Big Data - Project Report

Sebastiano D'Arconso (VR489066)
Tommaso Del Prete (VR488382)

1 Introduction

The goal of this project is to analyze a dataset of hotel reviews and to extract information about the hotel's quality based on the customers' reviews and ratings. We used the MapReduce programming model to process the data and to extract the information we needed, we then used a Deep Learning model to perform sentiment analysis on the reviews. Finally, using PySpark ML, we trained a model to perform the same task for ablation studies.

2 Background

In this section, we will briefly introduce all the concepts necessary to understand the project.

2.1 MapReduce

MapReduce is a programming model designed by Google for processing large data sets with a distributed algorithm on a cluster.

It consists of two main phases: Map and Reduce.

Map Function

- **Input:** Takes a set of data.
- **Processing:** Generates intermediate key-value pairs.
- **Output:** Emits intermediate key-value pairs.

Reduce Function

- **Input:** Takes intermediate key-value pairs.
- **Processing:** Merges values for each key to produce final results.
- **Output:** Emits final key-value pairs.

The MapReduce approach offers several advantages, particularly when dealing with large-scale data processing tasks. One of the main benefits is its scalability. By breaking down tasks into smaller, manageable chunks (map tasks) and then combining the results (reduce tasks), MapReduce can efficiently handle vast amounts of data across numerous distributed machines. This ability to distribute the workload allows for parallel processing, which significantly speeds up data processing times.

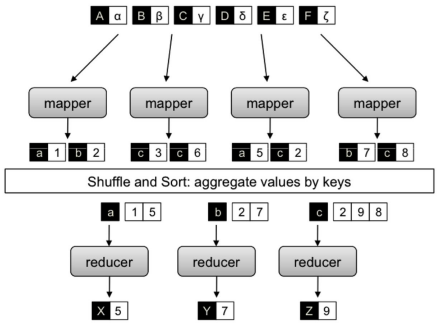

Figure 1: An example of MapReduce workflow

Another advantage is fault tolerance. MapReduce systems are designed to be resilient to individual node failures. If a node crashes or becomes unavailable, the system can reassign the task to another node without affecting the overall job. This ensures reliability and consistency in data processing, even in the face of hardware failures.

Moreover, MapReduce also simplifies the programming model for distributed computing. Programmers do not need to worry about the complexities of parallelization, data distribution, and fault tolerance, as the framework abstracts these details away. This allows developers to focus on writing the core logic of their applications using a straightforward map and reduce functions.

2.2 Sentiment Analysis with Deep Learning

Sentiment analysis involves determining the sentiment or emotional tone behind a series of words. It is a crucial task in natural language processing (NLP) with applications in various fields such as marketing, customer service, and social media monitoring. Deep learning has revolutionized sentiment analysis by providing sophisticated models that can capture intricate patterns in textual data, leading to improved accuracy and performance.

One of the most popular deep learning models for sentiment analysis is the Long Short-Term Memory (LSTM) network. LSTMs are a type of recurrent neural network (RNN) that can capture long-range dependencies in sequential data. They are well-suited for processing text data due to their ability to model context and sequential patterns.

However, nowadays more advanced models such as the Transformer architecture have further improved sentiment analysis performance, using self-attention mechanisms to capture global dependencies in the input sequence and enabling them to model long-range relationships more effectively than RNNs or LSTMs.

2.3 PySpark, PySpark ML and PySpark SQL

Apache *Spark* is a multi-language engine for executing data engineering, data science, and machine learning on single-node machines or clusters. [1] *PySpark* is the Python API for Apache Spark, a powerful open-source distributed computing system that provides an interface for programming entire clusters with implicit data parallelism. PySpark allows the user to interface with RDDs (Resilient Distributed Datasets) in Python, which are distributed collections of objects across a cluster that can be manipulated in parallel. *PySpark ML* is the machine learning library provided by PySpark, which provides a set of high-level APIs built on top of DataFrames that help users create and tune practical machine learning pipelines. *PySpark SQL* is a Spark module for structured data processing, which provides more information about the structure of both the data and the computation being performed, it also allows to query of the data using SQL.

3 Dataset

The dataset we used is the "TripAdvisor Dataset", which is provided as a text file containing reviews of hotels scraped from TripAdvisor from around the world. The dataset contains 878561 reviews taken from 4333 hotels, and each review is a JSON object with the following structure:

- root
  - ratings (float from 1 to 5)
    - overall
    - cleanliness
    - location
    - rooms
    - service
    - sleep_quality
    - value
  - title (string)

[1] The "cluster" consists of one or more computers/machines working together to provide high availability, reliability, and scalability towards the service being provided to clients. If one server/machine fails then work/resources get distributed among other machines in the same cluster. The "single node machine" refers to a Spark deployment where all components of the Spark application run on a single physical or virtual machine. This setup contrasts with a multi-node cluster, where Spark components are distributed across multiple machines.

