Inferring Complementary Products from Baskets and Browsing Sessions

Ilya Trofimov
Yandex Market
Moscow, Russia
trofim@yandex-team.ru

ABSTRACT

Complementary products recommendation is an important problem in e-commerce. Such recommendations increase the average order price and the number of products in baskets. Complementary products are typically inferred from basket data. In this study, we propose the BB2vec model. The BB2vec model learns vector representations of products by analyzing jointly two types of data - **B**askets and **B**rowsing sessions (visiting web pages of products). These vector representations are used for making complementary products recommendation. The proposed model alleviates the cold start problem by delivering better recommendations for products having few or no purchases. We show that the BB2vec model has better performance than other models which use only basket data.

KEYWORDS

embeddings learning, multi-task learning, recommender systems, e-commerce

ACM Reference Format:
Ilya Trofimov. 2018. Inferring Complementary Products from Baskets and Browsing Sessions. In *Proceedings of Workshop on Intelligent Recommender Systems by Knowledge Transfer and Learning (RecSysKTL'18).* ACM, New York, NY, USA, 8 pages.

1 INTRODUCTION

Recommender systems in e-commerce are the "must have" technology now. These systems are used by almost all major e-commerce companies. Recommender systems help users to navigate in a vast assortment, discover new items and satisfy various tastes and needs. For online shops, such systems help to convert browsers into buyers (increase conversion rate), do cross-selling, improve users loyalty and retention [22]. These effects overall increase the revenue of an e-commerce company and improve customer experience. Worldwide retail e-commerce sales in 2016 are estimated as $1,915 Trillion and will continue to grow by 18%-25% each year [1]. The share of e-commerce in all retail in 2016 is estimated as 8.7% and continues to grow [2].

The main types of recommendations at e-commerce web site are:

- **Personalized**: "Products you may like".
- **Non-personalized**: "Similar products", "Complementary products", "Popular products".

Personalized recommendations enjoy a great body of research in recent years [2, 8, 10, 14, 20, 26]. In the same time, non-personalized recommendations are less studied. "Similar products" recommendations are typically placed on the product web page. Similar products are *substitutes* and can be purchased interchangeably. The major goal of similar products recommendation is to persuade a user to purchase by presenting him/her a diverse set of products similar to the original interest. *Complementary* products can be purchased in addition to each other. For example, an iPhone cover is complementary for iPhone, the second part of a film is complementary to the first part, etc.

When a user already has an intention to purchase - it's high time for complementary products recommendation. Such recommendations might appear on the product web page, during addition to the shopping cart and the checkout process. In the latter case, recommendations can be based on the whole content of the basket. People naturally tend to buy products in bundles. Complementary products recommendation satisfies this natural need and increases average order price as a consequence.

Online shops try to maximize the revenue by combining all the recommendation scenarios. The significance of recommendations in e-commerce can be illustrated by the following fact: in Amazon, 35% of purchases overall come from products recommendations [3].

These considerations show that complementary products recommendation is an important scenario. Manual selection of complementary products fails when the number of products in an online shop (or a marketplace) is large and assortment changes quickly. Some companies [4] provide "recommendations as a service" for online shops. These companies deliver real-time recommendations for hundreds and even thousands of shops from various business areas in a fully automated manner.

Machine learning models can solve this problem. There are two principled ways for making complementary products recommendations via machine learning. First way - is to use human assessment to generate the "ground truth" - a set of product-complement pairs. Then some machine learning model can use these pairs as positive examples and build a classifier. This classifier could be applied for identifying complements for all products. The application of this approach is limited for two main reasons. Firstly, it relies on the

[1] https://www.emarketer.com/Article/Worldwide-Retail-Ecommerce-Sales-Will-Reach-1915-Trillion-This-Year/1014369
[2] https://www.statista.com/statistics/534123/e-commerce-share-of-retail-sales-worldwide/
[3] http://www.mckinsey.com/industries/retail/our-insights/how-retailers-can-keep-up-with-consumers
[4] For example richrelevance.com, gravity.com, yoochoose.com, etc.

RecSysKTL'18, October 2018, Vancouver, Canada

"ground truth" which should be collected by human assessment. Human assessment is costly and error-prone when the number of products is large and the assortment changes quickly.

