Project 1

Final Report

Big Data and Cloud Computing

Grupo:
Afonso Pinto - up201503316
Edgar Carneiro - up201503784

Faculdade de Ciências da Universidade do Porto
Departamento de Ciência de Computadores

April 7, 2019

1 Summary

In this project, we made use of Apache Spark to process MovieLens datasets. We wrote a number of Python functions that involve these datasets and the use of the TF-IDF and Jaccard index metrics. Beyond the requested functions, all the extra challenges were also completed (see end of notebook).

2 Technologies & Workflow

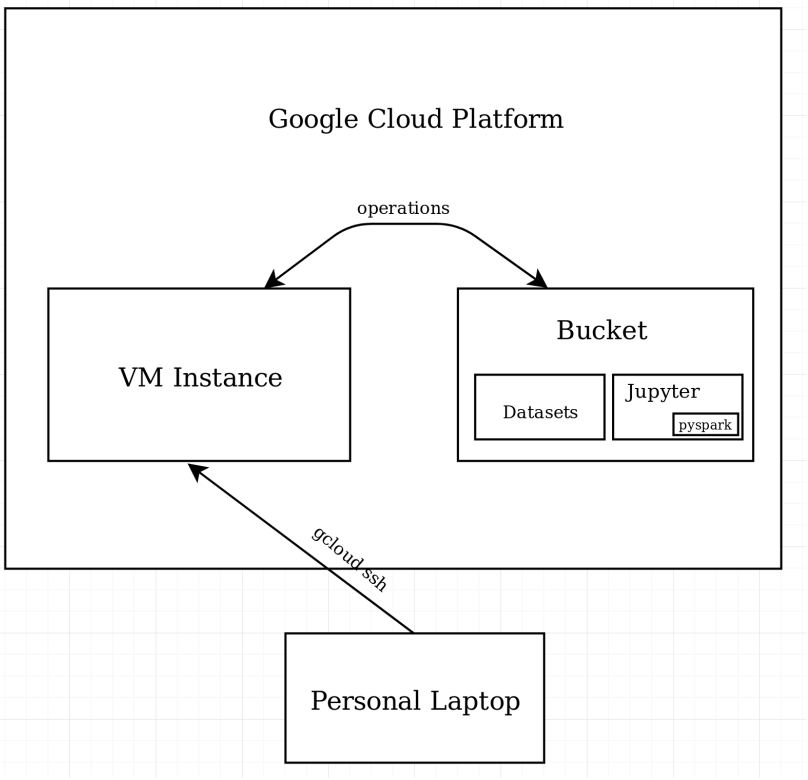

Figure 1: Technologies & Workflow Diagram

3 Development

The development of the project relies on two major concepts - TF-IDF and Jaccard index. With this in mind, we chose to follow a generic approach that would easily allow the reuse of these frequent notions.

3.1 TF-IDF

The TF-IDF metric is a numerical statistic that serves as a measure to reflect how important a word is to individual text documents in the context of a corpus of documents. In order to implement it - under `def tfidf(data, term , document, debug=False)` - we:

1. Compute the number of times `term` has been used in association to `document` .
2. Compute the maximum absolute frequency of any `term` used for `document`.
3. Calculate the term-frequency value of `term` for `document`.
4. Perform a join operation with the number of `document`s with `term` appearing at least once.
5. Calculate the inverse document frequency of `term` considering all `document` s with `term`.
6. Effectively calculate the term frequency–inverse document frequency of `term` for `document`.

3.2 Jaccard index

Given sets A and B the Jaccard index of sets $A, B \neq \emptyset$ is given by the number of elements in $A \cap B$ divided by the number of elements in $A \cup B$. In order to implement it - under `def jiSimilarity(data, col_ref, col_set, debug=False)` - we:

1. Structure dataframe as `col_ref` & Set of `col_set` that are related with `col_ref`.
2. Cross join different `col_ref[:-1]` and the respective sets of `col_set`.
3. Calculate the intersection of `col_set` as i.
4. Calculate the union of `col_set` as u.
5. Computed JI out of i and u.

3.3 Other relevant notes

- Good use of Spark primitives.
- Detailed care with both spatial and temporal performance issues.
- Good use of User-Defined Functions.
- Use of comments in the code to explain the steps in the algorithms.
- Long lines of code avoided.
- Good use of debug messages which help to explain the steps in the algorithms.

4 Expected Results / Tests

4.1 tiny1

☒ tfidfTags matches expected output.

☒ recommendByTag matches expected output.

☒ recommendByTags matches expected output.

☒ jiMovieSimilarity matches expected output.

☒ recommendBySimilarity matches expected output.

4.2 tiny2

☒ tfidfTags matches expected output.

☒ recommendByTag matches expected output.

☒ recommendByTags matches expected output.

☒ jiMovieSimilarity matches expected output.

☒ recommendBySimilarity matches expected output.

4.3 tiny3

☒ tfidfTags matches expected output.

☒ recommendByTag matches expected output.

☒ recommendByTags matches expected output.

☒ jiMovieSimilarity matches expected output.

☒ recommendBySimilarity matches expected output.

4.4 medium1

☒ tfidfTags matches expected output.

☒ recommendByTag matches expected output.

☒ recommendByTags matches expected output.

☒ jiMovieSimilarity matches expected output.

☒ recommendBySimilarity matches expected output.

4.5 medium2

☒ tfidfTags matches expected output.

☒ recommendByTag matches expected output.

☒ recommendByTags matches expected output.

☒ jiMovieSimilarity matches expected output.

☒ recommendBySimilarity matches expected output.

5 Conclusions

The realization of this project provided the group with extensive learning. We were able to acquire a greater knowledge in the area of cloud computing, in particular in the area of data manipulation, more specifically using Spark techniques. We believe all the requirements were met.