Report HW#4

Team Member

Pawel Urbanowicz 108015016
Martin Ledl 108012012
Tobias Kick 108998413

Github Repository

The Github repository for this project can be found here:
https://github.com/mledl/BDMA_HW/tree/master/HW4
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.
- Take care of task 3: list the top-'similar' movies based on the *cosine similarity* of previous ratings each movie received
- Contribute the solved parts to report.

Martin Ledl

- Setup project to work on HW#4.
- Take care of task 5: Write recommender system for using user- and item- based collaborative filtering. Used the similarity results from task 3 and 4.
- Adaption of task 2 and 3 in order to be used for recommendations.
- Contribute the solved parts to report.

Tobias Kick

- Take care of task 1: list the top-rated movies based on the 'average' rating score
- Take care of task 2: list the top-'similar' users based on the *cosine similarity* of previous ratings each user has given
- Contribute to report.

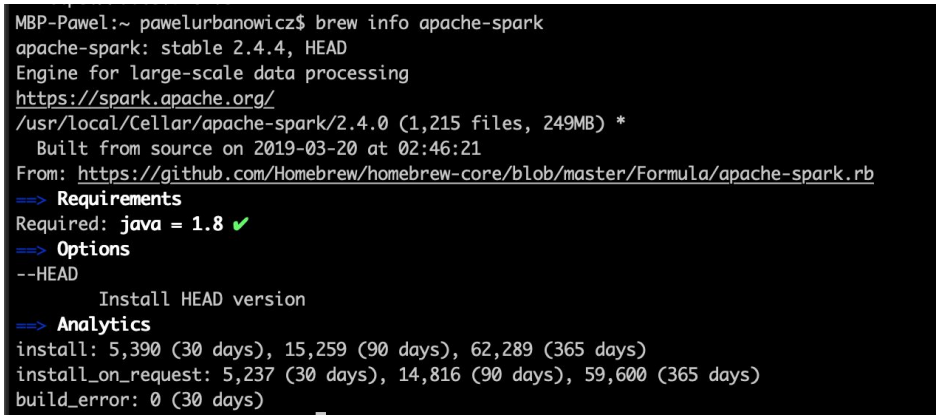
Environment Setup

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:

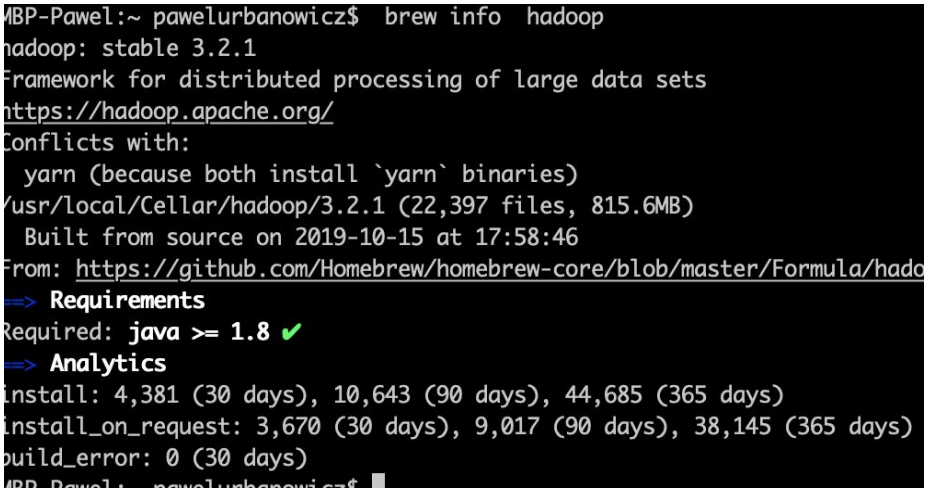
- Python 3.7 to write our code
- Spark version 2.4.4
- Hadoop version 3.2.1
- Docker engine version 18.09.2.

Environment setup for OSX

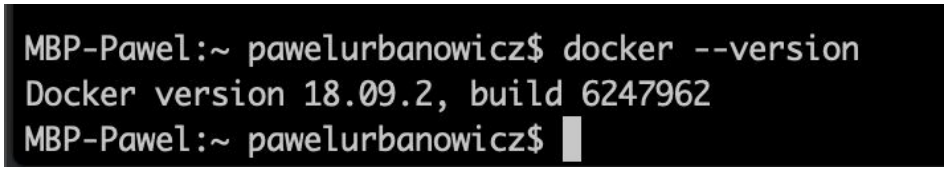
a. Install Spark
brew install apache-spark



