Research project: recommendation system improvement through dataset expansion

Quentin De Haes

June 19, 2022

1 Introduction

In this research project, we try to improve the efficacy of recommender systems. We do not develop a new recommender, instead we try to make alterations to the training data. We show that the recommender trained using our expanded dataset achieves better results than the same recommender trained on the original data. The dataset on which we enacted this experiment was this ecommerce dataset from kaggle. The main fields within the dataset we consider are "product_id", used to distinguish the different items, and "user_session", to group items frequently bought together. The code for this project can be found in this github repository

2 Expanding the dataset

In order to expand the dataset, we start by acquiring the pairwise similarity between each item within the dataset. In this project, we investigated four different similarity metrics to try and get their efficacy. All these similarity metrics are based on the Support of an item A, i.e. the number of different sessions in which item A occurs.

1.
$$Cosine\_similarity(A, B) = \frac{Support(A \cap B)}{\sqrt{Support(A) * Support(B)}} \tag{1}$$

2.
$$Jaccard\_similarity(A, B) = \frac{Support(A \cap B)}{Support(A) + Support(B) - Support(A \cap B)} \tag{2}$$

3.
$$Conditional\_Probability(A, B) = \frac{Support(A \cap B)}{Support(A)} \tag{3}$$

4.
$$Pointwise\_Mutual\_Info(A, B) = \frac{Support(A \cap B) * Support(\varnothing)}{Support(A) * Support(B)} \tag{4}$$

Using these metrics, we can calculate the similarity score between each pair of items A and B. The highest possible similarity score we can achieve for each of these metrics will always be achieved through the same means, The similarity between item A and itself will always have the highest similarity. For all metrics except for Pointwise_Mutual_Info, distance(A, A) = 1. Pointwise_Mutual_Info is the exception due to $Support(\varnothing)$ being used within this formula. We can normalize the similarity scores acquired by Pointwise_Mutual_Info for each pair A, B as similarity of item B relative to item A by dividing the score by Pointwise_Mutual_Info(A, A) ensuring 1 is the highest similarity score achievable for all similarity metrics used.

2.1 Acquiring the most similar pairs

When we have decided which metric to investigate, we use this metric to acquire the k most similar items for each item A within the dataset, along with the similarity score the similar items achieved. This means we require a mapping going from each item A to a tuple containing to lists $([B_1, B_2, ..., B_k], [S_1, S_2, ..., S_k])$, where B_i is the item that achieved similarity score S_i, the i^{th} highest similarity score. In each and every mapping of each item A, $B_1 = A$ and $S_1 = 1$. The optimal value of k is something we will investigate within this project.

2.2 Expanding the dataset

Using our mapping, we generate a new dataset. To start with, we copy each session within our dataset n times. This already makes it so that our data set is n times as large, with n times as many items and n times as many sessions. Once we finish this we alter the items within our new dataset based on our mapping. In this project, we propose 2 different techniques to alter each session within our new dataset.

2.2.1 Replacing items

Each item A, we replace by one of the similar items in the list $[B_1, B_2, ..., B_k]$. We do not give each item B_i equal probability to be the replacement. The chance of B_i replacing A is based on the similarity score S_i. More specifically B_i has a probability of $\frac{S_i}{\sum_{j=1}^{k} S_j}$ of replacing A. Once we have done this for every item A within our already n times expanded dataset, we have our new dataset to train our recommender on.

2.2.2 Adding items

The second technique, A is not replaced by a singular item from the list $[B_1, B_2, ..., B_k]$. Instead, a subset of the list gets added to the session. Once again , each item B_i has a certain change of being added. In this case, the chance for B_i to be added is the same as their similarity score S_i, but in this technique the adding of one item does not impede the adding of the other items from within the list. Once we do this for every item, in the expanded dataset, we have our new dataset to train our recommender on. This dataset will be even larger, as at least 1 (itself) and up to k items will be within the dataset for each item in the dataset that is already n times as large as the original dataset.

3 Results

The first recommender we tried was a rule-based recommender, where we use eclat to generate itemsets. A variety of different combinations were tried, the results were compiled in the folowing Excel sheet. All these combinations were well documented, each used variable was recorded, along with the hitrate@10 that was achieved with these variables. The documented variables are: the metric that was used; whether items were replaced by 1 of their similar items, or similar items were added based on their similarity; The number of times each session is copied (n) ;The maximum number of similar items are considered per item for replacement/addition; the minimum support for Eclat; the minimum confidence for the generated rules. After trying many combinations, we can see that all metrics except for Pointwise_Mutual_Info provide some improvement when compared to the base method, and while both techniques of expanding the dataset experience improvement, the improvement provided by Adding extra items is greater than the improvements provided by replacing the items.

The only combination not shown within the file is adding items based on the Pointwise_Mutual_Info metric. The reason for not including this metric was rather simple. The similarity scores acquired through Pointwise_Mutual_Info were too high, when only 5 similarities were considered, on average, between 4 and 5 items were added. Adding the fact that 10 copies were made, this means this dataset was about 45 times as large as the original dataset, with each session also being 5 times as large. The machine requirements to run this would be too high. This is the greatest weakness currently noticed within this technique. Since our technique significantly increases the training data, the machine requirements, as well as the training time for the recommender increase drastically compared to using the original dataset.

Along with that, due to the nature of using probability to add/replace items in the dataset, this means that if no seed is used, the results will be slightly different every time the technique is used.

After looking through the results, it can be seen that a higher number of copies of the session result in a higher hitrate, this does on the other hand, also cause a larger dataset every time this value is increased.

We then tried to use the recommender system based on k nearest neighbours. Generally, we can conclude that this recommender provided better results than the rules based recommender. But if we compare the knn-based recommender (using conditional probability as similarity metric) trained on the base dataset, this provided a hitrate of 13.2%. When we use that same recommender trained on an expanded dataset, this ended up with a hitrate if 14%. This means our method improved the results 6.06%.

4 Conclusion

After many different experiments, we can see that our expansion of the dataset improves, the results achieved by the recommender. The larger we make the generated dataset through copies of the sessions, the better the results, but along with that an increase in required resources and time is also the effect.

In this research, we have shown the efficacy of expanding the dataset, but also make it clear that the level of improvement should be weighed against the increase in resources and time. Along with that, we have shown that the similarity metric should be carefully considered, as not all metrics are equally capable.