Report HW#2

Team Member

Pawel Urbanowicz	108015016
Martin Ledl	108012012

Github Repository

The Github repository for this project can be found here:
https://github.com/mledl/BDMA_HW/tree/master/HW2
Please check the repository for results under "/data/results/", because the limited upload size restricted us from uploading the result files to cyberclassroom.

Responsibilities

Pawel Urbanowicz

- Configure the Production Spark Environment and run the final solution for benchmark.
- Calculate the average popularity of each news by hour, and by day, respectively in social feedback data (2).
- Calculate the sum and average sentiment score of each topic, respectively in news data (3).
- Generate proper output for each of the solved tasks (2)(3)
- Contribute to report.

Martin Ledl

- Setup project to work on HW#2.
- Proper data preprocessing for textual data (remove duplicates, punctuation marks, stopword and dimension reduction)
- Count the words in two fields: 'Title' and 'Headline' respectively, and list the most frequent words according to the term frequency in descending order, in total, per day, and per topic, respectively in news data (1).
- From subtask (1), for the top-100 frequent words per topic in titles and headlines, calculate their co-occurrence matrices (100x100), respectively. Each entry in the matrix will contain the co-occurrence frequency in all news titles and headlines, respectively (4).
- Generate proper output for each of the solved tasks (1)(4)
- Contribute to report.

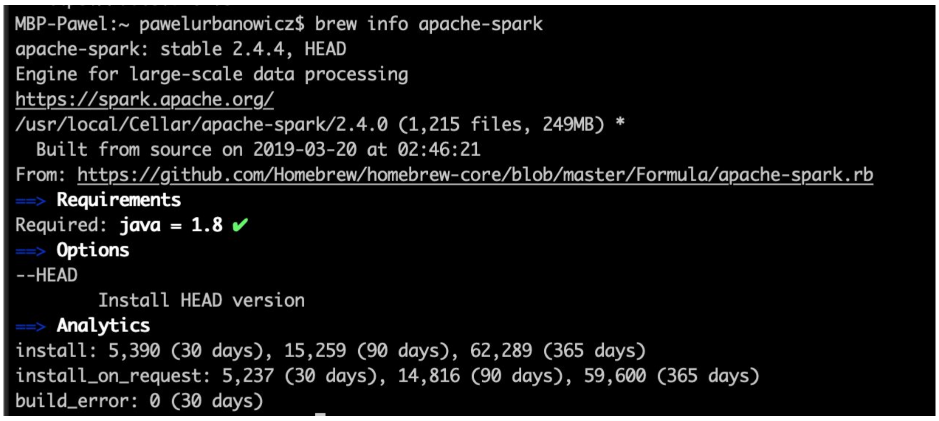
Environment Setup

For local development we tested our code on a locally installed spark instance and for target stage we used Docker technology to wrap spark master instance, 2 spark workers instances and our Python script into separate containers. Software/Frameworks in use:

- Python 3.7 to write our code
- Spark version 2.4.4
- Hadoop version 3.2.1
- Docker engine version 18.09.2.

