Data Mining

Assignment 2: Recommendations using non-derivable association rules

1 Ma INF 2022-2023

Name: Thomas Van Daele
Student ID: 20195723
e-mail: Thomas.VanDaele@student.uantwerpen.be
Github: https://github.com/thdaele/Data-Mining

June 26, 2023

1 Apriori

The first important part to notice about the apriori implementation of GPT4 is that it doesn't use monotonicity, without this property the implementation of apriori is incorrect. So instead of only generating candidate itemsets (an itemset is called a candidate itemset if all of its subsets are known to be frequent), it will generate all possible itemsets. I fixed it on line 45 in the file Apriori.py, here I enforce that the candidate itemsets of length k will be generated instead of all itemsets of length k.

I also noticed that the join_set function can be further optimized, in class we saw the following: for all itemsets X, Y with X[:-1]=Y[:-1], X + Y[-1:] is a candidate itemset. The method create by GPT4 will create more candidate itemsets from which not all subsets are actually frequent. For example: {1, 3} and {2, 3} would generate candidate itemset {1, 2, 3} by implementation of GPT4, for this itemset not all subsets are frequent. The method proposed in class would not generate this candidate itemset, I didn't implement this because it requires indexing. Python sets don't support indexing, it could be implemented by only using set operations, but this would make it a lot slower.

I also did some profiling on the GPT4 implementation, if we want to further improve its performance we need to optimize counting the support for each candidate itemset using the transaction database. Almost 100% of the time is spent in the following lines of code, if we run the apriori algorithm on the retail database.

```
for transaction in transactions:
    for itemset in itemsets:
        if itemset.issubset(transaction):
            support_count[itemset] += 1
```

2 Ranking methods

For the rest of the assignment, I continued with using the c++ apriori implementation of Bart Goethals.
I started by splitting the retail dataset into a train and test dataset, for this I used an 85/15 split. In test dataset, I did split the transactions, I take out 5 items as ground truth items. The rest of the test dataset is used for generating the recommendations. I also enforce a length of 10 on the transactions of the test dataset to make sure we have enough items to generate recommendations from.

I implemented the following rankers:

- average confidence, ranks by confidence with tiebreaker on support
- average support, ranks by support with tiebreaker on confidence

- total confidence
- total support
- weighted confidence
- weighted support
- rank on the amount of rules
- popularity
- rank by lift

I ran a parameter grid over a range of different supports, confidences, top n and the different ranking methods. Later on I also added the use of non-derivable itemsets into the parameter grid, this way I could easily compare the evaluation metrics between not using ndi and using ndi.

Changing the rankers didn't result in big improvements, however there were a few small observations that I could make.
First thing that I noticed in the results is that the ranker using the lift has the best score for low support and confidence (see figure 1). Another thing that we could notice is that the average support had very poor results for low support and confidence (see figure 2). All the other rankers performed very similar to each other, with just small variations in the evaluation metrics (see figure 3 and 4).

