Tensor-based Complementary Product Recommendation

Negin Entezari*
Department of Computer Science and Engineering
University of California Riverside
Riverside, CA, USA
nente001@ucr.edu

Evangelos E. Papalexakis
Department of Computer Science and Engineering
University of California Riverside
Riverside, CA, USA
epapalex@cs.ucr.edu

Haixun Wang
Instacart
San Francisco, CA, USA
haixun.wang@instacart.com

Sharath Rao
Instacart
San Francisco, CA, USA
sharath@instacart.com

Shishir Kumar Prasad
Instacart
San Francisco, CA, USA
shishir@instacart.com

***Abstract*—In recent years, online grocery shopping has become very popular, and platforms such as Instacart, Amazon Fresh, Shipt, and Walmart Grocery have attracted millions of customers. To satisfy the customers' needs, it is vital to provide relevant personalized recommendations and ease the customers' shopping experience. In this paper, we propose a tensor-based method that utilizes a three-mode tensor to represent product-to-product relations for users and applies tensor decomposition techniques to jointly learn user and product embeddings that can be used to infer within-basket recommendations. Products co-purchased in a single transaction are modeled in the form of a tensor. Then, we leverage RESCAL tensor decomposition technique to capture the latent factors that reveal the inherent user and product interactions. On the Instacart dataset, our proposed tensor-based method achieves a recall@10 of 0.192, whereas recall@10 for triple2vec, which is the state-of-the-art, is 0.149.**

I. Introduction

Customers face millions of products in an online grocery shopping experience, making the shopping process an exhausting and confusing task for them. Recommender systems are valuable tools that help the customer by narrowing down the search space to products that the customer is desired to see and purchase. Personalized recommender systems are critical components of online shopping platforms. Personalized recommendations based on customers' shopping habits are beneficial to customers and can lead to sales growth. In online grocery shopping, customers often follow repetitive shopping habits. They tend to purchase specific products over and over and rarely switch to other similar products. Therefore, analyzing customers' shopping behavior and providing personalized recommendations is of high importance in an online grocery shopping platform.

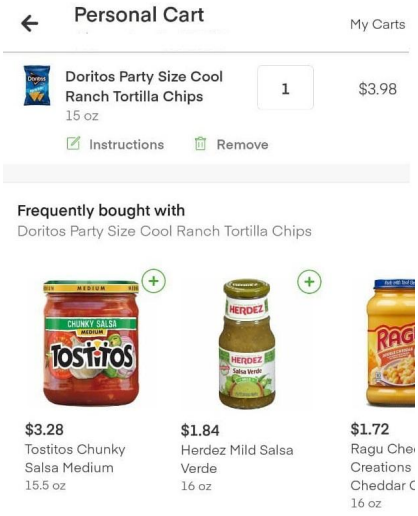


Fig. 1. Example of complementary products recommended according to the current product in the basket of the customer. Here different types of salsa are recommended to the customer with a bag of chips in his/her shopping cart.

One of the main types of personalized recommendations is complementary recommendations. Considering a single shopping session, products that are often purchased together in one basket are considered to be complementary to each other. Complementary products are related to each other in some way and together fulfill customer needs. Chips and salsa, burger and burger buns, peanut butter and jelly are examples of complementary products often purchased together. To improve customer's experience during online shopping, it is vital to recommend relevant products according to what currently exists in the customer's basket.

A good complementary recommender system is essential for various reasons:

- **Shopping efficiency**: It helps the customer to build the shopping basket efficiently and reduces exploration time.

*This work was done while the first author was an intern at Instacart.

Research was supported by the National Science Foundation CDS&E Grant no. OAC-1808591.

978-1-6654-3902-2/21/$31.00 ©2021 IEEE

They can quickly find the relevant products from the recommendation list, instead of having to search for them. Thus, this can help customers save time. According to Instacart platform, on average, it takes about 45 minutes to build a basket. By providing relevant and personalized recommendations, this time can be reduced and the shopping process will get more convenient for the customer.