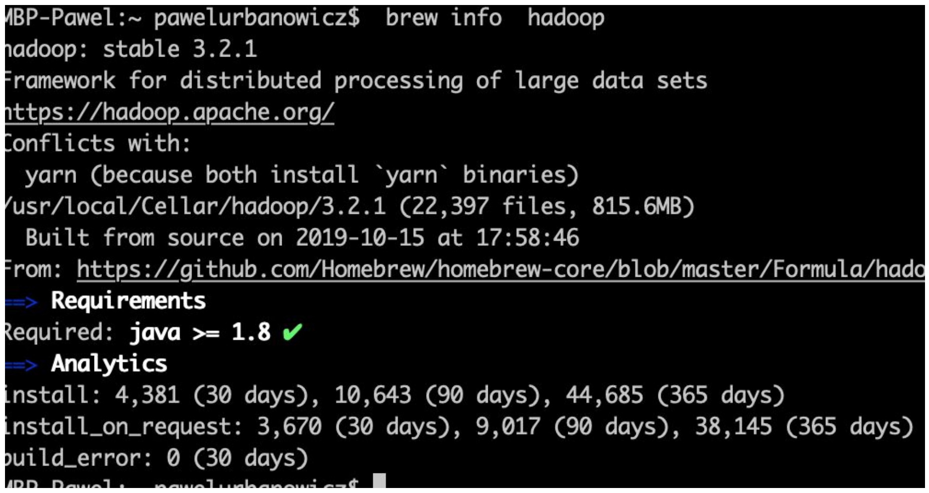
Environment setup for OSX

a. Install Spark
brew install apache-spark



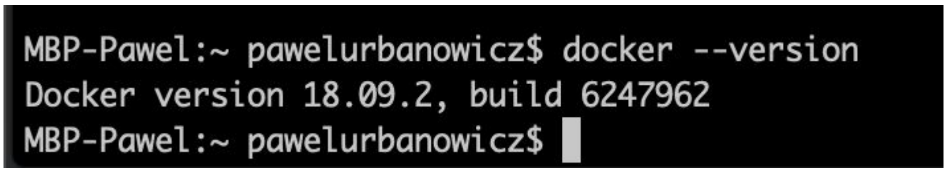
```
MBP-Pawel:~ pawelurbanowicz$ brew info apache-spark
apache-spark: stable 2.4.4, HEAD
Engine for large-scale data processing
https://spark.apache.org/
/usr/local/Cellar/apache-spark/2.4.0 (1,215 files, 249MB) *
  Built from source on 2019-03-20 at 02:46:21
From: https://github.com/Homebrew/homebrew-core/blob/master/Formula/apache-spark.rb
==> Requirements
Required: java = 1.8 ✔
==> Options
--HEAD
        Install HEAD version
==> Analytics
install: 5,390 (30 days), 15,259 (90 days), 62,289 (365 days)
install_on_request: 5,237 (30 days), 14,816 (90 days), 59,600 (365 days)
build_error: 0 (30 days)
```

b. Install Hadoop



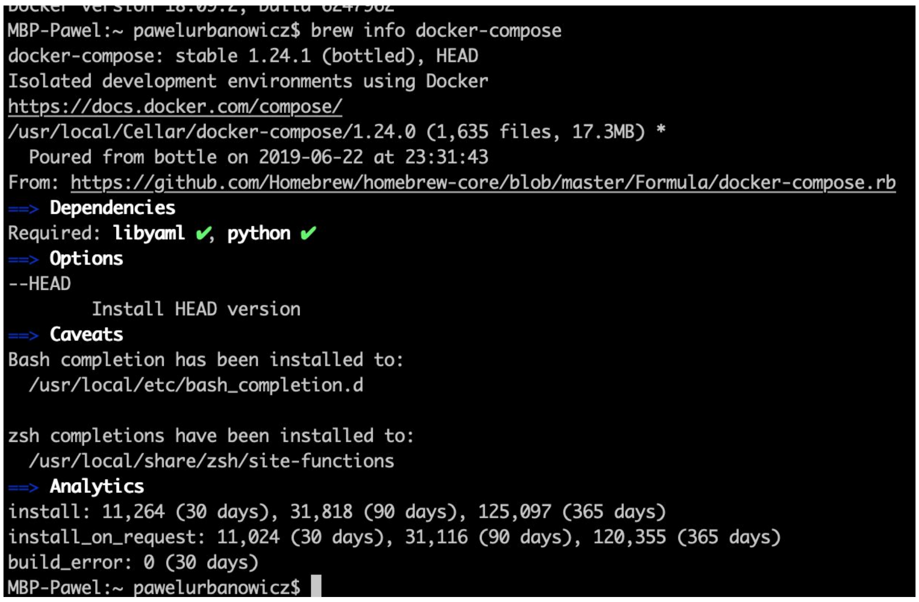
```
IBP-Pawel:~ pawelurbanowicz$  brew info  hadoop
nadoop: stable 3.2.1
Framework for distributed processing of large data sets
nttps://hadoop.apache.org/
Conflicts with:
  yarn (because both install `yarn` binaries)
/usr/local/Cellar/hadoop/3.2.1 (22,397 files, 815.6MB)
  Built from source on 2019-10-15 at 17:58:46
From: https://github.com/Homebrew/homebrew-core/blob/master/Formula/hado
==> Requirements
Required: java >= 1.8 ✔
==> Analytics
install: 4,381 (30 days), 10,643 (90 days), 44,685 (365 days)
install_on_request: 3,670 (30 days), 9,017 (90 days), 38,145 (365 days)
build_error: 0 (30 days)
```