- text (string)
- author (string)
  - username (string)
  - num_cities (integer)
  - num_helpful_votes (integer)
  - num_reviews(integer)
  - num_type_reviews (integer)
  - id (integer)
  - location (string)
- date_stayed (string)
- offering_id (integer)
- num_helpful_votes (integer)
- date (string)
- id (integer)
- via_mobile (boolean)

Where the *ratings* field contains the ratings from 1 to 5 of the hotel in different categories, the *title* field contains the title of the review, the *text* field contains the review itself, the *author* field contains information about the author of the review, the *date_stayed* field contains the date when the author stayed at the hotel, the *num_reviews* field contains the number of reviews received by the hotel, the *offering_id* field contains the id of the hotel, the *num_helpful_votes* field contains the number of helpful votes the review received, the *date* field contains the date when the review was written, the *id* field contains the id of the review, and the *via_mobile* field contains a boolean value indicating if the review was written from a mobile device.

4 Pipeline

Our goal is to extract and provide the best hotel over all the hotels in the dataset, ranked based on the quality of the reviews and the results of the Sentiment Analysis. To provide a more accurate solution, the quality of the hotel is not only based on the average ratings received but also on the sentiment value of the reviews, and the sentiment score which are extracted using a Deep Learning model. The extraction of the reviews and the aggregation under the same hotel (identified by a unique id) is done using MapReduce, while the sentiment analysis is done using a Deep Learning model. Both the MapReduce job and the sentiment analysis will be explained in detail in the next sections.

4.1 Data Preprocessing

The data preprocessing phase is entirely done using MapReduce, excluding a preliminary parsing part where a function takes each row of the dataset txt file and parses it as a JSON object, returning it. We defined two map functions and two reduce functions that will be used in two different MapReduce jobs.

4.1.1 MapReduce

To perform MapReduce we had to first define the *map* and the *reduce* functions. The *map function* takes a parsed JSON object (line) as input and converts it into a dictionary to have easier access to its fields 3. As explained in 2.1, this function emits a key-value pair where the key is the hotel id and the value is a dictionary containing the average of the ratings and the text of the review. The *reduce function* takes the key-value pairs emitted by the map function and merges the reviews for the same hotel also calculating the average of the ratings. The *MapReduce* job extracts and merges the reviews for the same hotel, producing a series of files where each of them contains the reviews merged for every single hotel identified by the same id. For each hotel id, we now have all its reviews merged and the average of the ratings it received. This process outputs multiple files ($\cong 35$), so we merged them into a single text file and the time taken for this job is $\cong 13$ seconds.

4.2 Sentiment Analysis

To perform Sentiment Analysis we had to first set some constraints to reduce the time taken for the process to complete. We decided to perform the analysis only on the hotels that received at least 1000 reviews, then, for each hotel that satisfies this constraint, we sampled **n** reviews from it, where **n** is given using a function that calculates the sample size for a finite population, derived by the formula for sample size in statistics. The formula is:

$$n = \frac{N \cdot Z^2 \cdot p \cdot (1-p)}{(N-1) \cdot e^2 + Z^2 \cdot p \cdot (1-p)} \tag{1}$$

and helps us reduce the reviews to a manageable size, but still, have a good representation of the reviews for the hotel.

For each hotel and its sampled reviews, we used *flair*, a powerful SoA Deep Learning framework to perform sentiment analysis on them. Each review is tagged with a label that can be either positive, negative or neutral. We then converted the labels to a numerical value, where positive is 2, negative is 0 and neutral is 1 and averaged them weighting with the confidence of the inferences, to get the sentiment of the hotel. The sentiment computed through this process is compared with the average rating of the hotel from 4.1.1 to have a measure of the quality of the hotel based on two different sources of information. The more equal the sentiment and the average rating are, the more reliable the quality of the hotel is. We then plotted a scatter plot where the x-axis is the average

rating of the hotel and the y-axis is the sentiment of the hotel, moreover, each point size is proportional to the number of reviews the hotel received, to avoid hotels with few reviews being considered as good or bad based on an inconsistent number of reviews. We also tried to load the model during the MapReduce jobs to parallelize the sentiment analysis, but the model was too heavy to be loaded in each node of the cluster, exponentially increasing the time taken for the job to complete. The Sentiment Analysis was performed on a GPU, specifically a *NVIDIA GTX 1080* taking $\cong$ 10 minutes to complete, without the GPU the time taken was $\cong$ 30 minutes.

4.3 Sentiment Analysis with PySpark ML

As an ablative study, we decided to implement our sentiment analysis pipeline using PySpark ML, to compare the results with the ones obtained using the traditional Deep Learning models. The dataset we used is the same as before but already processed by the MapReduce jobs.

- *Tokenization*: Each review is divided into words using the *Tokenizer* class from the *pyspark.ml.feature* module.
- *Stopwords removal*: The most frequent words in the English language are removed using the *StopWordsRemover* class from the *pyspark.ml.feature* module since they do not provide useful information for the sentiment analysis.
- *HashingTF*: The hashing is used to map a word to an index in a feature vector, using the *HashingTF* class from the *pyspark.ml.feature* module.
- *Logistic Regression fit*: We trained a Logistic Regression model using the *LogisticRegression* class from the *pyspark.ml.classification* module.
- *Evaluation*: We evaluated the model using the *BinaryClassificationEvaluator* class from the *pyspark.ml.evaluation* module.

