MSc in Data Science and Economics
Department of Economics, Management and Quantitative Methods
Università degli Studi di Milano

Market Basket Analysis-MeDal

Università degli Studi di Milano
Algorithms for massive data, cloud and distributed computing
JAMAL HIND – 989432

a.y. 2022/2023

I declare that this material, which I now submit for assessment, is entirely my 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 work. I 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 or any other person for assessment on this or any other course of study.

Index

1 Abstract

The main aim of this project consists on the possibility to apply on a MeDal dataset some data mining techniques, in order to perform a Market-Basket analysis.
The algorithms were implemented using the Apache Spark computational framework, while the code was run on Google Colab.
In particular, considering our dataset, texts are identified as baskets while the respective "words" as items, and the purpose of this analysis is to output only the frequent item sets, so frequent pairs of words we have per text.
For this aim we use the FP-Growth and the A-Priori algorithms.
After computing the frequent pairs and their relative frequencies (obviously obtaining the same result with both algorithms), we are going to inspect the support of frequent items, in order to output the confidence of the respective association rule. In fact, frequent item sets are useful to build meaningful association rule, from which one can infer that the presence of some items imply the presence of others.
In the design of the algorithms, the most typical Spark transformations and actions (e.g. 'map', 'flatMap', 'reduce', 'reduceByKey' etc.) were used. In the next section, a description of the two techniques used and their implementation will be presented.

2 Introduction

As we already said in the abstract, this project's aim consists on describing and applying different techniques that are generally used to conduct market-basket analysis over huge datasets, in order to find frequent itemset.
In this specific case, the dataset is taken from the public repository of Kaggle and it contains more than 14 million rows. The folder is composed of different files representing a large dataset offering abbreviations and descriptions of all medical terms, and designed in a natural language understanding for pre-training in the medical domain. The MeDal (Medical Dataset for Abbreviation Disambiguation for Natural Language Understanding) dataset was published at the ClinicalNLP workshop at EMNLP. Actually, for the aim of this project, we are focusing on just one Excel file, which is the "full_data.csv" file, with 14 391 698 rows and 3 columns.
In the first column, nominated "TEXT", we have the explanation of the medical terms and abbreviations; the second one is the "LOCATION"; and the last one is the "LABEL", so one or two words used to identify a medical term. The information that we will use to develop our analysis are related to the "LABEL" and the "TEXT" column, focusing only on texts with the label study on them.
Although in this setting only a part of the data has been examined, the processing has been implemented to be suitable for any size of data, so for large-scale data.
To find frequent itemset we implemented different algorithms with the aid of the Map Reduce programming model.

Before applying the two algorithms we need to conduct a pre-processing phase, used to normalize the plain text of our dataset, providing a complete and cleaned input to the Market Basket Analysis algorithms.

2.1 Data pre-processing

The file of our interest can be retrieved via Kaggle API. To download the MeDal dataset we therefore need to use the relative command, which consists on putting as an input the exact name of the resource, which in our case is xhlulu/medal-emnlp.

```
!kaggle datasets download -d  xhlulu/medal-emnlp
```

We will therefore unzip the file and before implementing and creating our algorithms, used to find frequent item sets, we need to convert our data and present them as a data frame.
As a first step then we have to create the framework that enables us to perform parallel processing, and that is a SparkContext environment.

One of the most useful and interesting pieces of information that we can extract from our database consists on the frequency of the different labels describing each of our Medical definition. As we can see in figure 2.1, the most frequent ones appear with the words study, after, factors, development or cancer. We will filter and focus therefore our analysis on the texts we have related to the "study" label, which contain a total of 294,977 rows.