c. Install Docker Desktop for Mac
https://docs.docker.com/docker-for-mac/install/



```
MBP-Pawel:~ pawelurbanowicz$ docker --version
Docker version 18.09.2, build 6247962
MBP-Pawel:~ pawelurbanowicz$
```

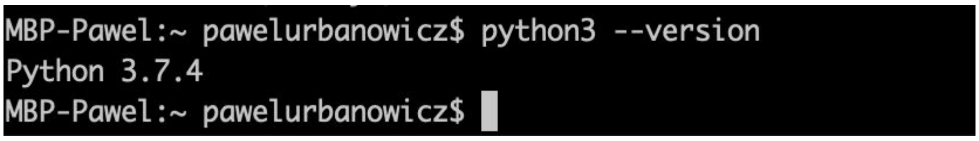
d. Install Docker Compose
brew install docker-compose



```
MBP-Pawel:~ pawelurbanowicz$ brew info docker-compose
docker-compose: stable 1.24.1 (bottled), HEAD
Isolated development environments using Docker
https://docs.docker.com/compose/
/usr/local/Cellar/docker-compose/1.24.0 (1,635 files, 17.3MB) *
  Poured from bottle on 2019-06-22 at 23:31:43
From: https://github.com/Homebrew/homebrew-core/blob/master/Formula/docker-compose.rb
==> Dependencies
Required: libyaml ✔, python ✔
==> Options
--HEAD
        Install HEAD version
==> Caveats
Bash completion has been installed to:
  /usr/local/etc/bash_completion.d

zsh completions have been installed to:
  /usr/local/share/zsh/site-functions
==> Analytics
install: 11,264 (30 days), 31,818 (90 days), 125,097 (365 days)
install_on_request: 11,024 (30 days), 31,116 (90 days), 120,355 (365 days)
build_error: 0 (30 days)
MBP-Pawel:~ pawelurbanowicz$
```

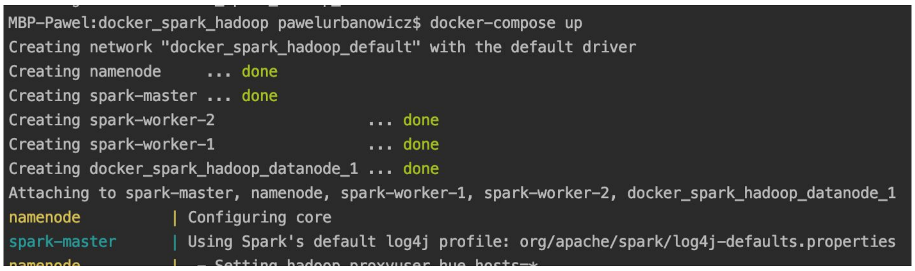
e. Install python 3
brew install python



```
MBP-Pawel:~ pawelurbanowicz$ python3 --version
Python 3.7.4
MBP-Pawel:~ pawelurbanowicz$
```

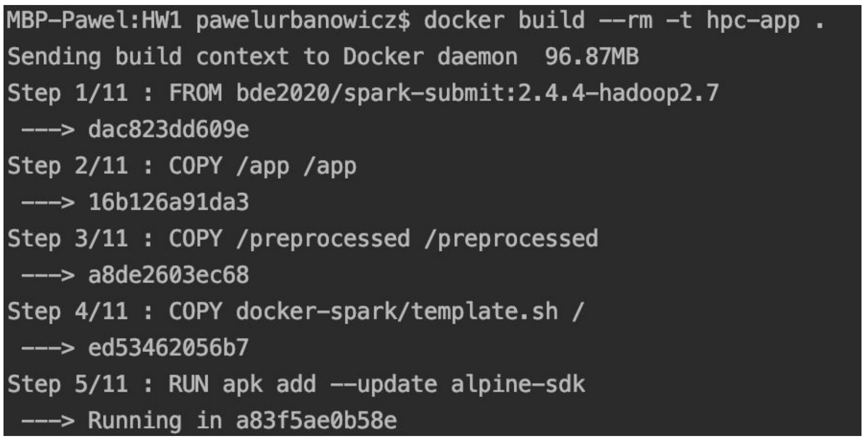
f. Clone code from repository
git clone https://github.com/mledl/BDMA_HW