- **Novel product recommendation**: Novel products for a customer are those that the customer has never purchased before. Sometimes, customers have no idea about what could be complementary to the products in their basket. Such basket-contextual, personalized recommendations can help customers discover novel products, especially at the end of the shopping process.
- **Business growth**: From a business perspective, novel product purchases can help to increase the basket size of the customer and generates incremental Gross Merchandise Value (GMV). Moreover, customers who have a seamless experience are highly likely to come back for future shopping.

This paper introduces a tensor-based method to address the complementary recommendation problem in online grocery shopping. Tensor is used to represent products co-purchased by customers and tensor decomposition techniques are used to find product and customer embeddings in low-dimensional space. In the next step, the embeddings are used to score products with respect to the current basket of the customer and products with the highest scores are recommended as complementary to the current basket. Tensor-based recommender systems have shown great success by considering multiple aspects of data and incorporating additional information such as context. Tensor modeling and factorization learns a joint representation of the items and the context, which has been shown to result in richer representations that can provide better estimation for missing ratings/scores [1].

Our contributions are as follows:

1) **Novel tensor-based formulation**: We introduce a tensor-based method that represents complementary product pairs in the form of a three-mode tensor and we use tensor decomposition techniques to infer product and user embeddings.
2) **Efficient Solution**: We consider mini-batch tensors that allow parallel and sequential tensor decomposition to handle large-scale datasets.

The rest of this paper is organized as follows. In Section II we discuss related work. We introduce our proposed method in Section III and provide experimental results in Section IV. Finally, in Section V we offer conclusions and discuss future work.

II. Related Work

A. Frequent Purchase Mining

In the field of complementary product recommendation, one basic and trivial method is to recommend products according to their frequency of purchase [2]. In a non-personalized recommendation task, most frequently purchased products by all customers are recommended to the user, whereas in a personalized task, most frequently bought products by the current user are recommended to him/her. In both cases, recommended products ignore the current basket content.

B. Collaborative Filtering and Matrix Factorization

Many of the recent work use collaborative filtering and matrix factorization techniques to model user-product and product-product relationships. Basket-sensitive Factorization Machine (BFM) and constrained BFM (CBFM) methods use a combination of matrix factorization and association rules to provide complementary recommendations [3]. Collaborative filtering and matrix factorization technique only consider user-item interactions and do not take advantage of additional information available such as product/user features and contextual information. However, tensor-based recommender systems are able to incorporate this additional information and improve the performance of the recommendation. When there is an inherent structure between the interactions, then unfolding that structure may not be efficient in terms of our ability to learn a good representation with the given amount of data. This observation has also been shown in non-factorization-based scenarios where structure is not ignored and helps learn better recommenders [4]. Powerful recommender systems also consider contextual features and matrix factorization techniques only consider first-order interaction of users and items and ignore the additional contextual features that can improve personalized recommendation. For instance, considering contextual information such as time and location lead to stronger recommendations [5], while matrix factorization methods cannot be easily adapted to leverage such information.

C. Representation Learning

Another group of studies use popular word representation learning techniques in NLP, like skip-gram, to generate product recommendations. Liang et al. [6] combined matrix factorization and word2vec item embeddings to learn product recommendations. **Item2vec** [7] is an extension of word2vec that infers item-item relations by learning items representations in a low-dimensional space. **Prod2vec** [8] is another method in this category that learns product representations from user purchase histories. An important characteristic of a complementary recommender system is to jointly learn product-product and user-product relations, and the aforementioned methods fail in this aspect . In a three-mode tensor representation where one aspect is product-product relationships and another aspect is user-product interactions, tensor decomposition provides latent factors that capture the hidden structure of data by jointly optimizing on both aspects. For instance, in a three-mode tensor, tensor factorization's objective is as follows:

$$\underset{A,B,C}{minimize} \|\underline{\mathbf{X}} - [\![A, B, C]\!]\|_F^2 \tag{1}$$

where A, B, and C are latent factor matrices derived from the tensor factorization and the factor matrices that minimize the objective function are learned at the same time.

One of the state-of-the-art methods in complementary product recommendation is the **triple2vec** method [9]. This method also utilizes skip-gram embedding learning framework. Triple2vec performs Skip-gram with negative sampling over (product i, product j, user u) triples. Two products co-purchased in a basket by a user form a triple. Triples are used to generate product and user embeddings. Triple2vec inference time increases with basket size. To address this problem, **RTT2vec** (Real-Time Triple2vec) [10] was proposed by Mantha et al. that transforms inference into a similarity search problem and improves the inference time by utilizing approximate nearest neighbor indexing methods such as ANNOY, Faiss, and ScaNN [11], [12], [13].

