Similar Product Clustering for Long-Tail Cross-Sell Recommendations

Vladislav Grozin, Alla Levina

Diginetica
{vlad.grozin,alla.levina}@diginetica.com

Abstract. One of the main reasons of the rapid growth and development of e-commerce is the ability of on-line stores to provide large varieties of products, unlike their off-line counterparts, confined by stock spaces. This also means that on-line stores have products, which have few individual purchases but form a large chunk of revenue all together. They are known as "long-tail" products.
Long-tail data sparsity makes it challenging to apply recommender algorithms. In this paper, we consider cross-sell recommendations using association rule mining. We consider application of clustering techniques to tackle this problem, and compare different clustering and distance calculation methods. Behavioral data, such as product views, and content data, such as category tree and product names, are used to calculate product similarity. Also, we develop a cross-validation method that allows stable metric calculations for algorithms that focus on long-tail products. We show that product clustering that uses session-based distances improve cross-sell recommendations for long-tail items.

Keywords: recommender systems, cross-sell recommendations, association rules, long-tail

1 Introduction

Nowadays, recommender systems are omnipresent in on-line retail. In the last decade, recommender systems have greatly developed. The growth of Internet and e-commerce sites such as Amazon.com, Netflix, and the iTunes Music Store, has opened the door to so-called "infinite-inventory" retailers, which offer large variety of products. This distinguishes them in a positive way from off-line stores which focus more on top-sellers and hits due to physical limitations of shops and costly stock spaces [10]. However, the large variety of products also means that there will are a lot of items that are purchased and viewed a few times. Indeed, majority of the products in site catalogs belong to "long-tail" - items with have much less number of purchases than items from "short-tail" (hits). This long-tail accounts for large chunk of revenue, so business cannot ignore long-tail products.

Recommendations are called "cross-sell" when recommended items are "complementary" to item being viewed (external battery and cases for phones; cables and supports for TV) [4]. Unlike "similar items", or "items similar to ones user

liked in past", such recommendations guide user through site catalog, making navigation easier, while generating additional sales of high-margin supplementary products [6]. This makes cross-sell important for both site functioning and user satisfaction [9]. A common way of generating cross-sell recommendations is association rule mining [8]. Such recommendations are often labeled as "commonly purchased together", "do not forget to buy".

However, data sparsity makes it challenging to apply recommender algorithms. In our paper, we investigate item clustering approaches that group similar items together, and use such dense representation of data to improve quality of cross-sell recommendations generated by the association rule mining algorithm.

Below we consider related research, describe the approach to a decision of the problem, list the methods of the clustering and methods of measuring the distances between items, describe the experiments and see the results.

2 Related research

association rule mining (ARM) is a technique that extracts rules from dataset. These rules are often written as $X \Rightarrow Y$, where X is condition (antecedent, driver), and Y is consequent. Both X and Y are sets of items. X represents the set of items that a user has already purchased, and Y is the set of items that the user is likely to purchase next. This technique was formalized in [1].

For given rule $X \Rightarrow Y$, support is defined as

$$support(X \Rightarrow Y) = P(A \cap B) \tag{1}$$

$P(T)$ is probability to find the transaction T in dataset. Support shows how often spe cified items appear in the dataset. Confidence of rule $X \Rightarrow Y$ is the ratio of the number of the transactions that contain X and Y over the number of the transaction containing Y, and defined as

$$confidence(X \Rightarrow Y) = P(A \cap B)/P(A) \tag{2}$$

Lift of rule $X \Rightarrow Y$ is the ratio of the observed support to that expected if X and Y were independent:

$$lift(X \Rightarrow Y) = P(X \cup Y)/(P(X) \cdot P(Y)) \tag{3}$$

ARM is an effective method for generating cross-sell recommendations. The authors of [4] consider a recommender system that for given user purchase history picks recommended items using plain confidence, lift, or profit-oriented metrics. It is shown that the lift and profit-oriented metrics provide better results than the confidence does.

In practice, recommendation systems face a long-tail problem when many items were purchased and viewed few times in the dataset. It was shown in [10] that ARM in long-tail shows low performance due to low support and confidence of mined rules for unpopular products.