g. Open terminal and go to docker_spark_hadoop directory and run:
docker-compose up



```
MBP-Pawel:docker_spark_hadoop pawelurbanowicz$ docker-compose up
Creating network "docker_spark_hadoop_default" with the default driver
Creating namenode     ... done
Creating spark-master ... done
Creating spark-worker-2                 ... done
Creating spark-worker-1                 ... done
Creating docker_spark_hadoop_datanode_1 ... done
Attaching to spark-master, namenode, spark-worker-1, spark-worker-2, docker_spark_hadoop_datanode_1
namenode          | Configuring core
spark-master      | Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
```

h. Go to HW2 directory to build image for python script docker build --rm -t app .

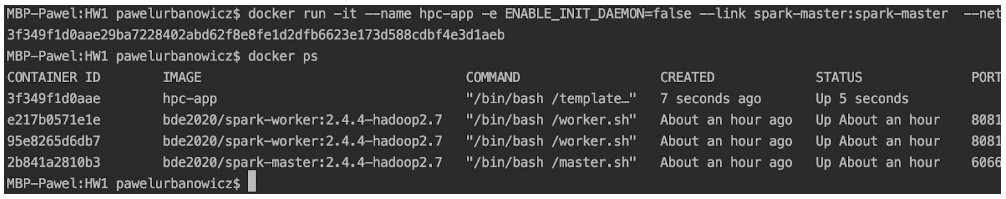


```
MBP-Pawel:HW1 pawelurbanowicz$ docker build --rm -t hpc-app .
Sending build context to Docker daemon  96.87MB
Step 1/11 : FROM bde2020/spark-submit:2.4.4-hadoop2.7
 ---> dac823dd609e
Step 2/11 : COPY /app /app
 ---> 16b126a91da3
Step 3/11 : COPY /preprocessed /preprocessed
 ---> a8de2603ec68
Step 4/11 : COPY docker-spark/template.sh /
 ---> ed53462056b7
Step 5/11 : RUN apk add --update alpine-sdk
 ---> Running in a83f5ae0b58e
```

It can take some time as some libraries must be built from sources

i. Add data to hadoop
docker cp data namenode:data
docker exec -it namenode bash
hadoop fs -put /data /data

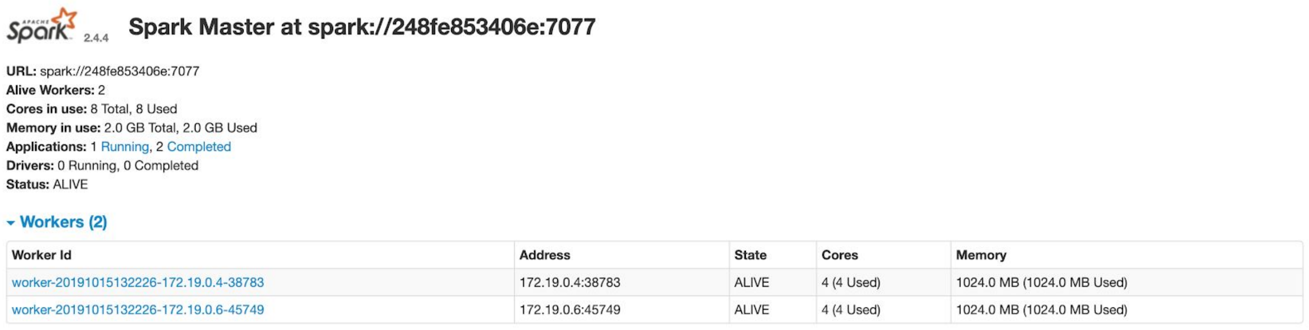
j. Run previously build image
docker run -it --name app -e ENABLE_INIT_DAEMON=false --link spark-master:spark-master --net docker_spark_hadoop_default -d app



```
MBP-Pawel:HW1 pawelurbanowicz$ docker run -it --name hpc-app -e ENABLE_INIT_DAEMON=false --link spark-master:spark-master  --net
3f349f1d0aae29ba7228402abd62f8e8fe1d2dfb6623e173d588cdbf4e3d1aeb
MBP-Pawel:HW1 pawelurbanowicz$ docker ps
CONTAINER ID        IMAGE                                  COMMAND                  CREATED             STATUS              PORT
3f349f1d0aae        hpc-app                                "/bin/bash /template…"   7 seconds ago       Up 5 seconds
e217b0571e1e        bde2020/spark-worker:2.4.4-hadoop2.7   "/bin/bash /worker.sh"   About an hour ago   Up About an hour    8081
95e8265d6db7        bde2020/spark-worker:2.4.4-hadoop2.7   "/bin/bash /worker.sh"   About an hour ago   Up About an hour    8081
2b841a2810b3        bde2020/spark-master:2.4.4-hadoop2.7   "/bin/bash /master.sh"   About an hour ago   Up About an hour    6066
MBP-Pawel:HW1 pawelurbanowicz$
```

