Market-basket Analysis in IMDb Dataset

Algorithms for Massive Datasets
Università degli Studi di Milano
Prof. Dario Malchiodi

UNIVERSITÀ
DEGLI STUDI
DI MILANO

Peesapati Venkata Sai Vamsi(933987)
Venkatasaivamsi.peesapati@studenti.unimi.it

Revan tokathi(924464)
revan.tokathi@studenti.unimi.it

Contents

We declare that this material, which will be now submitted for assessment, is entirely 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 our work. We understand that plagiarism, collusion, and copying are grave and serious offences in the university and accept the penalties that would be imposed should we engage in plagiarism, collusion or copying. This assignment, or any part of it, has not been previously submitted by us or any other person for assessment on this or any other course of study.

1 THE DATASET

One of the main issues in data science is frequent item sets, especially when dealing with larger amounts of data. This is because it is complicated for human examinations, thus a great effort is needed on handling the large amounts of data.

In this assignment, we have used IMDb dataset that is found in kaggle. Each dataset is contained in a gzipped, tab-separated-values (TSV). The first line in each file contains headers that describe what is in each column. A '\N' is used to denote that a particular field is missing or null for that title/name. It is framed by five distinct tables, of which just three have been considered:

- *name.basics*: it contains information about the crew members. The variables are the following:

  nconst : unique identifier of the actors.

  PrimaryName : name of the actor.

  birthYear : year of birth of the actor. It has been dropped.

  PrimaryProfession : different professions of the actors.

  KnownForTitles : list of movie Id's. It has been dropped.

  – *title.basic* : This dataset contains the information for titles.

    tconst : unique identifier of title.
    titleType : the type/format of the title. It has been dropped.
    primaryTitle : the popular title used at point of release.
    originalTitle : original title.
    isAdult : 0 - Adult, 1 - not Adult. It has been dropped.
    startYear : release year of the title. It has been dropped.
    endYear : TV series end year. It has been dropped.
    runtimeMinutes : primary runtime of title. It has been dropped.
    genres : includes three genres.

  – *title.principals* : This dataset contains the principal cast and crew for titles.

    ordering : a number to uniquely identify rows for a given title. It has been dropped.
    nconst : unique identifier of person/name.
    category : category of job that person was in.
    job : the specific job title, if applicable.
    characters : the name of the character played.

2 DATA ORGANIZATION

Data organisation is a method used to classify and organise the datasets to make them more useful. In this assignment, we have used the Spark DataFrame API for loading the data, preprocessing and building the detector of similar items. Spark DataFrame is a dataset organized into named columns, conceptually equivalent to a table in a relational database or a data frame used in by programs such as R or Python. Spark SQL is a module for structured data processing, that provides information about the structure of both the data and the computation that is being performed, including the

ability to join RDDs, represent a collection of items distributed across many compute nodes that can be manipulated in parallel, and SQL tables. There are several ways to interact with Spark SQL, including SQL, and Dataset API, and when computing a result, the same execution engine is used, independent of which API/language is used to express the computation. This unification means that developers can easily switch back and forth between different APIs based on which provides the most natural way to express a given transformation, meaning that when running SQL from within another programming language, like it will be done on this project, the results will be returned in the form of a Dataset or DataFrame. We have also used python in spark architecture to implement apriori algorithm.

3 DATA PRE-PROCESSING

Data cleaning is the first step and also one of the most important in any data science project. The main ideology of this assignment is to implement a system finding frequent itemsets, by considering movies as baskets and actors as items. From the above mentioned datasets, We read the first dataset (name.basics.tsv) to extract the information consisting of actors and remove missing values and select the actors whose primary profession is actor or actress. We read the other datasets (title.principals.tsv) and (title.basics.tsv) which consist of information about the movies related to actors and the actors respectively. From these datasets two extra tables have been extracted. The main table join information contains the correspondence between movieId's and actorId's and titles of the movies and names of the actor's. It is coordinated as follows:

movie id — movie title — actor id — actor name

The second table basket movies is considered by grouping by the movieId's and also eliminating the duplicates.

movie — [actor1, actor2, ...]

The factors that are considered for the assignment are the ID of the movies and the ID of the actors, since they are identifiers. All the datasets have been isolated into 10 partitions for dealing with enormous information. In this manner the work is distributed to various groups and parallelized.

4 Implementation of the algorithms

In order to achieve the frequent itemsets, we consider Apriori Algorithm.

4.1 Apriori Algorithm

The A-Priori Algorithm is designed to reduce the number of pairs that must be counted, at the expense of performing two passes over data, rather than one pass.

The First Pass of A-Priori : In the first pass, we create two tables. The first table, if necessary, translates item names into integers from 1 to n. The other table is an array of counts the ith array element counts the occurrences of the item numbered i. Initially, the counts for all the items are 0. As we read baskets, we look at each item in the basket and translate its name into an integer. We use that integer to index into the array of counts, and we add 1 to the integer found there.

