Big Data and Cloud Computing Report

André Castro
Department of Computer Science
Faculty of Sciences
University of Porto
Porto, Portugal
up201604447@fc.up.pt

José Pinto
Department of Computer Science
Faculty of Sciences
University of Porto
Porto, Portugal
up201603713@fc.up.pt

***Abstract*—Nowadays the overflowing amount of content existing across the internet has generated many challenges for many corporations from different fields like YouTube, Netflix, Amazon, and more. The ever-prevalent question only proves more challenging as time goes on: What content should prove most appropriate for each user? All these previously mentioned platforms have unimaginable amounts of data that are constantly being fetched, updated or added to. This paper focuses on presenting the work done by our group for the subject of Big Data and Cloud Computing, whose goal was the implementation of one of these recommendation systems for gigantic data sets.**

Project ID: bigdata-269209.

I. Introduction

This work, as previously stated, is focused on providing a system that is able to retrieve, handle and process large data sets in order to provide the most adequate of recommendations to users according to existing ratings and tags associated with each movie.

The project contains many different remote components provided by Google Cloud's IaaS (infrastructure as a service), such as FaaS (function as a service) and DBaaS (Database as a service) and Google Colab was also used which represents a sort of SaaS (Software as a service). Divided in three main phases, the project has for each one of them a corresponding Google Colab notebook. The first of these is the data processing phase, where data needs to be fetched from a certain database and subsequently processed into a certain desired state and stored. Next is the loading phase where a loader cloud function must retrieve the output of the previous phase and store it in Google Cloud's BigQuery service. This proves necessary in the last phase, the search phase, where a so called search cloud function answers http requests from users and makes the correct queries in order to obtain the best recommendations possible. The base algorithm used to ascertain these is the TF-IDF algorithm whose aim is to combine a certain element's term frequency and inverse document frequency in order to achieve a balanced and more relevant way to grade said element's relevance.

It should be noted that the project has three optional goals to achieve a more complete algorithm: an improvement over the original TF-IDF algorithm resorting on weighted factors; the implementation of an algorithm that compares movies according to the jaccard index and, lastly, the capability to obtain and process additional data sets from IMDB. From this set of additional works, all have been successfully implemented.

II. Summary

To summarize the work done by our group, these are the efforts we achieved:

- Data Processing Notebook exercises successfully done;
- LCF Notebook exercises successfully done and remote LCF integrated;
- SCF Notebook exercises successfully done and remote SCF integrated;
- Optional Weighted TF-IDF implemented;
- Optional Jaccard Index implemented;
- Optional IMDB compatible system integrated.

It should be noted that all components were tested for all given data sets from all sizes. In BigQuery there are 3 tables present for each data set:

- ”movies_agg” with information on each movie;
- ”tfidf” with information relating words, movies and TF-IDF values;
- ”ratings_tags” which is necessary for the success of SCF's Jaccard index function.

An example of an ID of one of these BigQuery tables would be ”bigdata-269209:large3.movies_agg”.

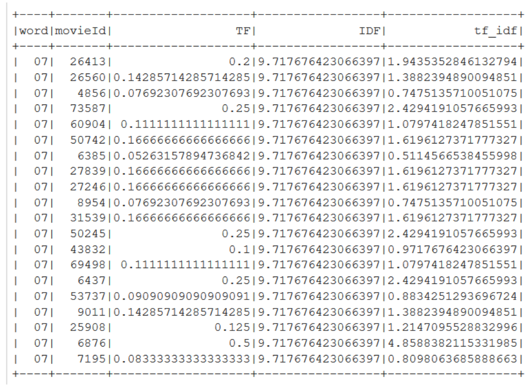
III. Data Processing

The first component is the Data Processing part, which is done in its respective notebook. Common to all notebooks, there are introductory parts destined to implement a connection to Google Drive and authentication within the GCP of the project as well as doing necessary setups for needed tools (in this case Spark).

Introductions aside, this notebook is responsible for obtaining the required data sets that will be processed afterwards (this being both movielens' data set and IMDB's name and crew data sets). In order to do this, the download from the correct repositories was made, as well as the extraction of its contents.