Result of setup:

http://localhost:8089/



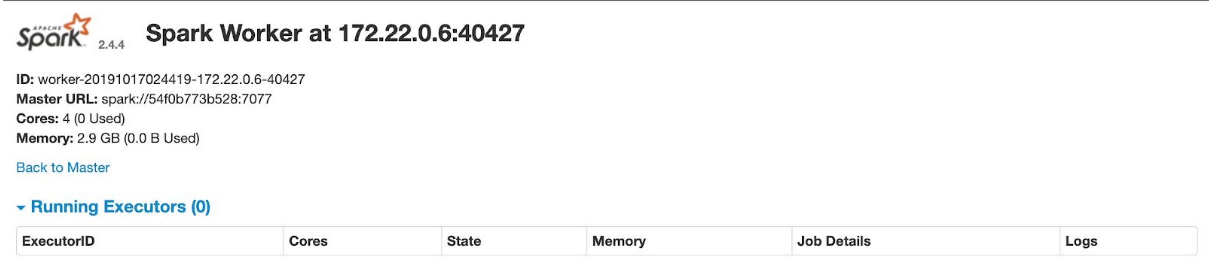
Spark Master at spark://248fe853406e:7077

URL: spark://248fe853406e:7077
Alive Workers: 2
Cores in use: 8 Total, 8 Used
Memory in use: 2.0 GB Total, 2.0 GB Used
Applications: 1 Running, 2 Completed
Drivers: 0 Running, 0 Completed
Status: ALIVE

Workers (2)

Worker Id	Address	State	Cores	Memory
worker-20191015132226-172.19.0.4-38783	172.19.0.4:38783	ALIVE	4 (4 Used)	1024.0 MB (1024.0 MB Used)
worker-20191015132226-172.19.0.6-45749	172.19.0.6:45749	ALIVE	4 (4 Used)	1024.0 MB (1024.0 MB Used)