D. Tensor-based Recommenders

Tensor-based methods can be considered as an extension of matrix factorization recommender system . In matrix factorization, we are dealing with 2-dimensional data, while in tensor factorization techniques, data is represented in higher dimensions (≥ 3). Tensor-based methods are able to analyze multiple aspects of data simultaneously and jointly. Matrix factorization models extract user-product interactions, while tensors are able to capture multi-aspect interactions. Tensors are great tools to represent multidimensional data and by considering multiple aspects of data into decomposition, they have been successful in recommender systems [5], [14], [15], [16]. In this paper, we leverage the multi-aspect property of tensors to model product-product interaction within each basket for different users as a three-mode tensor. Decomposing this tensor allows us to find latent components that reveal product-product and user-product interaction to infer personalized complementary recommendations.

III. Proposed Method

A. Tensor decomposition to learn product and user embeddings

Let $P = \{p_1, p_2, ..., p_M\}$ be the set of M products and $U = \{u_1, u_2, ..., u_N\}$ be the set of N users. Given $B_u \subset P$, the set of products in the current basket of user $u \in U$, the goal of complementary product recommendation is to recommend top-k products $R = \{p_1^*, p_2^*, ..., p_k^*\}$ such that $p_j^* \notin B_u$. Theses p_j^* products are considered complementary to products in the current basket B_u.

To model complementary products, we consider product-to-product relationships for each user . Representing product-to-product relationships per user allows us to provide personalized complementary recommendations. Two products that are always purchased together by a user may not be complementary for another user. For instance, user A mostly purchases chips and salsa, but user B purchases chips and guacamole most of the time. If we only consider global product-to-product relationships, complementary recommendations may be compatible with the need of the majority of users, but it does not fulfill the needs of users who do not behave like others and have their own preferences. For instance, assume most of the users consume meat, but there are small number of users who are vegetarian. By only considering the global product-to-product relations, we ignore the minority group. However, someone who is vegetarian does not want to see meat recommendations. Therefore, we require to capture the behavior of each user separately and recommend products compatible with their shopping profile. We represent this information in the form of a three-mode tensor (three-dimensional array) $\underline{\mathbf{X}}$. In this paper, A tensor is denoted by an underlined bold uppercase. Next, we learn product and user embeddings using tensor decomposition techniques. The idea behind tensor factorization is to represent users and products in a lower-dimensional space. Each element of the three-mode tensor $\underline{\mathbf{X}}$ represents the number of times two products have been purchased together by a user:

$$\underline{\mathbf{X}}\,(i, j, k) = c_{ijk}; \quad p_i, p_j \in P \text{ and } u_k \in U \tag{2}$$

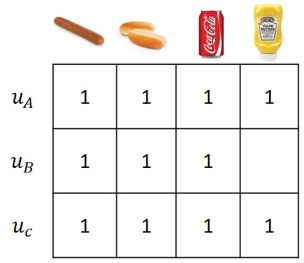
where c_{ijk} is the number of times that user u_k has purchased two products p_i and p_j together.

To capture complementary relationships between products, we need to track products that are co-purchased in a single basket. Traditional matrix factorization methods ignore such information and only consider user-product interactions. To better elaborate the difference between matrix and tensor representations, consider the following example. Given 3 users u_A, u_B, and u_c, assume the following transactions for them:

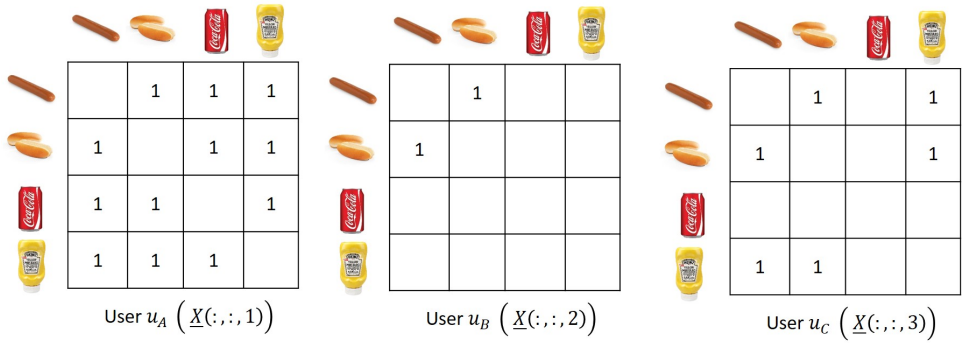
- User u_A performs one transaction:
  - Hot dog, hot dog buns, coke, and mustard
- User u_B performs the following three transactions separately:
  - Basket #1: Hot dog and hot dog buns
  - Basket #2: Coke
  - Basket #3: Mustard
- User u_C performs a single transaction:
  - Hot dog, hot dog buns, and mustard

Fig. 2 shows matrix vs. tensor representation corresponding to the aforementioned example. Using the matrix representation, users u_A and u_B are exactly similar because they have purchased the same products. However, using the tensor representation, users u_A is more similar to u_C than user u_B, and this is what we expect as users u_A and u_B have performed similar transactions. Classical matrix factorization methods predict the probability of recommending an item given a user ($P(item|user)$, whereas for the task of complementary recommendation, we are interested in computing the probability of recommending an item given a user and their current basket ($P(item|user, basket)$).

Traditional and popular tensor decomposition like CANDECOMP/PARAFAC (CP) [17] and Tucker [18] generate three different latent matrices corresponding to each mode of the tensor. Here, in our problem, the first two modes of tensor $\underline{\mathbf{X}}$ are identical and corresponds to products. Therefore, we only require two of the latent factor matrices. Tucker-2 is a restricted form of Tucker decomposition in which two of the factor matrices are equal. Another decomposition technique that can be applied to our problem is RESCAL [19]. RESCAL has been used to learn the inherent structure of relational



(a) Matrix representation



(b) Tensor representation

Fig. 2. (a) Matrix representation vs. (b) Tensor representation of transactions data.

data. Here, we are also interested in learning the relationship between products purchased together and RESCAL tensor decomposition method is able to capture this type of relationship between products. Interested readers may refer to [20], [21] for a detailed comparison of tensor decomposition techniques.

Fig. 3 shows the RESCAL decomposition of user-product tensor X. RESCAL decomposition can be formulated as follows:

$$\underline{\mathbf{X}} \approx A\underline{\mathbf{R}}A^T \tag{3}$$

$$\underline{\mathbf{X}}_k = A\underline{\mathbf{R}}_k A^T \tag{4}$$

where A is an $M \times d$ latent factor matrix that contains products embeddings and tensor $\underline{\mathbf{R}}$ which is an $d \times d \times N$ is the latent factor corresponding to user embeddings. $\underline{\mathbf{X}}_k = \underline{\mathbf{X}}(:,:,k)$ is called a frontal slice of the tensor $\underline{\mathbf{X}}$ and represents product co-purchased by user u_k. Likewise, frontal slice $\underline{\mathbf{R}}_k = \underline{\mathbf{R}}(:,:,k)$ is the user embedding corresponding to user u_k.

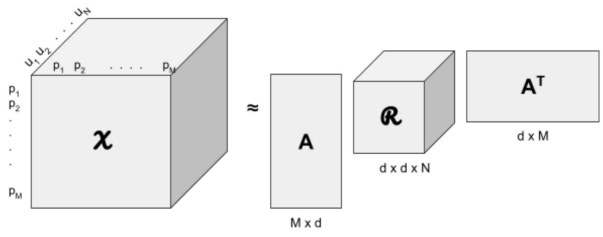


Fig. 3. RESCAL tensor decomposition

The factor matrices A and $\underline{\mathbf{R}}$ are computed by solving the regularized minimization problem [19]:

$$min_{A,\underline{\mathbf{R}}_k} f(A, \underline{\mathbf{R}}_k) + g(A, \underline{\mathbf{R}}_k) \tag{5}$$

$$f(A, \underline{\mathbf{R}}_k) = \frac{1}{2}\left(\sum_k \|\underline{\mathbf{X}}_k - A\underline{\mathbf{R}}_k A^T\|_F^2\right) \tag{6}$$

$$g(A, \underline{\mathbf{R}}_k) = \frac{1}{2}\left(\|A\|_F^2 + \sum_k \|\underline{\mathbf{R}}_k\|_F^2\right) \tag{7}$$

where $f(A, \underline{\mathbf{R}}_k)$ tries to minimise the distance and $g(A, \underline{\mathbf{R}}_k)$ is the regularization part to avoid overfitting.