Between the Passes of A-Priori : After the first pass, we examine the counts of the items to determine which of them are frequent as singletons. We set the threshold s sufficiently high that

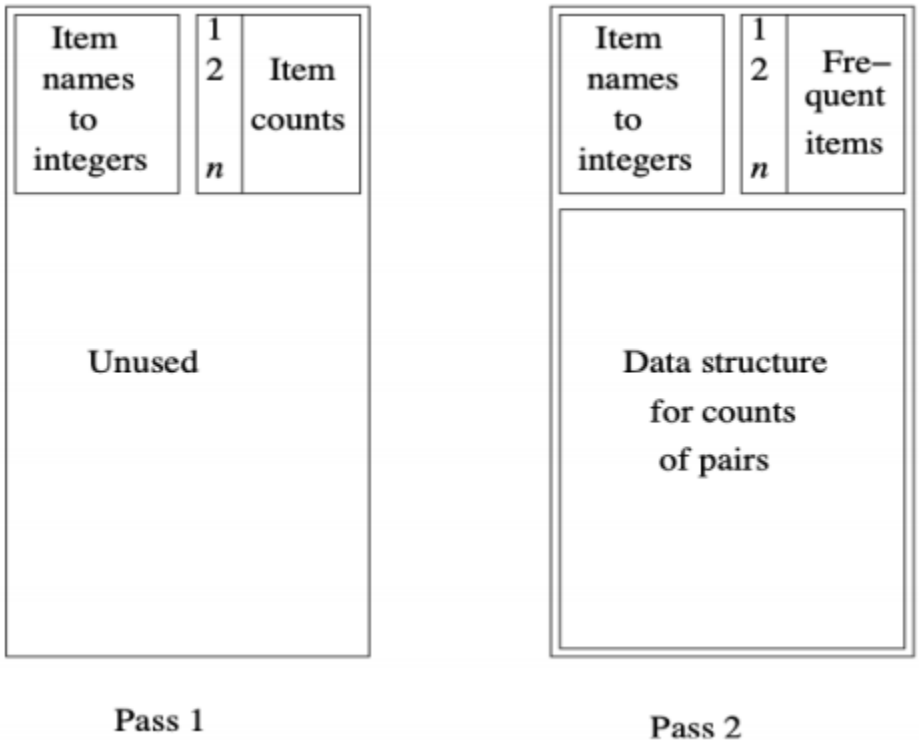
we do not get too many frequent sets; a typical s would be 1% of the baskets. For the second pass of A-Priori, we create a new numbering from 1 to m for just the frequent items. This table is an array indexed 1 to n, and the entry for i is either 0, if item i is not frequent, or a unique integer in the range 1 to m if item i is frequent. We shall refer to this table as the frequent-items table.

The Second Pass of A-Priori : During the second pass, we count all the pairs that consist of two frequent items. Consider that a pair cannot be frequent unless both its members are frequent. Thus, we miss no frequent pairs.

The mechanics of the second pass are as follows.

- For each basket, look in the frequent-items table to see which of its items are frequent.
- In a double loop, generate all pairs of frequent items in that basket.
- For each such pair, add one to its count in the data structure used to store counts.

Finally, at the end of the second pass, examine the structure of counts to determine which pairs are frequent.



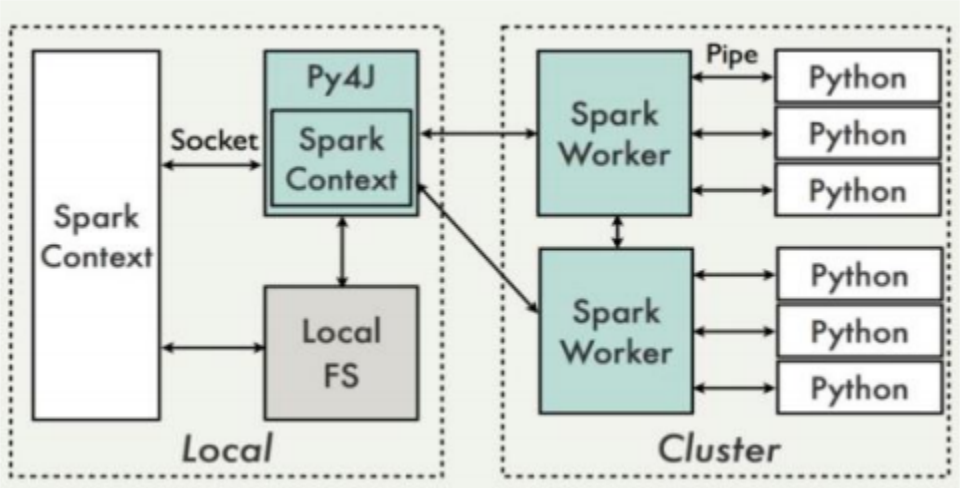
The same technique used for finding frequent pairs without counting all pairs lets us find larger frequent itemsets without an exhaustive count of all sets. In the A-Priori Algorithm, one pass is taken for each set-size k. If no frequent itemsets of a certain size are found, then monotonicity tells us there can be no larger frequent itemsets, so we can stop.