http://localhost:8084/ and http://localhost:8085/



Spark Worker at 172.22.0.6:40427

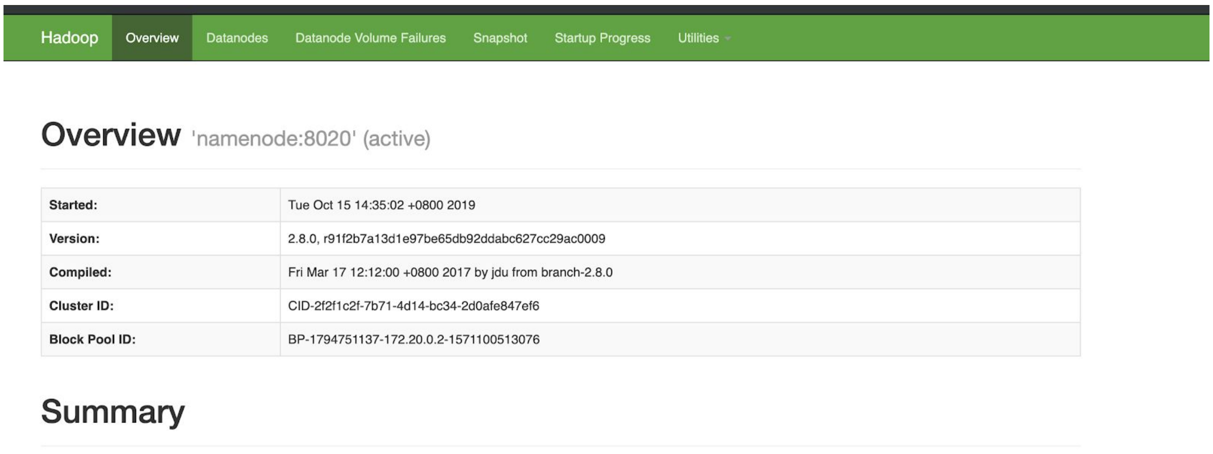
ID: worker-20191017024419-172.22.0.6-40427
Master URL: spark://54f0b773b528:7077
Cores: 4 (0 Used)
Memory: 2.9 GB (0.0 B Used)

Back to Master

Running Executors (0)

ExecutorID	Cores	State	Memory	Job Details	Logs

http://localhost:9870/

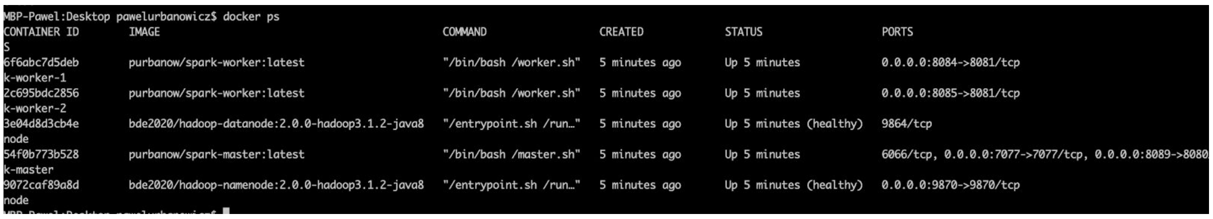


Overview 'namenode:8020' (active)

Started:	Tue Oct 15 14:35:02 +0800 2019
Version:	2.8.0, r91f2b7a13d1e97be65db92ddabc627cc29ac0009
Compiled:	Fri Mar 17 12:12:00 +0800 2017 by jdu from branch-2.8.0
Cluster ID:	CID-2f2f1c2f-7b71-4d14-bc34-2d0afe847ef6
Block Pool ID:	BP-1794751137-172.20.0.2-1571100513076

Summary

docker ps



```
MBP-Pawel:Desktop pawelurbanowicz$ docker ps
CONTAINER ID        IMAGE                                        COMMAND                  CREATED             STATUS                   PORTS
S
6f6abc7d5deb        purbanow/spark-worker:latest                 "/bin/bash /worker.sh"   5 minutes ago       Up 5 minutes             0.0.0.0:8084->8081/tcp
k-worker-1
2c695bdc2856        purbanow/spark-worker:latest                 "/bin/bash /worker.sh"   5 minutes ago       Up 5 minutes             0.0.0.0:8085->8081/tcp
k-worker-2
3e04d8d3cb4e        bde2020/hadoop-datanode:2.0.0-hadoop3.1.2-java8   "/entrypoint.sh /run…"   5 minutes ago       Up 5 minutes (healthy)   9864/tcp
node
54f0b773b528        purbanow/spark-master:latest                 "/bin/bash /master.sh"   5 minutes ago       Up 5 minutes             6066/tcp, 0.0.0.0:7077->7077/tcp, 0.0.0.0:8089->8080
k-master
9072caf89a8d        bde2020/hadoop-namenode:2.0.0-hadoop3.1.2-java8   "/entrypoint.sh /run…"   5 minutes ago       Up 5 minutes (healthy)   0.0.0.0:9870->9870/tcp
node
```

Data Preprocessing

A few different tasks needed to be done to preprocess the data, because we had to deal with textual data in this exercise.
First of all, duplicated entries needed to be removed from the raw dataset. Furthermore, the dimensions must be reduced to the task relevant ones.
As we are required to analyse word count and co-occurrence of words in the dataset, we need to remove all kind of punctuation marks from the textual data and transform the text into its lower case representation in order to match equal words (e.g. economy == Economy). The sentences must be tokenized into their words in the next step and trimmed in order to not start or end with whitespace characters.