The product embedding matrix A is shared across all users and by taking the dot product of this matrix and its transpose, products most frequently bought together will have a higher score. On the other hand, the user embedding matrix $\underline{\mathbf{R}}_k$ captures the interaction between products that are mostly purchased by a specific user u_k. Therefore, the matrix $\underline{\mathbf{R}}_k$ is used to adjusts the product-to-product scores for the user u_k and therefore product p_j that maximizes $A_i\underline{\mathbf{R}}_k A_j^T$ is the personalized complementary product with respect to product p_i . To elaborate the idea further, consider the element-wise form of the equation 6:

$$f(A, \underline{\mathbf{R}}_k) = \frac{1}{2}\left(\sum_{i,j,k} \left(\underline{\mathbf{X}}_{ijk} - A_i\underline{\mathbf{R}}_k A_j^T\right)^2\right) \tag{8}$$

where $A_i = A(i,:)$ and $A_j = A(j,:)$ are rows of latent factor matrix A that are embedding vectors of length d corresponding to products p_i and p_j, respectively. Assume, user u_k currently

has product p_i in their basket and our goal is to find a product that is complementary to p_i. To minimize $f(A, R_k)$, the term $A_i \underline{\mathbf{R}}_k A_j^T$ should be as close as possible to the value $\underline{\mathbf{X}}_{ijk}$. If products p_i and p_j are frequently purchased together by user u_k, the value of $\underline{\mathbf{X}}_{ijk}$ will be large and therefore the term $A_i \underline{\mathbf{R}}_k A_j^T$ should be maximized. Given a product p_i, its complementary product p_j will have an embedding which is closest to the embedding corresponding to p_i, i.e. the dot product of $A_i A_j^T$ will have the highest score. Thus, product(s) that maximize the following equations are considered as top N complementary products with respect to product p_i:

$$\underset{j \in P \setminus B_{u_k}}{\arg\max} A_i \underline{\mathbf{R}}_k A_j^T \qquad i \in B_{u_k} \tag{9}$$

B. Optimizing RESCAL Decomposition

The algorithm to compute factor matrices in RESCAL decomposition performs alternating updates of matrices A and $\underline{\mathbf{R}}_k$ for all k until $\frac{f(A, \underline{\mathbf{R}}_k)}{||X||_F^2}$ converges to some small threshold or a maximum number of iterations is exceeded.

For a large-scale dataset with millions of users and thousands of products, we will have a huge sparse tensor, and the alternating algorithm is very slow and inefficient. RESCAL is a restricted form of TUCKER decomposition in which one of the modes is left uncompressed, i.e., one of the latent factors is the identity matrix. This restricted Tucker decomposition is known as Tucker-2 [18]. To speed up the decomposition algorithm, we use Higher-Order Singular Value Decomposition (HOSVD) algorithm [22]to approximate factor matrices. HOSVD algorithm does not compute the optimal solution, however, it is very popular due to its simplicity. HOSVD algorithm computes the factor matrices by performing singular value decomposition on the matricized form of the tensor across each mode (dimension).

Moreover, performing tensor decomposition on such a large tensor requires lots of memory. To solve this problem, instead of performing decomposition on a single tensor containing all users' data, we split the tensor into smaller batches that only contain a subset of users and perform decomposition on each batch separately. This allows us to run the decomposition on datasets with millions of users. Also, adding new users to the dataset does not require retraining the model on the entire dataset and we can only train our model on the batch of recent users.