Next a remote repository was needed in order to access all these data sets and another where all processing done to these could be stored. For this reason, two GCP storage buckets were made, the first would receive all parquet files from the databases. From here, these parquet files along with the .csv link file and .tsv files from IMDB would be read.

The crew .tsv file was processed with pyspark in order to obtain lists of writers and directors for each movie to then merge these two tables to obtain all crew members. Afterwards a database with all ratings per movie and another with all movies along with their year of release, number and average of ratings were prepared. Additionally databases for crew members with either name or id by the respective movie were also implemented.



```
+----+-------+-------------------+-----------------+------------------+
|word|movieId|                 TF|              IDF|            tf_idf|
+----+-------+-------------------+-----------------+------------------+
|  07|  26413|                0.2|9.717676423066397|1.9435352846132794|
|  07|  26560|0.14285714285714285|9.717676423066397|1.3882394890094851|
|  07|   4856|0.07692307692307693|9.717676423066397|0.7475135710051075|
|  07|  73587|               0.25|9.717676423066397|2.4294191057665993|
|  07|  60904| 0.1111111111111111|9.717676423066397|1.0797418247851551|
|  07|  50742|0.16666666666666666|9.717676423066397|1.6196127371777327|
|  07|   6385|0.05263157894736842|9.717676423066397|0.5114566538455998|
|  07|  27839|0.16666666666666666|9.717676423066397|1.6196127371777327|
|  07|  27246|0.16666666666666666|9.717676423066397|1.6196127371777327|
|  07|   8954|0.07692307692307693|9.717676423066397|0.7475135710051075|
|  07|  31539|0.16666666666666666|9.717676423066397|1.6196127371777327|
|  07|  50245|               0.25|9.717676423066397|2.4294191057665993|
|  07|  43832|                0.1|9.717676423066397|0.9717676423066397|
|  07|  69498| 0.1111111111111111|9.717676423066397|1.0797418247851551|
|  07|   6437|               0.25|9.717676423066397|2.4294191057665993|
|  07|  53737|0.09090909090909091|9.717676423066397|0.8834251293696724|
|  07|   9011|0.14285714285714285|9.717676423066397|1.3882394890094851|
|  07|  25908|              0.125|9.717676423066397|1.2147095528832996|
|  07|   6876|                0.5|9.717676423066397|4.8588382115331985|
|  07|   7195|0.08333333333333333|9.717676423066397|0.8098063685888663|
+----+-------+-------------------+-----------------+------------------+
```

Fig. 1. Example of output relating TF, IDF and TF-IDF

All this proved necessary when further obtaining all the different words necessary for the TF-IDF algorithm since all these components aid in establishing how relevant a movie is according to a given user movie search input. Then, combining both TF (term frequency) and IDF (inverse document/movie frequency), we obtain a new data set with all words per movie with these respective values plus the intended TF-IDF metric.

Additionally a new data set was prepared regarding the Jaccard index (the level of similarity between two movies according to certain characteristics) where movies are aggregated with all the users that either have rated them or tagged them.

Finally the data sets containing movie information, the word's TF-IDF values and the one needed for determining the Jaccard index between two movies were saved as Parquet files and transferred to the previously mentioned output bucket. Ending this step, a PubSub message is sent to the Loader Cloud Function discussed further, notifying it of this step's successful termination.

IV. Loader Cloud Function

This component, despite brief, proves having great significance to the project due to being responsible for preparing the DBaaS (in this case GCP's BigQuery) necessary for accessing the required data to answer the user's needs.

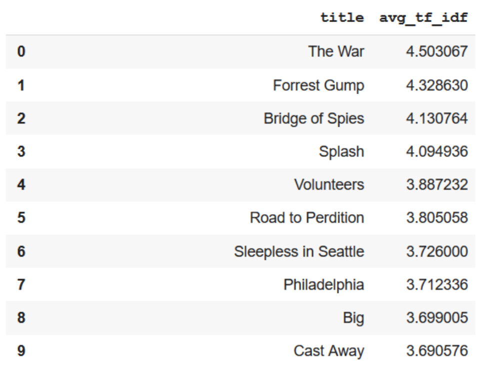
The setup process aside, the LCF fetches the parquet files from the output bucket after receiving the already mentioned PubSub message and starts preparing for each of them their respective BigQuery tables. Again, these are the movie table, the TF-IDF table and the ratings/tags table for Jaccard. This means that, first of all, the column fields were implemented according to the ones present in the corresponding Parquet files. All fields need to have a correct specified value type that was integrated correctly. These BigQuery tables were then populated with all corresponding entries and are now ready for the queries from the SCF.