b. Install Hadoop



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



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

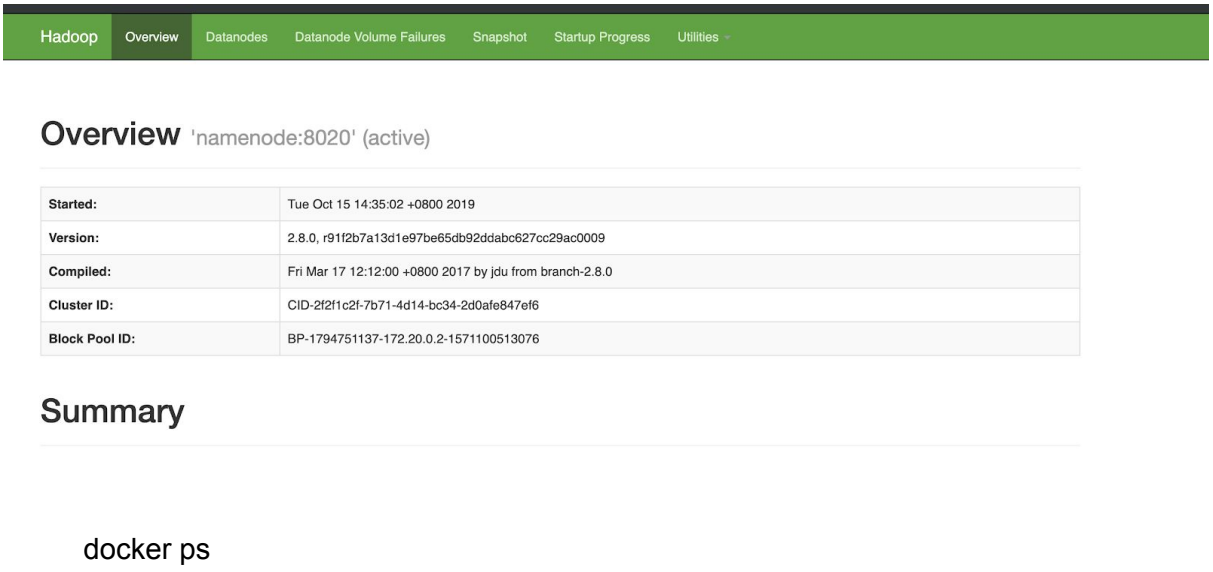
```
MBP-Pawel:HW1 pawelurbanowicz$ docker run -it --name hpc-app -e ENABLE_INIT_DAEMON=false --link spark-master:spark-master --net
3f349f1d0aae29ba7228402abd62f8e8fe1d2dfb6623e173d588cdbf4e3d1aeb
MBP-Pawel:HW1 pawelurbanowicz$ docker ps
CONTAINER ID        IMAGE                                COMMAND                  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/

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

http://localhost:9870/



docker ps

Data Preprocessing

The main preprocessing task for the given Movielens dataset was to reduce dimensions that are not needed for the calculations and recommendation tasks. To fulfill the given task we just needed the following features per dataset:

- ratings.dat: 'UserID', 'MovieID', 'Rating'
- movies.dat: 'MovieID', 'Title'
- The user.dat dataset is not needed at all

Task 1

For task 1 no further preprocessing tasks have to be done.

Task 2

To sufficiently calculate similar users for a given user, we need to established a normalized pivoted matrix where the rows represent the user and the columns represent the movies. One entry in this matrix is the rating of user u for item i. This pivoted table has to be normalized by the user mean (row mean) in order to be able to make the cosine similarity measure produce similarities between -1 and 1 (distributed around 0). Moreover, we establish a dataframe that holds a list of movie ratings per user, because it is easier to compute the cosine similarity in this format.

Task 3

To sufficiently calculate movie (item) similarities for a given user, we need to established a normalized pivoted matrix where the rows represent the movies and the columns represent the users. One entry in this matrix is the rating of user u for item i. This pivoted table has to be normalized by the user mean (row mean) in order to be able to make the cosine similarity measure produce similarities between -1 and 1 (distributed around 0). Moreover, we establish a dataframe that holds a list of user ratings for the given movie, because it is easier to compute the cosine similarity in this format.

Task 4 a+b:

No special preprocessing work has to be done as the recommendation is based on the similarities provided by tasks 2 and 3.

Output Format

Task 1

Printed a list of <movie, average rating score> pairs in descending order of the average rating to file '../data/results/task1/'. The results look like this for example (3 random tupel):

- 2503,4.666666666666667
- 2905,4.608695652173913
- 2019,4.560509554140127

The movie is described by its movieID and the average rating is used for sorting.

Task 2

Printed a list of <user, similarity score> pairs of similar users in descending order of the user similarity score to file '../data/results/task2/'. The results look like this for example (3 random tupel):

- 1337,0.18924150954139238
- 379,0.15989269391885147
- 5404,0.15515416058858933

The user is described by its userID and the similarity score is used for sorting.

Task 3

Printed a list of <movie, similarity score> pairs of similar movies in descending order of the user similarity score to file '../data/results/task3/'. The results look like this for example (3 random tupel):

- 3114,0.35929894435512616
- 588,0.2631298800458826
- 2355,0.24181875790214846

The movie is described by its movieID and the similarity score is used for sorting.
Note: the resulting lists are shorter because the item similarity for a specific user calculates the similarity of an item that is unrated for the given user with all items that have been rated by the given user.

Task 4 a:

Printed a list of <movie, predicted rating> pairs of similar movies in descending order of the predicted rating to file '../data/results/task4a/'. This prediction is based on the k similar users to a given user. Those k similar user have been obtained using task 2 implementation.

The movie is described by its title and the predicted rating is used for sorting.

Task 4b:

Printed a list of <movie, predicted rating> pairs of similar movies in descending order of the predicted rating to file '../data/results/task4b/'. This prediction is based on the k similar movies already rated by the given user to the unrated movies of the given user . Those k similar movies have been obtained using task 3 implementation for each unrated movie.The results look like this for example (3 random tupel):

- Waiting to Exhale (1995),4.402051386809718
- Grumpier Old Men (1995),4.3965653347667315
- Father of the Bride Part II (1995),4.364341325765423

The movie is described by its title and the predicted rating is used for sorting.