C. Optimizing Inference Time

To achieve real-time inference, we need further improvements. With a large number of products in the dataset, computing product scores is very time-consuming and model inference time increases with the basket size. To find top-k products that maximize the score with respect to the current basket, we need to perform dot product between basket product embeddings and all other products in the dataset. To speed up the process, we use hashing technique by using Approximation Nearest Neighbor (ANN) indexing library, ANNOY[1]. This allows us to perform the approximate dot product efficiently. The dot product of two vectors is maximized when they are most similar to each other. We create the ANN index on all products in the dataset. For each basket, the query vector Q_i is $A_i \underline{\mathbf{R}}_k$ for all products $p_i \in B_{u_k}$ and we would like to find the products in the dataset which are closest to the query vector. Therefore, the inference problem can be rewritten as follows:

$$\begin{aligned} &\underset{j \in ANN(Q_i)}{\arg\max}\ Q_i A_j^T \\ &Q_i = A_i \underline{\mathbf{R}}_k; \qquad i \in B_{u_k} \end{aligned} \tag{10}$$

Now, instead of searching through all products in the dataset to find top-K recommendations, we only need to search through L products where $L = |ANN(Q_i)| << M$.

IV. Experiments

A. Dataset and Experiment Setup

In our experiments we used Instacart public dataset published for Kaggle competition in 2017[2]. Th statistics of the Instacart dataset is reported in Table I.

To split transaction data into train/validation/test sets, we follow the setting mentioned in [9]: Transactions are sorted chronologically. For each user, the most recent transaction is used for testing, the second-to-last transaction is used for validation and all other transactions are used for training. For users with only one transaction, it will be used for training, and for users with two transactions, the first transaction is used for training and the last one is used for testing.

B. Evaluation

1) Metrics: We evaluate the performance of models with the following metrics: Recall@K, NDCG@K, and Precision@K. Recall@K measures the fraction of relevant items correctly recommended in the top-K items. NDCG@K (Normalized Discounted Cumulative Gain) is a ranking metric that uses position in the recommendation list to measure gain. Metrics are reported at K=10.

2) Results: To evaluate the performance of our method on the test dataset, we randomly select 80% of the products in the basket as input, and the rest of the products are used as reference for evaluation.

We compare our method with two baselines: **popularity**, and **triple2vec** [9]. Popularity method always recommends the most frequently purchased products by all users. For tensor-based and triple2vec methods, we report results for two different embedding dimensions 32 and 128 to better understand

[1]https://github.com/spotify/ annoy

[2]https://www.kaggle.com/c/instacart-market-basket-analysis/data

TABLE I
INSTACART DATASET STATISTICS

No. of transactions	3,345,786
No. of users	206,209
No. of products	49,684
No. of products purchased at least 10 times	42,987
Average basket size	10.10

TABLE II
PEFORMANCE OF THE PROPOSED MODEL VS. POPULARITY AND TRIPLE2VEC BASELINE METHODS.

Method	Recall@10	NDCG@10	Precision@10
Popularity	0.104	0.081	0.029
triple2vec (d = 32)	0.103	0.162	0.022
tensor (d = 32)	0.145	0.172	0.037
triple2vec (d =128)	0.149	0.178	0.030
tensor (d=128)	**0.192**	**0.193**	**0.047**

TABLE III
PERFORMANCE OF THE PROPOSED MODEL VS. TRIPLE2VEC ON NOVEL COMPLEMENTARY PRODUCT RECOMMENDATION.

Method	Recall@10	NDCG@10	Precision@10
triple2vec-novel (d = 32)	0.012	0.0	0.005
triple2vec-novel (d = 128)	0.015	0.0	0.007
tensor-novel (d = 32)	0.010	0.0	0.005
tensor-novel (d = 128)	0.013	0.0	0.006

the impact of embedding dimension on the recommendation performance. Table II shows the results. Popularity does not consider the context of the current basket and always recommends the most frequent products which might be irrelevant with respect to the current basket. The popularity method memorizes frequent purchases and lacks generalization power since it is unable to capture product semantics. However, customers are often loyal to certain products and brands and repeatedly purchase the same products and are reluctant to switch to some other products. Therefore, the popularity method achieves an acceptable performance by memorizing the shopping history of users. Our proposed tensor-based method outperforms popularity and triple2vec in all cases. A larger value of the embedding dimension (d=128 vs. d=32) helps to better estimate the original tensor and thus improves the performance of the model.