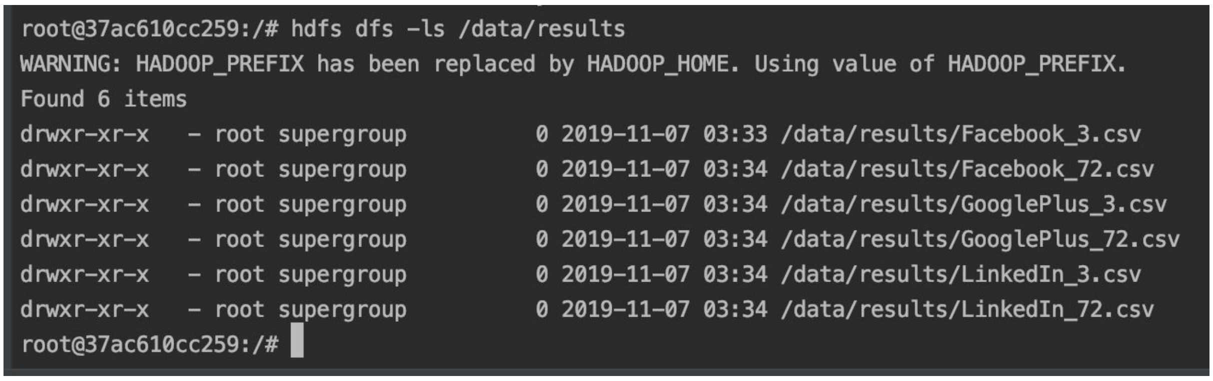
In the next step we use Sparks StopwordRemover in order to remove unimportant words from the lists of tokens (e.g. i, a, to, for, …). After preprocessing the data as described we can start implementing the word count and co-occurrence statistics as required.

Output Format

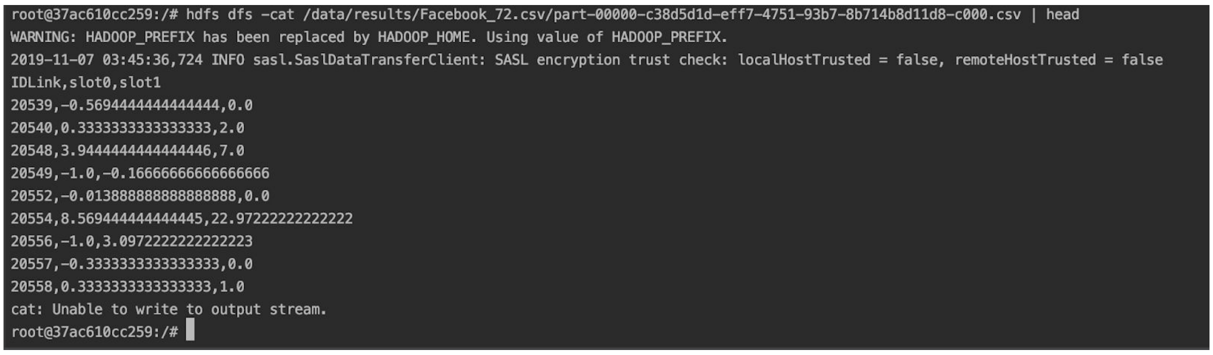
Task 1: 6 sorted lists of top-frequent words: {in total, per day, per topic}{for titles, headlines}
For this task we decided to write the results to files.
In Total: a list of decreasing sorted word, count can be found under "data/results/wordcount_{Title, Headline}_total/"

For the grouped data we decided to make a list of top-frequent words per day as well as per topic. Those lists are written to the following directories: "data/results/wordcount_{Title, Headline}_{PublishDate, Topic}/". Note that those files contain all resulting lists per PublishDate or Topic. The PublishDate or Topic is also written to them as column in order to distinguish between the different lists.