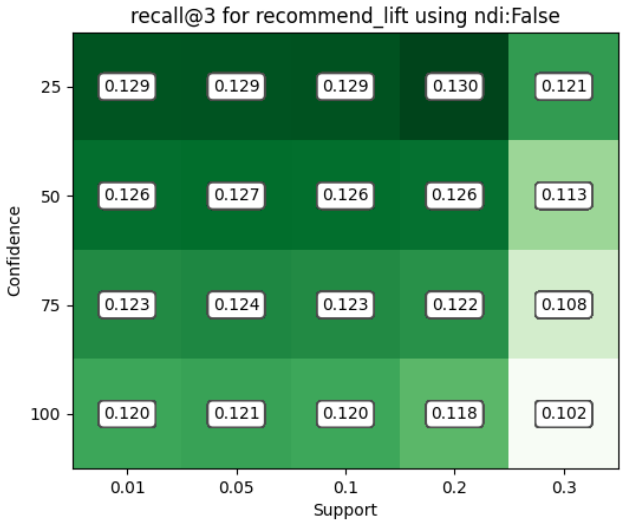

Figure 1: Recall@3 for lift

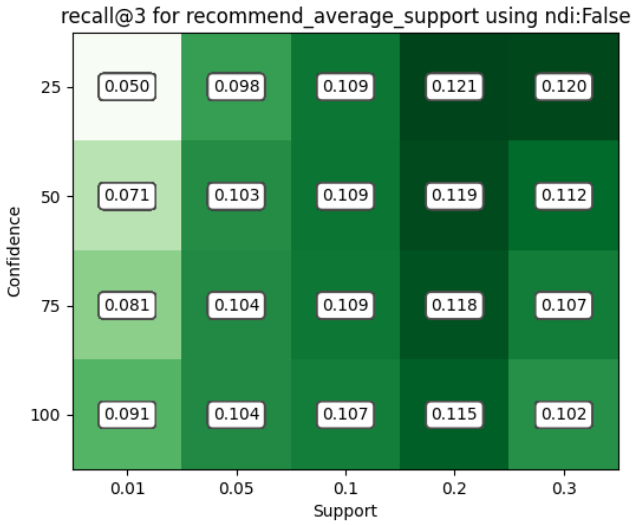

Figure 2: Recall@3 for average support

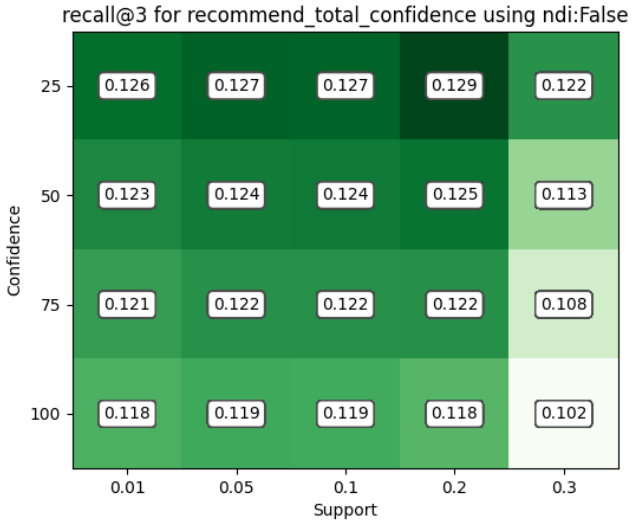

Figure 3: Recall@3 for total confidence

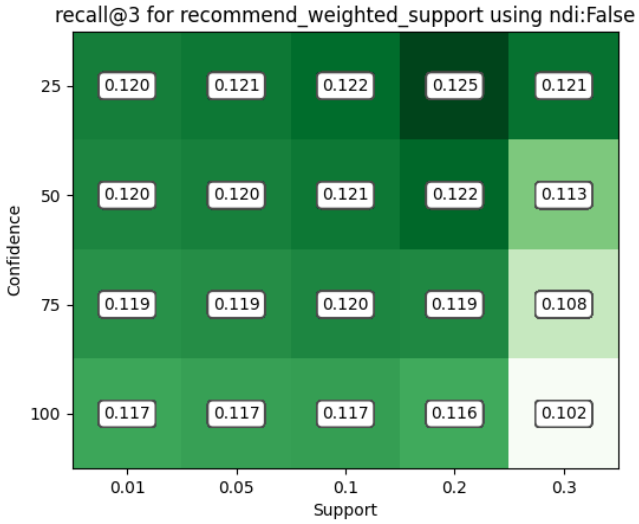

Figure 4: Recall@3 for weighted support

3 Non-derivable itemsets

Using the non-derivable itemsets to generate association rules instead of using the output of apriori to generate association rules didn't result in a big improvement. For some rankers, we could notice a slight increase in the evaluation metrics. The slight increase was mostly in the lower support and confidence counts.

In figure 5 we can see the recall@3 metric for average support ranker used on association rules made from the output of apriori, when compared to figure 6, notice the small increase in the recall for low confidence and support.
Between figure 7 and figure 8, we couldn't notice a difference in the evaluation metric. In this case, non-derivable itemsets don't improve the results of our recommender.