The pattern of moving from one size k to the next size k + 1 can be sum- marized as follows. For each size k, there are two sets of itemsets,

- Ck is the set of candidate itemsets of size k – the itemsets that we must count in order to determine whether they are in fact frequent.
- Lk is the set of truly frequent itemsets of size k

5 Scalability

Spark can load data into memory on the worker nodes, many distributed computa- tions, even ones that process terabytes of data across dozens of machines, can run in a few seconds. This makes the sort of iterative, ad hoc, and exploratory analysis commonly done in shells a good fit for Spark. Spark provides both Python and Scala shells that have been augmented to support connecting to a cluster. Our detector is fast and efficient and scales with the data size. This is because of the usage of Spark, which uses Regular expressions and Spark DataFrame based API, and the default datatype is "String", which occupies relatively very less memory. Apache Spark is a unified analytics engine for large-scale data processing. It achieves high performance for both batch and streaming data, using a state-of-the-art DAG scheduler, a query optimizer, and a physical execution engine. Spark's Catalyst op- timizer converts user inputs into a Directed Acyclic Graph (DAG) of transformations before distributing the optimized tasks across a network of computers for in-memory processing. Hence, the reason, the data cleaning was done in seconds.



A Resilient Distributed Dataset (RDD), the basic abstraction in Spark. Represents an im- mutable, partitioned collection of elements that can be operated on in parallel. This class contains the basic operations available on all RDDs, such as map, filter, and persist. RDD transformation that is used to apply the transformation function (lambda) on every element of RDD/DataFrame and returns a new RDD.

6 Description of experiments

The objective of the assignment is to recovery requent itemsets. Because of the huge computational expense of the Apriori when managing enormous datasets, the calculation has been considered on a subset of data by movies having the Musical class. The subset contains 35.111 baskets rather than the whole number of baskets in the dataset which is around multiple times greater, to be specific 3.531.063. We define an algorithm for getting the frequent itemsets using Apriori. Implementation of APriori algorithm for returning frequent itemsets. We start by initializing the class and define the functions for carrying out the process of frequent itemsets. Returns baskets in format [[item1,item2,...][item1,item2,...]]. Returns a list of frequent singletons [item1,item2,...]. Return candidate itemsets for itemsets with 'i' items by taking the cartesian product. Returns frequent itemsets by initializing the list of frequent itemsets and retrieving the frequent pairs, if candidate is present in the basket then we add 1 to the counter and retrieve the frequent itemsets from candidate sets. We follow the same process to get the frequent itemsets with items greater than three. Run the algorithm for minsupport as 0.0006 and also considering the runtime of the algorithm.

7 Results and Discussion

The runtime of the Apriori algorithm to get the frequent itemsets for the subset of data 'MUSICAL' is roughly 1.5 hours. The results are as follows:

items	freq
[Thomas Sanders]	1887
[Manuela do Monte]	472
[Rayssa Chaddad, Giovanna Grigio]	471
[Manuela do Monte, Rayssa Chaddad, Giovanna Grigio, Felipe Cavalcanti]	471
[Rayssa Chaddad, Felipe Cavalcanti]	471
[Rayssa Chaddad]	471
[Manuela do Monte, Rayssa Chaddad, Felipe Cavalcanti]	471
[Manuela do Monte, Rayssa Chaddad]	471
[Rayssa Chaddad, Giovanna Grigio, Felipe Cavalcanti]	471
[Giovanna Grigio, Felipe Cavalcanti]	471
[Manuela do Monte, Rayssa Chaddad, Giovanna Grigio]	471
[Felipe Cavalcanti]	471
[Manuela do Monte, Giovanna Grigio, Felipe Cavalcanti]	471
[Manuela do Monte, Giovanna Grigio]	471
[Manuela do Monte, Felipe Cavalcanti]	471
[Giovanna Grigio]	471
[Fernando Soberanes]	402
[Maisa Silva]	390
[Jean Paulo Campos]	390
[Maisa Silva, Jean Paulo Campos]	390

References

[1] J. Leskovec, A. Rajaraman, and J. D. Ullman, Mining of massive datasets. Cam- bridge University Press, 3 ed., 2014.

[2] Kaggle, "Dataset from IMDb to make a recommendation system". Retrieved from Kaggle, https://www.kaggle.com/ashirwadsangwan/imdb-dataset.

[3] Apache.org, "RDD Programming Guide". Retrieved from Apache.org, https://spark.apache.org/docs/latest/rdd-programming-guide.html.

[4] Apache.org, "RDD Programming Guide". Retrieved from Apache.org, https://spark.apache.org/docs/latest/ml-frequent-pattern-mining.html.

[5] Towards Data Science,"Big Data Market Basket Analysis". Retrieved from Towards Data Science, https://towardsdatascience.com/big-data-market-basket-analysis-with-apriori-algorithmon-spark-9ab094b5ac2c.