Task 2: 6 files: {by hour, by day} {3 platforms}

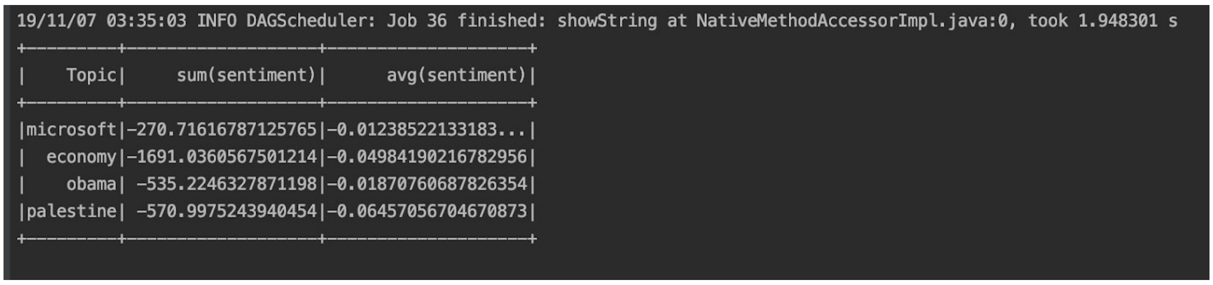


```
root@37ac610cc259:/# hdfs dfs -ls /data/results
WARNING: HADOOP_PREFIX has been replaced by HADOOP_HOME. Using value of HADOOP_PREFIX.
Found 6 items
drwxr-xr-x   - root supergroup          0 2019-11-07 03:33 /data/results/Facebook_3.csv
drwxr-xr-x   - root supergroup          0 2019-11-07 03:34 /data/results/Facebook_72.csv
drwxr-xr-x   - root supergroup          0 2019-11-07 03:34 /data/results/GooglePlus_3.csv
drwxr-xr-x   - root supergroup          0 2019-11-07 03:34 /data/results/GooglePlus_72.csv
drwxr-xr-x   - root supergroup          0 2019-11-07 03:34 /data/results/LinkedIn_3.csv
drwxr-xr-x   - root supergroup          0 2019-11-07 03:34 /data/results/LinkedIn_72.csv
root@37ac610cc259:/#
```



```
root@37ac610cc259:/# hdfs dfs -cat /data/results/Facebook_72.csv/part-00000-c38d5d1d-eff7-4751-93b7-8b714b8d11d8-c000.csv | head
WARNING: HADOOP_PREFIX has been replaced by HADOOP_HOME. Using value of HADOOP_PREFIX.
2019-11-07 03:45:36,724 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = false, remoteHostTrusted = false
IDLink,slot0,slot1
20539,-0.5694444444444444,0.0
20540,0.3333333333333333,2.0
20548,3.9444444444444446,7.0
20549,-1.0,-0.16666666666666666
20552,-0.013888888888888888,0.0
20554,8.569444444444445,22.97222222222222
20556,-1.0,3.0972222222222223
20557,-0.3333333333333333,0.0
20558,0.3333333333333333,1.0
cat: Unable to write to output stream.
root@37ac610cc259:/#
```

Task 3: 8 values: {sum, avg} {4 topics}



```
19/11/07 03:35:03 INFO DAGScheduler: Job 36 finished: showString at NativeMethodAccessorImpl.java:0, took 1.948301 s
+---------+------------------+--------------------+
|    Topic|     sum(sentiment)|      avg(sentiment)|
+---------+------------------+--------------------+
|microsoft|-270.71616787125765|-0.01238522133183...|
|  economy|-1691.0360567501214|-0.04984190216782956|
|    obama| -535.2246327871198|-0.01870760687826354|
|palestine| -570.9975243940454|-0.06457056704670873|
+---------+------------------+--------------------+
```

Task 4: 8 100x100 matrices: {title, headline} {4 topics}
The matrices are written to csv files and can be found under
“data/results/cooc_matrix_{Title,Headline}_{economy, obama, microsoft, palestine}/”