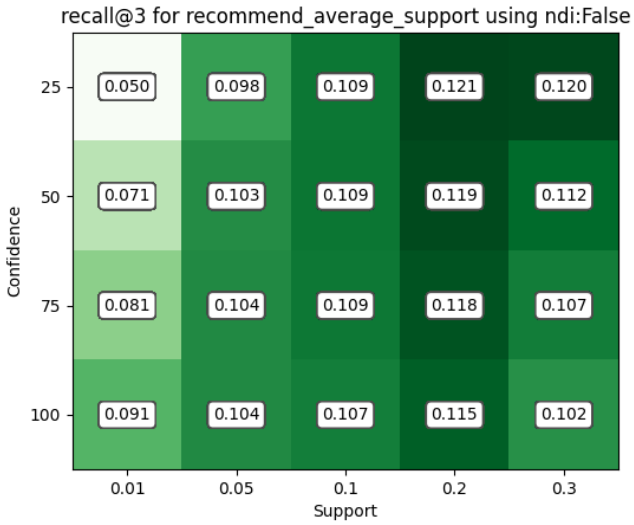

Figure 5: Recall@3 for average support without ndi

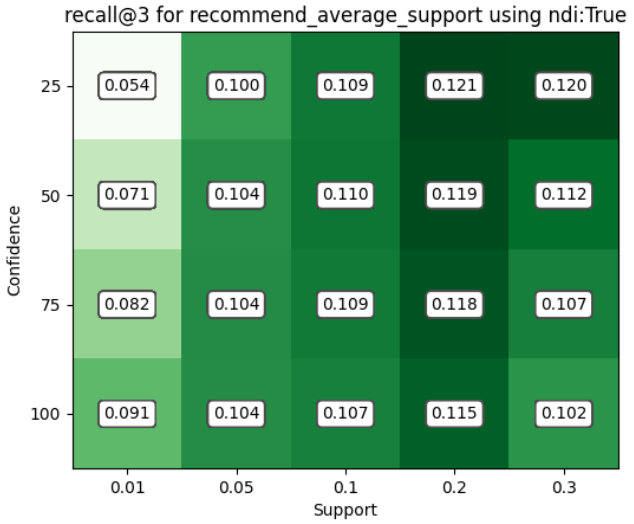

Figure 6: Recall@3 for average support with ndi

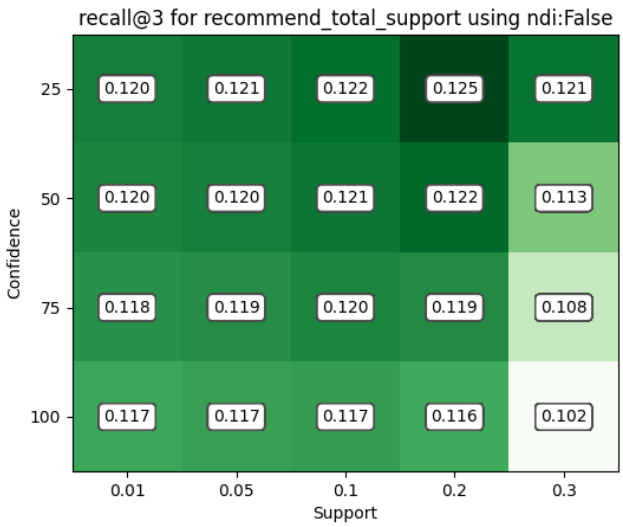

Figure 7: Recall@3 for total support without ndi

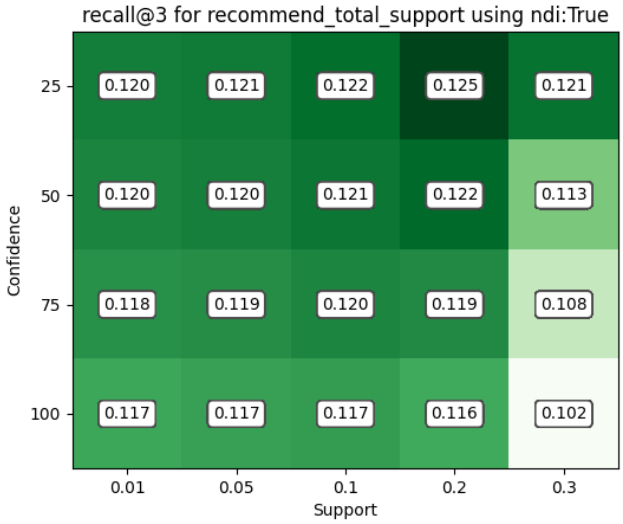

Figure 8: Recall@3 for average support with ndi