The model we trained is a simple Logistic Regression model, but we could have used more complex models such as Random Forests, Gradient-Boosted Trees, or even Neural Networks. The final accuracy of the model is $\cong$ 0.80, which is a good result considering the simplicity of the model and the small dataset used for training.

5 Experiments and results

In this section, we will show and explain the results we obtained.

5.1 MapReduce Results

The MapReduce job took ≅ 13 seconds to complete, producing 35 different files containing the reviews merged for each hotel. We then proceeded to merge them into a single file for the sentiment analysis. The sentiment analysis took ≅ 10 minutes to complete using a GPU, and ≅ 30 minutes without it. The scatter plot shows the best hotels with > 1000 reviews.

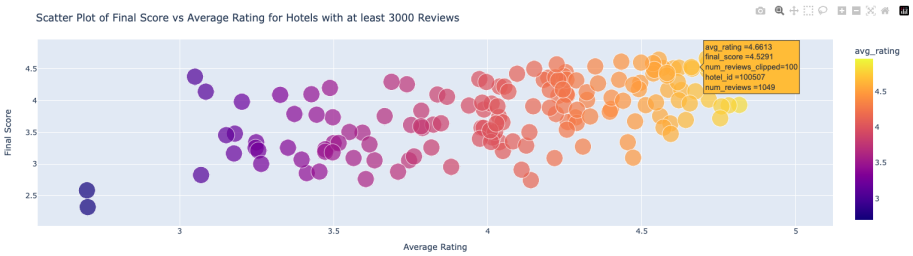

Figure 2: Scatter plot of the hotels with > 1000 reviews.

Passing the cursor over a point will show:

- the id of the hotel
- the final score of the hotel
- the number of reviews the hotel received
- the number of reviews the hotel received (clipped to 100 for plotting purposes)

The *final score* is a value obtained by the average of the sentiment score as given by the sentiment analysis scaled from 0 to 5, since the sentiment score is a value that varies from 0 to 2, with 0 being negative, 1 being neutral and 2 being positive. The dimension of the points is proportional to the number of reviews the hotel received, clipped to 100 and the color of the points is based on the average rating of the hotel. When choosing a hotel the upper right corner of the plot will show the "best" hotels based on the average rating and the sentiment score, so the hotels that are most consistent in the two measures. The dimension of the point will also give us an idea of the number of reviews the hotel received, so we can avoid hotels with few reviews being considered as good or bad based on an inconsistent number of reviews. The average sentiment score is the percentage of certainty of the sentiment analysis, averaged for all the reviews for that specific hotel, this means that a hotel with a high average sentiment had a more consistent sentiment certainty for all the reviews it received, so this depends on the model used for the sentiment analysis. In the table 1 and 2 we can see the top 5 hotels based on the average sentiment and the average rating, giving importance to the number of reviews the hotel received.

hotel_id	avg_sentiment	avg_rating
93340	1.8668	4.7108
224221	1.8585	4.5552
113317	1.8459	4.4336
1503474	1.8425	4.7154
84502	1.8397	4.4977

Table 1: Hotel Information and Ratings

final_score	avg_sentiment_score	num_reviews
4.6669	0.9901	1393
4.6462	0.9879	1682
4.6149	0.9906	2340
4.6063	0.9822	1842
4.5992	0.9840	1275

Table 2: Scores and Review Count

As shown in the tables, the hotel with id 93340 is the best in the dataset based, with a final score of 4.6669.

5.2 PySpark ML Results

For the experiments with PySpark ML, we used as a dataset the output of the MapReduce job. For this experiment, we used PySpark SQL, which is a Spark module for structured data processing, with a machine learning pipeline for NLP that includes the following steps, explained in 4.3:

- Splitting the dataset into training and testing sets
- Tokenization
- Stopwords removal
- HashingTF
- Logistic Regression fit
- Evaluation

Note that to train and test the model we had to provide also the labels for the reviews, and since we don't have them in the dataset we used the sentiment score obtained from the sentiment analysis. The dataset was automatically split into 80% training and 20% testing, with 29472 rows for training and 7450 rows for testing. The number of positive examples is 28785 and the number of negative examples is 8130. After fitting the model we obtained an accuracy of 88.10 %, taking $\cong$ 17 seconds to complete.

6 Conclusions

In this project, we implemented a pipeline to extract and analyze hotel reviews from a dataset using MapReduce and to perform sentiment analysis on the reviews using Deep Learning models. We also implemented the same pipeline using PySpark ML to perform ablation studies. The MapReduce job was able to extract all the information we needed working in parallel and saving a lot of time, the sentiment analysis was the bottleneck of the process, but it was able to provide useful information about the quality of the hotels. The PySpark ML model was able to achieve good accuracy, showing that it is possible to use PySpark ML for sentiment analysis on large datasets in a reasonable amount of time.