We also evaluate the proposed method and triple2vec for novel product recommendations. Novel purchases are products that customer is purchasing for the first time. Introducing novel products to customers and encouraging them to purchase more products leads to larger basket size and higher GMV for the business. In the novel product recommendation experiment, we are interested to see how many of the top K recommended items are novel and the customer has never purchased them before. Sometimes customers are not aware of possible complementary products and novel recommendations give them the chance to try new complementary products that are frequently purchased by other customers. For the test data in this experiment, we only considered transactions including at least 3 novel products. These 3 novel products are used for inference and the rest of the basket is used as input of the recommender system. The result for the novel complementary recommendation is shown in Table III. Both tensor-based method and triple2vec perform poorly in recommending novel products. They mostly recommend products that have already been purchased by the user. For future work, we aim to utilized coupled matrix-tensor factorization to leverage product features and improve the performance of novel product recommendations.

3) Handling Cold-Start Users: For new customers, we do not have a shopping history; therefore, it is not possible to have a personalized recommendation. In this case, we can either have a non-personalized recommendation and score products just based on the product embedding, or consider the average of all users' embeddings as the user embedding for the new customer. In Table IV we report the performance of the proposed model for the two mentioned cases. Here we only report the result for the embedding dimension 128 that achieves a higher performance.

The performance of personalized and non-personalized approaches to handle new users are about the same and both have an acceptable performance to deal with cold-start users.

4) Evaluating inference time: Here, We compare the inference time of the exact approach with the approximate method using ANNOY library. Table V compares these two approaches for different basket sizes. The exact inference method which computes dot products between all product embedding, gets very slow as the basket size increases and makes it inapplicable.

V. CONCLUSIONS AND FUTURE WORK

In this paper, we proposed a tensor-based model to address the complementary product recommendation task in online grocery shopping. A three-mode tensor was used to model product-product co-purchases by users. Next, RESCAL tensor decomposition technique was used to learn latent factors of the tensor that corresponding to product and user embeddings. These embeddings were utilized to infer complementary products given a current basket of a user. Tensor-based method outperforms popularity baseline and state-of-the-art triple2vec method. Tensor embedding performs poorly to predict novel products. In future, to improve the performance of the tensor method, we can consider product features as a side information and perform Coupled Matrix-Tensor Factorization (CMTF). CMTF will generate an embedding matrix corresponding to the product features which allows us to find similar products and use this information to increase the score for novel products.

To improve training process to handle large scale datasets, we performed decomposition on small tensors including a subset of users and their transactions. Decomposition mini-batches allows to train the model both in parallel and sequentially. If enough memory is available, one can take advantage and run the decomposition in parallel. However, in limited memory systems, we can perfectly run the model sequential but in a longer time. Moreover, we leveraged approximate nearest neighbor indexing library, ANNOY, to speed up the inference process and allow real-time inference.

ACKNOWLEDGMENT

Research was supported by the National Science Foundation CDS&E Grant no. OAC-1808591. Any opinions, findings, and

TABLE IV
PERFORMANCE OF TENSOR-BASED METHOD IN HANDLING COLD-START USERS.

Method	Recall@10	NDCG@10	Precision@10
tensor (d = 128)-average user	0.122	0.271	0.033
tensor (d = 128)-non-personalized	0.120	0.266	0.032

TABLE V
EXACT INFERENCE VS. APPROXIMATE INFERENCE USING ANNOY.

	Inference Time (sec)	
Basket size	Exact method	ANNOY
10	1.089	0.013
20	2.165	0.027
30	3.307	0.053
40	4.344	0.052
50	5.345	0.083
60	6.454	0.065
70	7.557	0.077
80	8.527	0.093
90	9.725	0.086
100	11.387	0.088
110	12.513	0.138
120	13.069	0.164

conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the funding parties.

REFERENCES

[1] L. Xiong, X. Chen, T.-K. Huang, J. Schneider, and J. G. Carbonell, "Temporal collaborative filtering with bayesian probabilistic tensor factorization," in *Proceedings of the 2010 SIAM international conference on data mining*. SIAM, 2010, pp. 211–222.

[2] J. Han, H. Cheng, D. Xin, and X. Yan, "Frequent pattern mining: current status and future directions," *Data mining and knowledge discovery*, vol. 15, no. 1, pp. 55–86, 2007.

