UNIVERSITÀ DEGLI STUDI DI MILANO
FACULTY OF SCIENCE AND TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE

Master in
Computer Science

ALGORITHMS FOR MASSIVE DATASETS
FINAL REPORT ABOUT RECOMMENDER SYSTEM

Teacher: Prof. Dario Malchiodi

Final report written by:
Samuele Simone
Matr. Nr. 11910A

Academic year 2022-2023

I/We declare that this material, which I/We now submit for assessment, is entirely my/our own work and has not been taken from the work of others, save and to the extent that such work has been cited and acknowledged within the text of my/our work. I/We understand that plagiarism, collusion, and copying are grave and serious offences in the university and accept the penalties that would be imposed should I engage in plagiarism, collusion or copying. This assignment, or any part of it, has not been previously submitted by me/us or any other person for assessment on this or any other course of study.

Contents

This report is made up of 8 chapters. In the **Chapter 1** we give at the reader a general view of what is a recommender system and where we can find it. **Chapter 2**, we start to discuss about the development setup for the project. Then, in the **Chapter 3** we focus on the dataset, how is composed and we explore the data. Later, in the **Chapter 4** we explain the recommender system. the different approaches and the mechanisms behind. In order to evaluate the project developed there are some notion about scalability and complexity in the **Chapter 6**. Consequently , we summarize the aspects and the results obtained during the various experiments in the **Chapter 7**. Finally there is the conclusion in last **Chapter 8**.

Chapter 1

Introduction

A recommender system is used everywhere nowadays. Indeed all big companies are pushing in these systems because they can increase the sells about their product, e.g., when we are scrolling a product on Amazon, then they show us a list of recommendation based on the item selected.

Recommendation system use a number of different technologies. We can split these system into two broad groups: [1]

- *Content-based systems*: examine properties of the items recommended. For instance, if a Netflix user has watched many cowboy movies, then recommend a movie classified in the database as having the "cowboy" genre.
- *Collaborative filtering* systems recommend items based on similarity measures between users and/or items.

Chapter 2

Environment setup

Before going deeper into the project we must discuss about the entire environment was setup. Indeed I used different libraries in order to create the recommender system. Here the snippet of all libraries used in the project:

```
#importing the required pyspark library
import pyspark
from pyspark import SparkConf
from pyspark.sql import SparkSession
from pyspark.ml.evaluation import RegressionEvaluator
from pyspark.ml.recommendation import ALS
from pyspark.sql.functions import split
from pyspark.sql.functions import array_contains
from pyspark.sql.functions import col
from pyspark.sql.functions import from_json
from pyspark.sql.functions import when
from pyspark.sql import functions as F
from pyspark.ml.feature import VectorAssembler
from pyspark.ml.feature import StringIndexer
from pyspark.ml.feature import MinMaxScaler
from pyspark.ml.feature import Normalizer
from pyspark.ml.linalg import Vectors
from pyspark.sql import Row
from pyspark.sql.window import Window
#import for manage global var
import os
#import for graphics
import matplotlib.pyplot as plt
import pandas as pd
#import for regular expression
import re
import numpy as np
from sklearn.metrics.pairwise import cosine_similarity
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
```

```
31 from sklearn.model_selection import cross_val_score
32 from sklearn.metrics import accuracy_score
```

Listing 2.1: Loading all the python libraries

2.1 Pyspark setup

PySpark is the Python API for Apache Spark. It enables you to perform real-time, large-scale data processing in a distributed environment using Python. It also provides a PySpark shell for interactively analyzing your data.

PySpark combines Python's learnability and ease of use with the power of Apache Spark to enable processing and analysis of data at any size for everyone familiar with Python.

PySpark supports all of Spark's features such as Spark SQL, DataFrames, Structured Streaming, Machine Learning (MLlib) and Spark Core. [2]

Refering to the Listing 2.1 we can see that there are several libraries about PySpark. I will discuss about the most important:

- **SparkConf**: Configuration for a Spark application. Used to set various Spark parameters as key-value pairs. [3]
- **SparkSession**:The entry point to programming Spark with the Dataset and DataFrame API. [4]
- **pyspark.ml**: DataFrame-based machine learning APIs to let users quickly assemble and configure practical machine learning pipelines. [5]
- **pyspark.sql.functions**:A list of function that allow the user to explore dataframe [6]

In the project I used PySpark Dataframe. They are distributed collections of data that can be run on multiple machines and organize data into named columns. These dataframes can pull from external databases, structured data files or existing resilient distributed datasets (RDDs).

2.2 Kaggle setup

The data were hosted on the Kaggle platform. Kaggle is a subsidiary of Google, it is an online community of data scientists and machine learning engineers.