This LCF was implemented both in the Google Colab Notebook and in its own GCP Cloud Function within our project.

V. Search Cloud Function

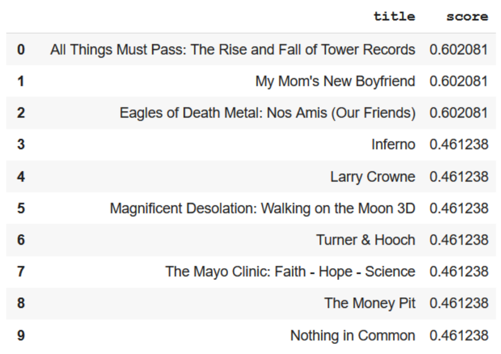
Lastly there is the Search CLoud Function, responsible for providing the best movie choices to users regarding their movie search inputs sent through HTTP requests.

There are many possible operations within this component that can be accessed as needed thanks to a parameter named "op". These can be simple listings of the movie or TF-IDF BigQuery tables, the more complex TF-IDF algorithms (normal and weighted TF-IDF) that return the most appropriate movies according to the user's search choices, or even the listing of the movies more similar to a given one according to the Jaccard Index. All these operations are done via SQL queries sent to the correct BigQuery databases and contain other general parameters such as a limit of the data that is to be retrieved and the specific data set that is to be accessed.



	title	avg_tf_idf
0	The War	4.503067
1	Forrest Gump	4.328630
2	Bridge of Spies	4.130764
3	Splash	4.094936
4	Volunteers	3.887232
5	Road to Perdition	3.805058
6	Sleepless in Seattle	3.726000
7	Philadelphia	3.712336
8	Big	3.699005
9	Cast Away	3.690576

Fig. 2. TF-IDF search on "Tom Hanks War Movie" from the Large5 data set



	title	score
0	All Things Must Pass: The Rise and Fall of Tower Records	0.602081
1	My Mom's New Boyfriend	0.602081
2	Eagles of Death Metal: Nos Amis (Our Friends)	0.602081
3	Inferno	0.461238
4	Larry Crowne	0.461238
5	Magnificent Desolation: Walking on the Moon 3D	0.461238
6	Turner & Hooch	0.461238
7	The Mayo Clinic: Faith - Hope - Science	0.461238
8	The Money Pit	0.461238
9	Nothing in Common	0.461238

Fig. 3. Weighted TF-IDF search on "Tom Hanks War Movie" from the Large5 data set

There are slight differences in some parameters that each of these functions receive, such as the Jaccard Index function (which relates two movies according to the similarity of users who tagged or rated either of them) needing a certain movie name in order to know what it will base its returning recommendations on. During the implementation of the SCF it was made sure that each of them are mandatory, otherwise an error is thrown.

Here we also had to take some precautions regarding both versions of the TF-IDF algorithm. The normal version is a simple average of all TF-IDF movie/word combination values and the weighted TF-IDF, among other changes combines the normal version with a formula using the ratings in order to recommend better rated movies easier. Both this formula and the TF-IDF average have weights associated with them in order to establish how relevant each component is. It should be noted that this average TF-IDF does count words received by the user may not be present and, therefore, some specific TF-IDF values need to be attributed a value of zero.

Again, all these functions were also integrated both in the Google Colab Notebook and the respective GCP Cloud Function.

VI. Conclusion and Final Thoughts

This project's main goal is the creation and implementation of a movie recommendation system based on technologies taught during the course's classes, such as Python, libraries like PySpark and Pandas and the whole GCP. From the expected project requirements, all have been successfully achieved plus all the optional objectives. A whole system that is able to fetch data from large existent data sets like IMDB, process it according to its needs and grant recommendations to users based on many different algorithms.

For this reason we believe the project proved a fruitful learning experience and a great way to delve into some practices regarding Big Data and to obtain more familiarity with Cloud Computing systems that offer a grand array of services.