[3] D. T. Le, H. W. Lauw, and Y. Fang, "Basket-sensitive personalized item recommendation." IJCAI, 2017.

[4] A. Beutel, P. Covington, S. Jain, C. Xu, J. Li, V. Gatto, and E. H. Chi, "Latent cross: Making use of context in recurrent recommender systems," in *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, 2018, pp. 46–54.

[5] A. Karatzoglou, X. Amatriain, L. Baltrunas, and N. Oliver, "Multiverse recommendation: n-dimensional tensor factorization for context-aware collaborative filtering," in *Proceedings of the fourth ACM conference on Recommender systems*, 2010, pp. 79–86.

[6] D. Liang, J. Altosaar, L. Charlin, and D. M. Blei, "Factorization meets the item embedding: Regularizing matrix factorization with item co-occurrence," in *Proceedings of the 10th ACM conference on recommender systems*, 2016, pp. 59–66.

[7] O. Barkan and N. Koenigstein, "Item2vec: neural item embedding for collaborative filtering," in *2016 IEEE 26th International Workshop on Machine Learning for Signal Processing (MLSP)*. IEEE, 2016, pp. 1–6.

[8] M. Grbovic, V. Radosavljevic, N. Djuric, N. Bhamidipati, J. Savla, V. Bhagwan, and D. Sharp, "E-commerce in your inbox: Product recommendations at scale," in *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, 2015, pp. 1809–1818.

[9] M. Wan, D. Wang, J. Liu, P. Bennett, and J. McAuley, "Representing and recommending shopping baskets with complementarity, compatibility and loyalty," in *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, 2018, pp. 1133–1142.

[10] A. Mantha, Y. Arora, S. Gupta, P. Kanumala, Z. Liu, S. Guo, and K. Achan, "A large-scale deep architecture for personalized grocery basket recommendations," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 3807–3811.

[11] M. Aumüller, E. Bernhardsson, and A. Faithfull, "Ann-benchmarks: A benchmarking tool for approximate nearest neighbor algorithms," in *International Conference on Similarity Search and Applications*. Springer, 2017, pp. 34–49.

[12] J. Johnson, M. Douze, and H. Jégou, "Billion-scale similarity search with gpus," *IEEE Transactions on Big Data*, 2019.

[13] R. Guo, P. Sun, E. Lindgren, Q. Geng, D. Simcha, F. Chern, and S. Kumar, "Accelerating large-scale inference with anisotropic vector quantization," in *International Conference on Machine Learning*. PMLR, 2020, pp. 3887–3896.

[14] H. Ge, J. Caverlee, and H. Lu, "Taper: A contextual tensor-based approach for personalized expert recommendation," in *Proceedings of the 10th ACM Conference on Recommender Systems*, 2016, pp. 261–268.

[15] H. Chen and J. Li, "Adversarial tensor factorization for context-aware recommendation," in *Proceedings of the 13th ACM Conference on Recommender Systems*, 2019, pp. 363–367.

[16] Z. Zhu, X. Hu, and J. Caverlee, "Fairness-aware tensor-based recommendation," in *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, 2018, pp. 1153–1162.

[17] R. Harshman, "Foundations of the parafac procedure: Models and conditions for an" explanatory" multimodal factor analysis," 1970.

[18] L. Tucker, "Some mathematical notes on three-mode factor analysis," *Psychometrika*, vol. 31, no. 3, pp. 279–311, 1966.

[19] M. Nickel, V. Tresp, and H.-P. Kriegel, "A three-way model for collective learning on multi-relational data," in *Icml*, 2011.

[20] E. E. Papalexakis, C. Faloutsos, and N. D. Sidiropoulos, "Tensors for data mining and data fusion: Models, applications, and scalable algorithms," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 8, no. 2, p. 16, 2017.

[21] T. Kolda and B. Bader, "Tensor decompositions and applications," *SIAM review*, vol. 51, no. 3, 2009.

[22] H. Huang, C. Ding, D. Luo, and T. Li, "Simultaneous tensor subspace selection and clustering: the equivalence of high order svd and k-means clustering," in *Proceedings of the 14th ACM SIGKDD international conference on Knowledge Discovery and Data mining*. ACM, 2008, pp. 327–335.