Long-tail effect is well-known, and many authors tried to improve the recommendations for long-tail. Authors of [10] adopted random walk on bipartite user-item graph to make personalized recommendations for long-tail. By using products as starting point of random walk, and measuring average steps to the user of interest, the authors promoted recommendations of unpopular items. However, this algorithm finds similar items to the ones that user has interacted with, thus making it not applicable to the cross-sell.

Park in [7] employed product clustering to create dense data representation. He applied adaptive clustering for the task, and showed that it performed better than the conventional k-nearest neighbors clustering. Euclidean distance between vectors of basic product features was used as a distance between the products. This distance didn't take into account item category or genre similarity, or how many users were interested in both items. For example, if two movies of different genres have similar average rating and unbiased average rating, these movies would be considered similar. The author run the linear model in order to predict user preferences. In contrast, we solve the different recommender problem. However, this paper gives insight into how we can solve sparsity problems by clustering items together.

3 Approach description

Lift is used to determine, which items are the best to be recommended for given user's history [4]. We consider only single-item antecedents of mined rules because increase in condition complexity decreases support, which is already low in long-tail [10]. So, in this research, when a user purchases or adds to a cart some item, recommendations are generated using only this particular item. Also, we focus on recommendations for long-tail items.

Item clustering is used to reinforce individual item data. For the item of interest, we pick several nearest neighbors using some distance metric, and treat this set as a single item [7]. Let C be the set of N nearest neighbors with individual items C_i picked by clustering algorithm. If we treat this set of items as a single item, we can extend the lift definition:

$$Lift(C, Y) = P((C_1 \cup C_2 \cup ... \cup C_N) \cap Y)/(P(C_1 \cup C_2 \cup ... \cup C_N) \cdot P(Y)). \quad (4)$$

We also have to employ some distance between products that can capture semantic similarity. For example, if many users have viewed specific pair of products together, these products are likely to be similar.

3.1 Methods of clustering

We considered two methods of clustering: total clustering and adaptive clustering [7]. Note that the neighbor relationship in these approaches is not symmetric relation in these: if item A has item B as a cluster neighbor, it is not guaranteed that B will have A as a cluster neighbor.

Total clustering For a specified item, this method picks into cluster K nearest neighbors to the item of interest, according to some distance metric.

Adaptive clustering Premise of this method is that we want bigger clusters if the products have few records available for them, and smaller clusters if the products have enough data. So, let us define some threshold N, and pick the nearest neighbors one by one until the total sum of the picked items' records exceeds the threshold. In our case, we count purchase orders. Note that different items will have different size of neighborhood. For instance, if an item is popular, it may even contain only the item itself.

3.2 Metrics of distances

Category Distance. Hierarchical category structure is commonly found in e-commerce resources. We have a few top-level categories that have successors, categories-successors has their successors and so on until we come to the leaves of this structure — items. Category Distance between two items is the length of the shortest path between these items [10].

Session Distance. We build the bipartite user-item graph [10], and use link prediction techniques [5] to assess item similarity. One of popular and simple metrics to measure similarity is the Jaccard's measure. In our case, we use this metric to measure similarity of sets of sessions viewed two items. For example, if majority of the users has viewed both items within one session, we can assume that those items are similar.

Prod2Vec Distance. Prod2Vec model was proposed in [2]. The premise of this approach is that we can consider user sessions as "sentences", and individual items as "words". After applying conventional Word2Vec model to our dataset, we get embeddings for each item. So, we can measure thus item similarity as the scalar product of their vector representations.

4 Experiments

We have to evaluate how well we are giving recommendations for users, for given clustering algorithm and distance metric. NDCG [3] is a common and widely used metric to calculate recommendation quality. Our baseline we is the cross-sell recommendations using ARM and lift as recommendation value [4]. Such baseline picks items with best lift measure for given user purchase history.

In order to determine the best way of computing item similarity and the best clustering method, all combinations of the clustering methods and distance formulas are run. Also, we have to pick the best clustering hyperparameters. Therefore, each clustering-pair distance is evaluated several times, each time with different cluster size. Number of items per cluster is varied from 2 to 10

with step of 1 for total clustering, and number of purchases per cluster is varied from 15 to 300 with step of 15 for adaptive clustering.

