Algorithms for Massive Data Project

Loris Bartesaghi. (Mat. 967378)

December 2022

1 Introduction.

The aim of this project is to implement market basket analysis on the "Old Newspaper" dataset from Kaggle using the FPGrowth and the Apriori Algorithm.
The market basket analysis consists in finding frequent itemsets, which are the items appearing together in the same baskets at least a sufficiently large number of times. For this analysis, the whole textual content are considered as baskets and the single words as items.

Market basket analyses is usually performed in order to understand what products customers usually buy together and these informations could be very beneficial for several reasons:

- predict the rise or fall in demand for one product by the price change of another
- improve recommendation systems
- improve marketing offers

Sales purposes are not the only field on which market basket analysis is applied. It can be also used in another context as well, as in text analysis. In this paper I'm going to find the couple of words that appear together most of the time in the text of newspapers, social media posts or blogs.
Performing market basket analyses, the analyst is typically confronted with a lot of different items and itemsets that should be analyzed for finding useful information. This abundance of data imposes a restriction on using naïve approaches like cycling over each item and each itemset to find frequent joint appearances. In case of huge number of documents and huge number of words, computation time and computing capacity could become a problem.
In this paper, I'm going to apply the Apriori algorithm, to find the most frequent pairs of words in the text. For computing reason only 10.000 observations are considered for the analysis.
Furthermore, FPGrowth is applied. It is a developed algorithms that allow to fit a model on a larger number of observation of the dataset in a reasonable time. The algorithm will be also run on a random 10% of the total english text and the result will be compared.

The link to the github repository can be found here: GITHUB
The link to the colab notebook can be found here: COLAB

2 Data.

The "Old Newspaper" dataset contains natural language text from various newspapers, social media posts and blog pages in multiple languages.
The format of the file imported in the process is a tsv file and it will be imported directly in the colab notebook through the code
The columns of each row are:

- Langauge: Language of the text.
- Source: Newspaper from which the text is from.
- Date: Date of the article that contains the text.
- Text: Sentence/paragraph from the newspaper.

  The corpus contains 16,806,041 sentences/paragraphs in 67 languages. For the project only the observations containing english text will be considered and only the column containing the text is necessary for the aim of the project.

2.1 Data Pre-processing.

The first step in the project is to apply several map functions to create a set of key-value pairs containing words, on which different algorithms for market basket analysis could be applied. The pre-processing steps applied on the dataset are the following:

- Selection of english sentences or paragraph from the newspaper;
- Removing punctuation and number;
- Removing sequence of letters that could lead to poor result of out analysis (e.g. "p.m.", "a.m.");
- Remove stopwords.

2.2 Data Exploration.

The next step is to extract information about our dataset. The most useful informations are obviously about the frequence of each word, the total number of words contained in our analysis, the mean value of the frequence of the words and the maximum and minimum value of the less and most frequent word. These kind of information will be very useful to set realistic parameters for the algorithms.
The different words present in the analysis are 272,019 with a total count of words of 18,446,087. The word with the highest frequence is "said", which is present 227,872 times in our observations. The lowest frequence is one, for many different words.

The following histogram shows the frequence of the 10 most frequent words.

