming links.

Submission:

`1-queries.txt` Your cypher queries.

2 Cypher Data Model (60 P)

Extend the already provided data model from *Task 1*: 1) Add the article text and date of last change. 2) Add Wikipedia authors who can each have authored multiple articles and have a user name and email-address.
Describe the datamodel and provide some queries to insert example entities. Create a screenshot of your example graph. You may use `http://console.neo4j.org/` to create an example graph.

Submission:

`2-datamodel.txt` Your cypher queries for creating the data model.
`2-example.pdf` Example graph.

3 Clustering Wikipedia Articles (R) (150 P)

In this task we form clusters of Wikipedia articles based on relative word frequencies. Data is provided in: `/home/bigdata/8/enwiki-clean-10MiB.csv`.
Using the k-means algorithm test separating the dataset into multiple clusters, inspect your results. Are the resulting clusters useful, do their members share characteristics?
Use hierarchical clustering on the same data, for an example, see: `https://www.r-bloggers.com/hierarchical-clustering-in-r-2/`
Discuss your programs execution time, compare the run time of hierarchical clustering to k-means.

3.1 Hints

For testing purposes working with a small subset of the data can be useful.

3.1.1 Code Skeleton for R

```
library(tm)
d = read.csv("wikipedia-text-tiny-clean.csv", sep=";", quote="", header=F, stringsAsFactors=FALSE)
colnames(d) = c("title","content")

corpus = Corpus(VectorSource(d$content))

# We can set metadata, e.g. title:
i=0
corpus = tm_map(corpus, function(x) {
   i = i + 1
   meta(x, "title") = d$title[i]
   x
})

# you can do:
# inspect(corpus)

# To access an element in the corpus use:
# corpus[[X]]su

# we apply functions to clean the data
corpus = tm_map(corpus, removePunctuation)
corpus = tm_map(corpus, stripWhitespace)
corpus = tm_map(corpus, removeNumbers)
corpus = tm_map(corpus, content_transformer(tolower))
corpus = tm_map(corpus, removeWords, stopwords("english"))
# Create a sparse matrix for each document the word frequency
dtm = DocumentTermMatrix(corpus)
# To see data: e.g. call inspect(dtm[1:5,1:50] )

# Now apply clustering algorithms

# Inspect results....
```

Submission:

`3-wikipedia-clusters.pdf` A lab notebook with your code, analysis and results.

4 HDFS REST API (Python) (120 P)

In this task, we explore the use of HDFS's REST API.
The specification of the API is available online: `http://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-hdfs/WebHDFS.html`.
For testing purpose, first, we use `curl` to perform the following operations:

- Listing of directory
- Creation of an file
- Fetching file content
- Deletion of a file
- Renaming a directory
- Deleting of a directory

Document your arguments to `curl` and the return values.
Now create a Python program which provides an FTP-alike interface to the user and performs the REST calls to WebHDFS, thus it allows to interactively browse, upload and download files from HDFFS. The operations offered by the Python program should be analogous to the FTP `https://www.cs.colostate.edu/helpdocs/ftp.html`. Implement at least the following (hadoop fs) commands: ls, put, get, mkdir, rmdir and rm [-r]. Also implement the local operations `ls` and `chdir`.

4.1 Hints

If necessary, you may use `http://httpbin.org/` to debug your HTTP requests. The authentication on our HDFS is disabled, thus, you have to provide the query argument `user.name` with your name for all modifying operations.

4.1.1 Code skeleton

Submission:

`4-curl.txt` Your documented `curl` calls with responses.
`4-hdfs-ftp.py` Your Python FTP-alike HDFS REST client.

5 Optional bonus task: Finding places near you (60 P)

For this task, you will use the Google geocoding API to enable fuzzy finding of places from the wikipedia near you.
The script should be invoked as follows: `./near.py Hamburg` and should in this case return a list of articles that are tagged as being located near Hamburg.
Take a look at the documentation: `https://developers.google.com/maps/documentation/geocoding/intro`.
Please consider that the API is subject to rate-limits, please don't access it repeatedly in an automated fashion.

Submission:

`5-near.py` Your script to find places close to a given location.