4.1 Data sources

We use data provided by e-commerce resource that have agreed to conduct experiments. Date ranges of datasets are non-disclosed, datasets contain a anonymized sampled data. Page views other than item page views are not used in this research. We have 2.3M of unique sessions, 8.5 of item view events, 50k products in catalog, 1k categories, and 434k of purchase orders. Most commonly purchased product has 130 purchases, and top-1% of most commonly purchased items account for 78% of revenue.

4.2 Cross-validation

One of the challenges of working with a long-tail is cross-validation. If we split data with conventional k-fold split, then we would end up with dozen samples in train and test sets, and some products may have no samples in the test set at all. Therefore, we have designed the cross-validation procedure in a way that it would stratify data by products, guaranteeing data entries in both train and test for each item. Also, we considered highly-purchased products in order to have enough data to work with, and emulated long-tail sparsity by moving majority of the records to the test set. In order to ensure reproducibility, random seed was fixed.

Full evaluation procedure is described below:

1. Take all top-sellers (top-1% of items, ordered by sold quantity). Randomly assign half of them to "test items" set.
2. For every item in test items:
   (a) Mark random 10% of sessions that have viewed this item
3. Put all events from sessions that have been marked, or have not viewed items from test item set, into train set, and move all other events to the test set. This action leaves only a small portion (10%..20% of original record count of test items) of data in the train set, effectively emulating environment with sparse data. Howewer, we keep a lot of records (80%..90%) in the test set to accurately assess the algorithm performance.
4. For each session in the test set:
   (a) For each item from the test item set, which was purchased within this session:
      i. Build recommendations for this particular purchase event, item and session:
         A. Run clustering with specified distance and settings to determine nearest neighbors to the item being purchased.
         B. Fetch all orders that contain at least one item from the previous step.

C. Iterate over all items in these orders, and calculate $Lift(C, Y)$, where C is purchased item neighbor cluster, and Y is a candidate item.
D. Exclude all the items selected at stage A from the result obtained at stage C.
E. Take 20 items with the highest lift value from obtained in stage D list, and recommend these items to user.

ii. Assess recommendations quality for this particular purchase event, item and session.

Conventional stratified k-fold split would hide a portion $(1/k)$ of data for each product. We are interested in long-tail, so we have to pick products with few records in train set. This also means that we would have even less records in test set for metric measurement. If we simply put most of the data in test set, it would put models in unrealistic situation where all items in train set have few records. For example, models would not be able to reinforce data by picking nearest popular product to the long-tail ones because all products will have few records left. This can be solved by hiding majority of records only for some products, and leave other products unaffected.

Also, users sessions can have complex structure. This is important for models that use sequence of events, for example Prod2Vec [2]. Therefore, we should not hide single events, we should put entire sessions in test set.

Thus, our algorithm takes portion of popular products, and puts majority of sessions that have viewed these products in test set, leaving only few sessions in train set. After that, we measure how well algorithm makes recommendations for these products. Such cross-validation procedure forces algorithm to make recommendations using sparse train data for these products, but we also have large amount of records for metric measurement. This way we can properly assess recommender performance for long-tail products.

4.3 Quality measurement

We use well known and often used NDCG@20 to asses recommendations quality [3]:

$$NDCG = \frac{DCG}{IDCG}. \tag{5}$$

$$DCG = \sum_{i=1}^{n} \frac{rel_i}{log(i+1)}, \quad IDCG = \sum_{i=1}^{|orderedRel|} \frac{orderedRel_i}{log(i+1)}, \tag{6}$$

where rel_i is relevance of ith recommended item, $orderedRel$ is the set of items ordered in descending order by their relevance. So, $IDCG$ is normalization coefficient that ensures that $NDCG$ lies between 0 and 1.

NDCG metric is calculated at step 5.a.ii of our algorithm for each instance when a test user buys a test item; after that, the values are averaged. Our

goal is to guess what the user is going to purchase in the future. Therefore, we evaluate *rel* of the the recommended item as 1 if the user is going to purchase recommended item in future, and 0 otherwise. We skip instances when the user will not purchase any items in the future.

5 Results

Table 1 shows performance of the total clustering method, and Table 2 shows performance of adaptive clustering. The tables contain only values for best hyperparameter. Baseline score is 0.0557, and scores higher than baseline are highlighted in the tables.