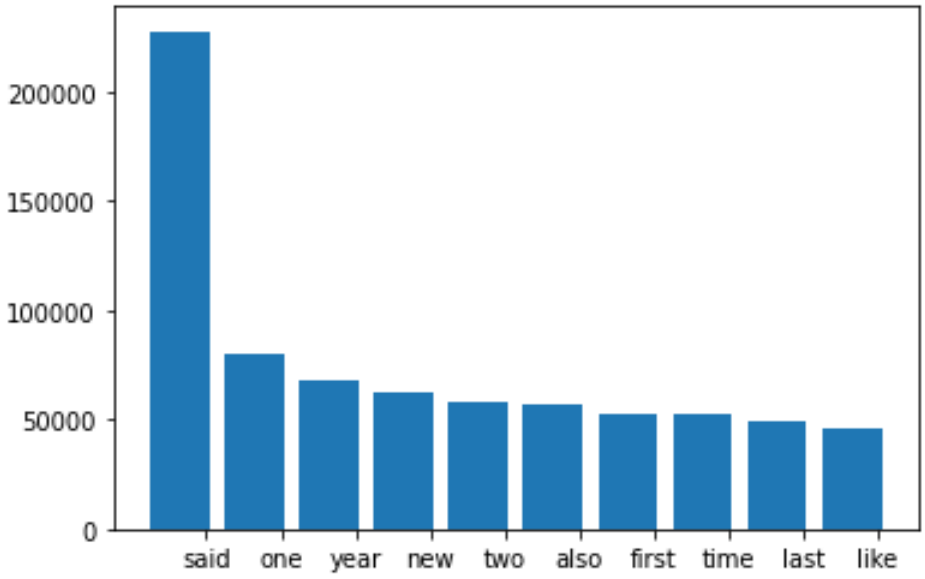

Figure 1: Word Frequnce

The image shows us that the most frequent words are usually used to report news. For example: "The president said..", "The hottest year in history.." or "Today, the Prime Minister will meet the Parliement for the first time..". And, considering the general arguments of the resources, there are no related-topic words.

3 Algorithms

In order to find the frequent itemsets, two algorithms have been considered in this project. In particular, we will use FPGrowth and Apriori algorithms.

3.1 Apriori.

Apriori algorithm has two phases.The input are the basket of items along with a support threshold. This algorithm exploits the monotonicity property that affirm that the support of an itemset cannot be greater than the support of its subsets. This implies that all the subsets in a itemset should be frequent as well. This clear approach allows the Apriori to reduce number of possible items to be evaluated. The algorithm uses the following metric to decide whether an itemset is frequent or not :

$$supp(j) = \frac{freq(j)}{|baskets|} > threshold$$

First Pass.

- Each item is initialized with a counter equal to one. key-value pair:(item,1).
- Sum the occurrences of each item collected in the first step. key-value pair:(item, number of occurrences)
- Keep only the items where number of occurrences > threshold.

Second Pass

- Those words whose count are greater than the support are combined to create new itemsets. These are the candidate pairs.
- For each new pair created in the previous point, the algorithm checks if such pairs are included in the basket, if so, the corresponding counter is increased by one.
- If the resulted 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 algorithm takes as input the items in a basket along with a user-defined minimum threshold value.In the first step, it calculates the frequency of items and checks them against the minimum threshold. In the second step, it creates a data structure called frequent pattern tree where only the frequent singletons and their count is stored. Every time a pattern is found, the corresponding item's count is increased. After the second step, the frequent items can be extracted from the tree.

4 Results.

The first algorithm applied on the data is the Apriori Algorithm. I decide to use a threshold of 0.005, and then select only the itemsets with at least two words. In the following table are shown the 10 itemsets with the highest values for the support.

Itemset	Support
said, one	0.0181
people, said	0.0163
year, last	0.0143
going, said	0.0141
said, get	0.0132
said, us	0.0131
like, said	0.0123
new, said	0.0118
said, time	0.0116
year, said	0.0115

It is possible to notice that the words in the itemsets are also present in the most frequent words in the whole dataset. This result is expected, indeed it is reasonable to find frequent pairs that are composed by frequent words. An improvement of the algorithm taken from the library *mlxtend* could be the introduction of the measures *confidence* and *interest* which allow to find the itemsets which are connected by a strong relation.

The second algorithm applied is the FPGrowth Algorithm. The threshold is equal to the threshold applied to the Apriori algorithm. In the following table are shown the 10 itemsets (with at least two words) with the highest values for the support.

Itemset	Support
said, one	0.0168
people, said	0.0146
said, get	0.0145
year, last	0.0144
going, said	0.0139
like, said	0.0131
said, new	0.0123
said, time	0.0116
us, said	0.0116
also, said	0.0112

Comparing the result of the two algorithm, most of the pairs are in both the output. From these result we can conclude that, despite the two algorithm compute frequent itemset using different methods, the output are similar and thus the result of the two algorithm could be considered reliable. Also in this case, the package *pyspark.ml* doesn't offer an option to include in the analysis the values of *confidence* and *interest*.

5 Conclusion.

The huge amount of data produced in the world every second is astonishing, and new methods for storing and process data have been introduced. For this project, for example, Google Colab allowed me to have a platform ready to use, without the need to install packages or download data on personal laptop and Pyspark allowed me to create processes, written in python, that could be run on several hardware and, thus, to make analysis on very huge amount of data.
The output of the two different algorithm for finding frequent itemsets are similar and this suggest us that the final result is reliable. More important, the process used to produced the results is scalable. The limit of this project is the lack of the values of *confidence* and *interest* that could improve the results of the analysis. Future improvement could be the introduction of function that, starting from the supports computed from Apriori and FPGrowth, return values of *confidence* and *interest*.

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.