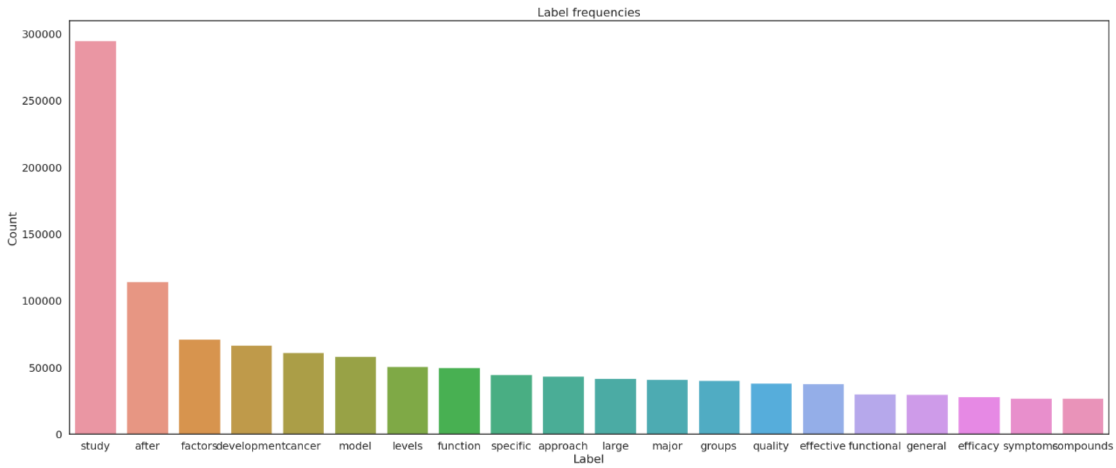

Figure 2.1. “LABEL” frequency-words.

As previously stated, our interest consists in finding the occurrences of n - items set in our file that satisfy a certain threshold, but before doing this we need to select the

data of interest and execute some kind of functions to clean our data in the best possible way, so that the output is not affected by other additional inputs.
So, once we select the “TEXT” column, we are going to implement different map functions using SparkNLP, an open-source text processing library which integrates with both Python and Spark and works directly on Dataframes.
Thanks to this phase, we are able to create a set of key-value pairs containing words, on which different algorithms for Market-Basket analysis could be applied. These pre-processing steps consist in tokenizer, required to split each text content into smaller shingles (or tokens), which corresponds to single words or punctuation marks; a map function converting each letter of each row in the “TEXT” column in a lower-case; a map function removing punctuation; and a map function removing stop-words, such as conjunctions or articles, which could negatively impact our results. We also need to apply an additional map function, required to remove any kind of words duplication per row.
What we obtain as a result is a column of cleaned words.

Figure 2.2 represents a table composed of three columns, in which we have the processed text, the cleaned-out words and a function counting the number of cleaned out words per row.

processed_text	words_clean	word_count
alphabisabolol ha...	[case, alteration...	26
a report is given...	[pronounced, deve...	123
the virostatic co...	[substance, prese...	33
rmi rmi and rmi a...	[hypotensive, lev...	105
a doubleblind stu...	[clear, obtained,...	93
stroma from eithe...	[lysis, pathway, ...	60
the effect of the...	[pathway, concent...	25
in one experiment...	[feeding, rumens,...	60
the presence of a...	[vmax, bindingdep...	96
the reaction of g...	[step, stable, ad...	81
choline acetyltra...	[using, ionexchan...	76
increasing concen...	[protein, globula...	65
the properties of...	[protein, builtin...	95
primary amines re...	[protein, react, ...	60
a purification pr...	[step, dhf, equil...	91
dihydrofolate red...	[chromatography, ...	115
ionization effect...	[indication, bett...	78
kinetic analyses ...	[dipeptidase, nac...	120
the nearultraviol...	[exhibit, fragmen...	93
the circular pola...	[appears, portion...	99

only showing top 20 rows

Figure 2.2. Complete table analysis.

3 Market-Basket analysis

The Market Basket Analysis is used to describe the relationship we have between two items. In particular, on one hand we have the items, while on the other one we have the baskets, and the Market Basket Analysis tries to understand which items can be considered to be frequent ones, by comparing their number of occurrences to a support threshold s and by making sure that the number of times such items appear in a basket is higher than s.
Note that this kind of analysis is linked to the association rule I → j, where I is a set of items and j is an element, and according to this rule we have that whenever in a basket all the elements in the subset I are contained, also the element j is contained in that basket, so, in other terms, it consists on the probability of j being in the basket with I. The association rule is evaluated related to two measures:

- Confidence. It consists on a fraction between the support of the itemset I and the element j, and the support of the itemset I. Confidence consists therefore on the probability that an element j appears in some baskets, given that subset I is in them.

$$Conf\ (I \to j) = \frac{supp(I\ U\ \{j\})}{supp\ (I)}$$

- Interest. It consists on the difference between the confidence and the fraction of baskets that contain j. If I has no influence on j, then the interest is equal to 0 and the fraction of baskets including I that contain j is the same as the fraction of all the baskets that contain j; if I has a high interest, then the presence of I in a basket encourages the presence of j in that same basket; and if I has a low interest the presence of I in a basket discourages the presence of j in that same basket.

$$Int\ (I \to j) = \ Conf(I \to j) - \frac{supp(\{j\})}{total\ baskets}$$

In order to find frequent itemset, two algorithms have been considered in this project, that obviously give us back the same results.
In particular, we will use Frequent Pattern Growth and A-priori algorithms.