It can be seen that the adaptive clustering shows better results, and the best distance method is Jaccard's distance between sessions. Adaptive clustering with session-based distance gives +6.8% NDCG over the baseline.

Table 1. Results of Total Clustering

Distance	Category	Session	Prod2Vec
Resulting cluster size	2	4	2
NDCG	0.0545	**0.0560**	0.0550

Table 2. Results of Adaptive Clustering

Distance	Category	Session	Prod2Vec
Minimum number of purchases in cluster	60	135	45
NDCG	0.0553	**0.0595**	0.0554

We can see that session-based distance works best for both types of clustering. Category-based distance provides low performance. We account this for the fact that categories ofter contain diverse sets of products, so clustering that uses category-based distance may include unrelated items in cluster. Prod2Vec distance also have not beaten baseline. This can be explained by the fact that long-tail products do not have enough data for this model.

6 Conclusion and future work

We have used association rule mining in order to generate recommendations. We have shown that by using Jaccard's measure as a distance between products, we can cluster similar items together; such clustering improves cross-sell quality for

long-tail. Adaptive clustering that increases cluster size when there is lack of data works better than total clustering that always picks constant amount of neighbors.

In future works, we will should extend our approach. For example, we can combine several distance measures into one that works better. Also, currently we mine only rules with single cluster in antecedents, and item in consequents. We should consider mining rules with multiple clusters in antecedents, and clusters (instead of items) in consequents.

References

1. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules. In: 20th VLDB Conf. (Sep 1994)
2. Grbovic, M., Radosavljevic, V., Djuric, N., Bhamidipati, N., Savla, J., Bhagwan, V., Sharp, D.: E-commerce in your inbox: Product recommendations at scale. In: Cao, L., Zhang, C., Joachims, T., Webb, G.I., Margineantu, D.D., Williams, G. (eds.) KDD. pp. 1809–1818. ACM (2015), http://dblp.uni-trier.de/db/conf/kdd/kdd2015.html#GrbovicRDBSBS15
3. Järvelin, K., Kekäläinen, J.: Cumulated gain-based evaluation of ir techniques. ACM Transactions on Information Systems (TOIS) 20(4), 422–446 (2002), http://scholar.google.de/scholar.bib?q=info:6Bdw8cs-UYMJ:scholar.google.com/&output=citation&hl=de&as_sdt=0,5&ct=citation&cd=0
4. Kitts, B., Freed, D., Vrieze, M.: Cross-sell: a fast promotion-tunable customer-item recommendation method based on conditionally independent probabilities. In: Ramakrishnan, R., Stolfo, S.J., Bayardo, R.J., Parsa, I. (eds.) KDD. pp. 437–446. ACM (2000), http://dblp.uni-trier.de/db/conf/kdd/kdd2000.html#KittsFV00
5. Liben-Nowell, D., Kleinberg, J.: The link-prediction problem for social networks. Journal of the American Society for Information Science and Technology 58(7), 1019–1031 (2007), http://dx.doi.org/10.1002/asi.20591
6. Oktar, D.: Recommendation systems: Increasing profit by long tail. http://en.webrazzi.com/2009/09/18
7. Park, Y.J.: The adaptive clustering method for the long tail problem of recommender systems. IEEE Trans. Knowl. Data Eng. 25(8), 1904–1915 (2013), http://dblp.uni-trier.de/db/journals/tkde/tkde25.html#Park13
8. Riaz, M., Arooj, A., Hassan, M.T., Kim, J.B.: Clustering based association rule mining on online stores for optimized cross product recommendation. In: ICCAIS. pp. 176–181. IEEE (2014), http://dblp.uni-trier.de/db/conf/iccais/iccais2014.html#RiazAHK14
9. Schafer, J.B., Konstan, J., Riedl, J.: Recommender systems in e-commerce. In: Proceedings of the ACM Conference on Electronic Commerce (1999)
10. Yin, H., Cui, B., Li, J., Yao, J., Chen, C.: Challenging the long tail recommendation. PVLDB 5(9), 896–907 (2012), http://dblp.uni-trier.de/db/journals/pvldb/pvldb5.html#YinCLYC12