MapReduce

CAIM Lab, Session 6

Oscar Mañas, Julián Suárez

About the implementation

Our implementation of the algorithm follows the structure suggested in the statement, but we write a single program containing all parts.

We start by reading the input file `groceries.csv` and (implicitly) create the collection `transactions`, where each document is associated to a transaction. That way, each document holds the set of items involved in that transaction.

Next, we write the javascript code that defines the mappers and reducers. We need to count in how many transactions each item appears, and also in how many transaction pairs of items appear. In order to accomplish that, we can use the same reducer for both counts, since the work of the reducer is just adding the frequency values that each mapper has output for a given key. One thing to note is that for each pair of items we only have one key, that is, we consider that $X \rightarrow Y$ and $Y \rightarrow X$ are the same. The criterion we follow is always construct the pairs in lexicographical order. We will take that into account when computing the support and confidence measures.

We should also point out that we had minor problems with the `mapperPairs` function. At first, instead of assigning a value to the `key` variable depending the result of the `localCompare` function, we did a conditional `emit` based on the same result. That is, we had one `emit` inside an `if` clause and another `emit` inside an `else` clause. For some reason, when we ran our code the computer freezed and we had to reboot it several times. After doing some tweaks to the `mapperPairs` code, we found that doing this change solved our problem.

Finally, we use the counts of both mapreduce operations, stored in the `item_counts` and `pair_counts` documents, to compute the values of support and confidence for each pair of items. As we pointed out earlier, when counting pairs of items we consider $X \rightarrow Y$ and $Y \rightarrow X$ to be equivalent, but actually we have that $conf(X \rightarrow Y) \neq conf(Y \rightarrow X)$. So for each element (X, Y) of `pair_counts` we extract the pair $(X \rightarrow Y)$ and the pair $(Y \rightarrow X)$, and compute support and confidence values for each pair. For each pair whose values of support and confidence pass the given thresholds, we add a new association rule.

Results

Once we finished the script, we ran it several times to be able to fill out the table below. For each pair of support/confidence thresholds, we count how many association rules pass the test, i.e., their support and confidence values are equal or greater than the threshold values.

Table of results:

Row	*Support*	*Confidence*	*Nr. of association rules found*
1	1%	1%	426
2	1%	25%	96
3	1%	50%	0
4	1%	75%	0
5	5%	25%	4
6	7%	25%	2
7	20%	25%	0
8	50%	25%	0

Below we give the list of association rules found corresponding to rows 4, 5, and 6 of the previous table:

```
Row : 4  |  Support : 1 %  |  Confidence : 75 %  |  Nr. of association rules found : 0

-

Row : 5  |  Support : 5 %  |  Confidence : 25 %  |  Nr. of association rules found : 4

other vegetables -> whole milk       : sup = 0.0748347737672 , conf = 0.38675775092
whole milk       -> other vegetables : sup = 0.0748347737672 , conf = 0.292877039395
rolls/buns       -> whole milk       : sup = 0.0566344687341 , conf = 0.307904919845
yogurt           -> whole milk       : sup = 0.0560244026436 , conf = 0.401603498542

Row : 6  |  Support : 7 %  |  Confidence : 25 %  |  Nr. of association rules found : 2

other vegetables -> whole milk       : sup = 0.0748347737672 , conf = 0.38675775092
whole milk       -> other vegetables : sup = 0.0748347737672 , conf = 0.292877039395
```

From the previous results we observe:

1. The support is not usually very high for any pair of products. This means that there are no “universal” pairs that appear in a high percentage of transactions.
2. Given a rule $X \to Y$ with a certain confidence threshold, it does not necessarily satisfy the same with the rule $Y \to X$ and the same threshold.

3. For any association rule, the value of confidence never surpasses 50%. If we have the association rule $X \rightarrow Y$, this means that if we buy product *X* we have less than a 50% chance of buying product *Y*.

4. The pair (other vegetables, whole milk) appears in a 7,5% of the transactions. If we buy other vegetables, there's a chance of 38,7% that we also buy whole milk. On the other hand, if we buy whole milk, there's a chance of 29,3% that we also buy other vegetables.

From these results we can conclude that whole milk and other vegetables are very related products: maybe because they are both basic products? Also, 30% of people who buy rolls/buns also buy whole milk: maybe they are looking for something to pair with the buns? Finally, 40% of people who buy yogurt also buy whole milk: maybe milk and yogurts are placed together in the supermarket since both are dairy products?

Association rule mining seems like a very useful technique for discovering interesting relations between variables in large databases. In the particular case of market basket analysis, for example, it could help supermarket owners distributing their groceries in the store, placing together those products that appear in significant association rules.