3.1 A-priori

To implement the A-priori algorithm we defined a function considering as an input the RDD of baskets and based on the Map-Reduce programming model.
In particular, we will set each key to 1 and the values will contain a list of words of each text. The basic idea of this algorithms consists on the possibility to avoid computing the frequency of all possible itemsets generated from our dataset, by

creating a set of candidates whose frequency will be checked to avoid false positives.
This approach is in particular facilitated thanks to the monotonicity property, according to which if an itemset is frequent, all its subsets must be frequent as well. Therefore, it cannot exist a frequent itemset generated by items which are not frequent themselves, and thanks to such property we will avoid getting also false negatives.

The Apriori algorithm presents two phases.
First Phase: each item is set with a counter equal to one, so that we are forming some key-value pair (item,1). Then, the algorithm is going to sum the occurrences of each item collected in the first step and keep only the items whose number of occurrences exceeds the threshold.
Second Phase: those words whose count are greater than the support threshold are combined to create new itemset, and this is the set of candidate pairs. For each new pair created the algorithm is going to check if such pairs are included in the basket and in an affirmative case the respective counter will be increased by one. If the resulting counter of a given candidate pair is higher than the support, they are appended to the frequent item list.

3.2 FP-Growth

FP-Growth is a highly developed algorithm that allow us to fit a model on a larger number of observations of a dataset requiring a smaller amount of time if compared to the previous approach.
The FP-Growth algorithm can overcome the two shortcomings we have identified with the Apriori algorithm, which are the extremely large size of the candidate itemset, since this set is actually no longer required for the FP-Growth algorithm; and the high costs given by the fact that with A-priori algorithm we need to scan and do some computations over and over again on the same item sets.
In the first step this algorithm computes the frequency of the items and compare it with a minimum threshold; in the second step the algorithm creates a data structure called frequent pattern tree, where only the frequent singletons and their count is stored.

4 Results

As previously stated, the two different methods adopted to perform the Market-Basket Analysis give us back the same result.

items	freq
[t0]	294977
[patients]	58487
[patients, t0]	58487
[aim]	46949
[aim, t0]	46949
[results]	45547
[results, t0]	45547
[study]	42329
[study, t0]	42329
[using]	40581
[using, t0]	40581
[present]	38754
[present, t0]	38754
[used]	38166
[used, t0]	38166
[may]	31004
[may, t0]	31004
[two]	30838
[two, t0]	30838
[also]	30148

items	freq
[patients, t0]	58487
[aim, t0]	46949
[results, t0]	45547
[study, t0]	42329
[using, t0]	40581
[present, t0]	38754
[used, t0]	38166
[may, t0]	31004
[two, t0]	30838
[also, t0]	30148
[data, t0]	29668

Figure 4.1. Frequent singletons and pairs of items.

The output from the FP Growth Algorithm is represented in figure 4.1 and 4.2. These tables show the frequent item sets and their corresponding frequencies, which are computed using a minimum support threshold of 0.1.