Kaggle allows users to find datasets they want to use in building AI models, publish datasets, work with other data scientists and machine learning engineers, and enter competitions to solve data science challenges.

Kaggle got its start in 2010 by offering machine learning and data science competitions as well as offering a public data and cloud-based business platform for data science and AI education. [7]

In order to access to the data Kaggle platform gives to every register account an API credential that is necessary to download the datasets.

```
os.environ['KAGGLE_USERNAME'] = 'your_kaggle_username'
os.environ['KAGGLE_KEY'] = 'your_kaggle_key'
```

Listing 2.2: API Credential for accessing the data

2.3 Pandas setup

Pandas is a Python package providing fast, flexible, and expressive data structures designed to make working with "relational" or "labeled" data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, real-world data analysis in Python. Additionally, it has the broader goal of becoming the most powerful and flexible open source data analysis/manipulation tool available in any language. It is already well on its way toward this goal. [8]

I used it to explore the data, as we will see in the chapter 3, and to build the content-based recommendation system. My goal was to implement the recommendation system with different types of approaches, and Pandas is one of them. However, in the course of the paper I will emphasize the different ways in which I tried my hand at it.

2.4 Scikit-learn setup

Scikit-learn is a library in Python that provides many unsupervised and supervised learning algorithms. It's built upon some of the technology like NumPy, pandas, and Matplotlib. The functionality that scikit-learn provides include [9]:

- Regression, including Linear and Logistic Regression
- Classification, including K-Nearest Neighbors
- Clustering, including K-Means and K-Means++
- Model selection
- Preprocessing, including Min-Max Normalization

Indeed some of these functionality are used in this project such as the K-NN, the second different approach to build the Content based recommender system.

2.5 Numpy setup

NumPy is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more. [10]

It was useful in constructing the function `cosine_similarity_scratch` to perform multiplication and normalization of vectors.

Chapter 3

Dataset: A look inside

3.1 Data loading

We start to look how the data is loaded inside the project. First of all the data is downloaded from Kaggle with this code:

```
!kaggle datasets download -d yelp-dataset/yelp-dataset
```

Listing 3.1: Download dateset

and proceeded to unzip the dataset obtaining:

```
Archive:  /content/yelp-dataset.zip
  inflating: yelp-dataset/Dataset_User_Agreement.pdf
  inflating: yelp-dataset/yelp_academic_dataset_business.json
  inflating: yelp-dataset/yelp_academic_dataset_checkin.json
  inflating: yelp-dataset/yelp_academic_dataset_review.json
  inflating: yelp-dataset/yelp_academic_dataset_tip.json
  inflating: yelp-dataset/yelp_academic_dataset_user.json
```

Listing 3.2: Unzip dateset

These json files are huge and to manage it we load the data into a Pyspark df with this command:

```
df_review = spark.read.json('/content/yelp-dataset/
    yelp_academic_dataset_review.json')
```

Listing 3.3: Loading json data into Pyspark df

We repeat this operation many time as the number of the json files. So we obtain these dataframe:

- **df_review**: (review_id,user_id,business_id,stars_review). These are the attributes I selected from the df. However, there is one field that I think should be mentioned which is the text field, very useful for doing further

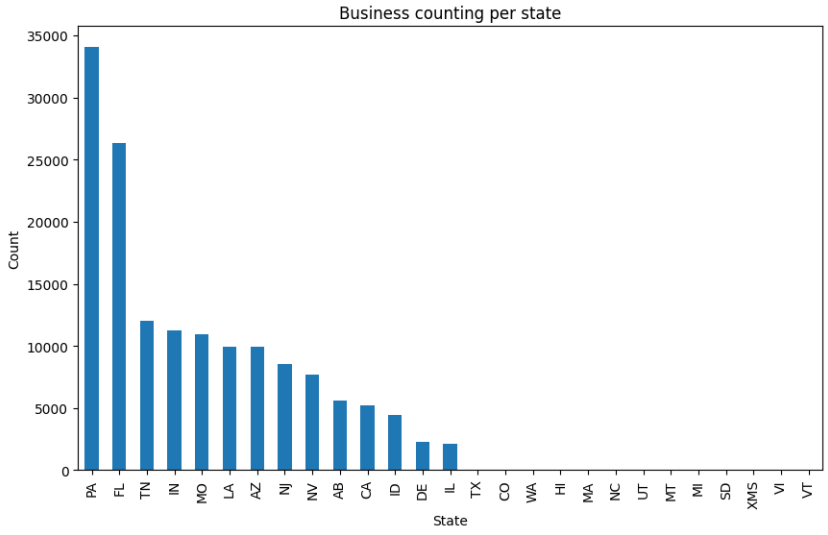
analysis. In my case it was not selected as I believe that the quoted data is sufficient for our purposes.