antecedent	consequent	confidence	lift	support
[using]	[t0]	1.0	1.0	0.13757343792905888
[used]	[t0]	1.0	1.0	0.12938635893645944
[may]	[t0]	1.0	1.0	0.10510649982880022
[two]	[t0]	1.0	1.0	0.10454374408852216
[data]	[t0]	1.0	1.0	0.10057733314800815
[study]	[t0]	1.0	1.0	0.14349932367608323
[present]	[t0]	1.0	1.0	0.1313797346911793
[t0]	[may]	0.10510649982880022	1.0	0.10510649982880022
[t0]	[data]	0.10057733314800815	1.0	0.10057733314800815
[t0]	[aim]	0.15916156174888213	1.0	0.15916156174888213
[t0]	[using]	0.13757343792905888	1.0	0.13757343792905888
[t0]	[study]	0.14349932367608323	1.0	0.14349932367608323
[t0]	[used]	0.12938635893645944	1.0	0.12938635893645944
[t0]	[patients]	0.19827647579302793	1.0	0.19827647579302793
[t0]	[also]	0.10220457866206518	1.0	0.10220457866206518
[t0]	[two]	0.10454374408852216	1.0	0.10454374408852216
[t0]	[results]	0.1544086488099072	1.0	0.1544086488099072
[t0]	[present]	0.1313797346911793	1.0	0.1313797346911793
[also]	[t0]	1.0	1.0	0.10220457866206518
[patients]	[t0]	1.0	1.0	0.19827647579302793

Figure 4.2. Association rule.

The table above allow us to obtain more information about the relationships between different items in the same dataset.
In particular, such representation shows us the association rule generated by the FP Growth algorithm, including the antecedent (items that occur before) and the relative consequent (items that occur after).
Each item is also described by specific measures, like the confidence, which is the number of baskets that contain the antecedent item set; the lift, which measures the strength of the association between the antecedent and the subsequent item, compared to what would be expected; and the support, which measures the frequency of occurrences of the item sets in baskets.

We then applied the Apriori Algorithm. In the following table we are showing the items with a frequency higher than the support threshold s.

```
[('t0', 294977),
 ('used', 38166),
 ('two', 30838),
 ('also', 30148),
 ('study', 42329),
 ('using', 40581),
 ('patients', 58487),
 ('results', 45547),
 ('data', 29668),
 ('may', 31004),
 ('present', 38754),
 ('aim', 46949)]
```

Figure 4.3. Frequent singletons.

Given such singletons, what we can do is to proceed to the second phase of our a-priori algorithm, according to which frequent items must be combined the one with the other to create new itemsets, known as the set of candidate pairs. Then, the algorithm will check for the threshold, to state whether such pairs can be defined as frequent ones or not.

```
[(('study', 't0'), 42329),
 (('also', 't0'), 30148),
 (('data', 't0'), 29668),
 (('aim', 't0'), 46949),
 (('present', 't0'), 38754)]
```

Figure 4.4. Frequent pairs.

In both algorithms, there are no triple items with high frequency, maybe because the value of the support threshold is set to a large value with respect to the frequencies that we have, and therefore the algorithms cannot return back frequent triples or pairs with a number of occurrences lower than the threshold value s (equal to 29498). Another possible motivation can be the fact that we are not considering the whole dataset, but only the texts with a label equal to “study”. We have scalability, and therefore we can affirm with certainty that there is the possibility to perform such analysis on the whole dataset, without any kind of issues from the RAM or the memory of our notebook on Google Colab.

5 Scalability

Our codes and results were all implemented by having always in mind scalability, since the dataset we are referring to is a real-world scenario, with extremely large and big amount of data.
In the pre-processing, A-priori and FP-Growth algorithms implementation we made great use of the Spark's rdd data structure and functions and also of the Map and Reduce functions. Thanks to this we have a parallel and distributed computation, allowing us to reach a good degree of scalability and to process our data without any kind of issue.

6 Conclusion

In this paper we just presented a few of the different approaches that could be used to deal with frequent item sets, but it is important to keep in mind that we still have many other algorithms and functions that could perform in a different or better way. Comparing the results of the two algorithms, the outputs are basically the same. We can therefore conclude that, despite the fact that the two algorithms computing frequent item sets are using different methods, since the outputs are equal the results of the two algorithms can be considered reliable.
However, because of the higher amount of computations required by the A-priori algorithm, the amount of time spent is higher with respect to the one we need for the FP-Growth algorithm. This last one is also able to avoid the issues related to the A-priori algorithm presented before, giving back the same identical result.

7 Bibliography

https://towardsdatascience.com/market-basket-analysis-using-pysparks-fpgrowth-55c37ebd95c0

https://towardsdatascience.com/fp-growth-frequent-pattern-generation-in-data-mining-with-python-implementation-244e561ab1c3

Mustakim, D. Herianda, and A. Ilham. Market basket analysis using apriori and fp-growth for analysis consumer expenditure patterns at berkah mart in pekanbaru riau. Journal of Physics, pages 1–10, 2018.