- **df_users**:(user_id,username,average_stars)
- **df_business**: (business_id,address,attributes,categories,city,is_open,name,stars,state). There are others column that I prefer to remove because is not adding values for our analysis.

3.2 Visualizing the data

To give a main idea of the data I create a chart where on the x-axes there are all the states and on the y-axes the count of the total business for that particular state. From the Figure 1 we can see that in "PA", we refer to Pennsylvania, a

Figure 1: Business counting per state



state located in the northeastern part of the United States. So I decided to conduct my analysis over the "PA" state due the amount of businesses present over there. Furthermore due the high dimension of the df it's difficult to create others type of graphics.

3.3 Data Filtering

As we discussed in the previous section 3.2 we want to targeting our recommender system in a specific area for a specific type of business. So I decided to take *"PA"* as

state and *"Restaurant"* as type of business. Moreover I filtered the new dataframe called `restaurant_df` with another parameter called `is_open` that when is equal to 1 means that this business is actually a running business.

```
# Filtering data
restaurant_df = df_business_pandas[(df_business_pandas['state'] ==
    state) & (df_business_pandas['is_open'] == 1) &
   df_business_pandas['categories'].str.contains(type_business)==
   True].reset_index()
```

Listing 3.4: Filtering Pyspark df

Here an example of `restaurant_df` printed: Note that the attribute and category

index	address	attributes	business_id	categories	city	is_open	name	stars	state
3	935 Race St	(None, None, 'full_bar', {'touristy': False, '...	MTSW4McQd7CbVtyjqoe9mw	Restaurants, Food, Bubble Tea, Coffee & Tea, B...	Philadelphia	1	St Honore Pastries	4.0	"PA"

Table 1: Example of a restaurant_df's row

columns must be preprocessed before being used.

3.4 Data preprocessing

Lets start to analyze the attribute column. As we can see they are in this form:

```
Row(AcceptsInsurance=None, AgesAllowed=None, Alcohol="u'none'",
   Ambience=None, BYOB=None,..)
```

Listing 3.5: Attributes item example

So we need a function that allow us to extract key-value pairs in a format understandable to Python. Here the snippet:

```
# Function for extracting key-value from attributes
def extract_values(row):
    if row is None or row == "{}":
        return {}

    attributes = {}
    pattern = r"(\w+)\s*=\s*([^,]+)"
    matches = re.findall(pattern, row)

    for key, value in matches:
        value = value.strip("'")
        attributes[key] = value

    return attributes
```

Listing 3.6: Extracting key-value from attributes

Table 2: Example of `restaurant_df_attr` elements

BestNights	RestaurantsCounterService	WiFi	...	Smoking	CoatCheck
0	1	0	...	0	0
1	0	1	...	1	1

To achieve the result I used the regular expression. More specifically the line 7 defines a regular expression pattern.The pattern captures two groups:

- the key (composed of one or more word characters \w+)
- the value (composed of one or more characters that are not a comma \[^,]+), separated by an equal sign with optional whitespace around it.

Then we apply the function over all the rows of column attributes. Now that the format seems to be good we proceed to create an entire `restaurant_df_attr` in which the attributes are extracted making a new column of the df and populating it with 0 or 1 based on their original value. An example are in the Table 2.

We apply this reasoning also for categories obtaining the `df_categories_dm`. It will be useful to create the `df_final`, a fundamental component for the recommender system. So in the Chapter 4 we are going to understand how recommendation systems work in theory.

Chapter 4

Recommender system

Chapter 5

Creation of a Content-based Recommender system

Chapter 6

Scalability and complexity

Chapter 7

Results and experiments

Chapter 8

Conclusion

Listings

Bibliography

[1] Anand Rajaraman, Jure Leskovec, and Jeffrey D. Ullman. *Mining Massive Datasets*. Cambridge University Press, 2014.

[2] Apache Spark. Pyspark overview, 2023.

[3] Apache Spark. pyspark.sparkconf, 2023.

[4] Apache Spark. pyspark.sparksession, 2023.

[5] Apache Spark. pyspark.ml, 2023.

[6] Apache Spark. pyspark.sql.function, 2023.

[7] Mahmoud Hassan Mahmoud. What is kaggle?, 2022.

[8] Pandas. Pandas package overview, 2023.

[9] codecademy. What is scikit-learn?, 2023.

[10] codecademy. What is numpy?, 2023.