An alternative approach is inferring complementary products by analyzing user purchases (baskets) and detecting items which are *frequently purchased together*. To the best of our knowledge, it is typically done in e-commerce companies by using some heuristical co-occurrence measure (cosine similarity, Jaccard similarity, PMI). The more sophisticated way is to develop a predictive model for such co-occurrences in baskets. In this study, we follow the latter approach.

We make the following contributions in this paper:

(1) We propose the statistical model BB2vec. The model itself learns vector representations of products by analyzing jointly two types of data - baskets and browsing sessions (visiting web pages of products). These types of data are always available to any e-commerce company. We apply vector representations learned by the BB2vec model for doing recommendations of complementary products.
(2) In experimental studies, we prove that the proposed model BB2vec predicts products which are purchased together better than other methods which rely only on basket data. We show that the BB2vec model alleviates the cold-start problem by delivering better predictions for products having few or no purchases.
(3) We show how to make the BB2vecmodel scalable by selecting the specific objective function. It is an important issue since e-commerce company typically has a vast amount of browsing data, much more than basket data.

The source code of our model is publicly available [5].

2 RELATED WORK

After invention of the word2vec model [16, 17] which is dedicated to learning of distributed word representations, it was rapidly extended to other kinds of data: sentences and documents [11] (doc2vec), products in baskets [5] (prod2vec, bagged-prod2vec), nodes in graph [6] (node2vec), [19] (DeepWalk), etc. The original word2vec algorithm in its skip-gram variant proved to solve the weighted matrix factorization problem, where the matrix consists of values of shifted PMI of words [12]. The extension of the prod2vec for using items metadata was proposed in [24] (MetaProd2Vec) and for the textual data in [3]. Sequences of items selected by a user are modeled in [7] by means of the word2vec-like model. User and item representations learned by their model are applied for making recommendations.

Another associated area of research is *multi-relational learning*. Multi-relational data is a kind of data when entities (users, items, categories, time moments, etc.) are connected to each other by multiple types of relations (ratings, views, etc.) Collective matrix factorization [23] was proposed for relational learning. Each matrix corresponds to a relation between items. Latent vectors of entities are shared. Other variants of models for multi-relational learning are: RESCAL [18], TransE [1], BigARTM [25].

In computational linguistics, distributed word representations for two languages could be learned jointly via multi-task learning [9]. This approach improves machine translation.

The importance of learning-to-rank in context of recommendations with implicit feedback is discussed in [20].

2.1 The most similar studies

Sceptre [15] consider two types of relationships between products "being substitutable" and "being complementary". The "Sceptre" model builds two directed graphs for inferring these two types of relations. As a ground truth for training the model uses recommendation lists crawled from *amazon.com*. The topic model based on user reviews which also takes into account items taxonomy was used for feature generation. At the top level, the logistic regression did the classification. The application of this approach is limited for two main reasons. Firstly, it relies on "ground truth" which in the real situation should be collected by human assessment. Human assessment is costly and error-prone when the number of products is large. Secondly, it requires rich text descriptions (product reviews) for doing topic modeling.

The **prod2vec** [5] model learns product representations from basket data. These representations are used further for doing recommendations of type "Customers who bought X from vendor V_1 also bought Y from vendor V_2" which are shown alongside emails in a web interface. Our model is a combination of several prod2vec models which are learned simultaneously with partially shared parameters. We show that this combination has better predictive performance than the single prod2vec model.

P-EMB. Modeling embeddings for exponential family distributions were presented in [21]. The particular model of this family is "Poisson embeddings" (P-EMB) which was applied to model quantities of products in market baskets. Vector representations of products learned by the P-EMB model can be used for inferring substitutable and complementary products. The application of exponential distributions is orthogonal to our research. We consider the BB2vec model can be modified accordingly, i.e., learn several P-EMB models with partially shared parameters instead of prod2vec models.

CoFactor [13] improves recommendations from implicit feedback via weighted matrix factorization. The improvement is done by the regularization term which is a factorization problem of a matrix with items co-occurrences (shifted PMI) which is also used in our paper. Our research is different in three main points. Firstly, two parts of the objective (14) come from different sources of data (baskets and browsing sessions), while two parts of the objective in [13] (weighted matrix factorization term and regularization term) come from the one kind of data - user-item implicit feedback. Secondly, the objective of the BB2vec is mixed: it includes the skip-gram objective and the matrix factorization one. The procedure of inference is also different - SGD, while [13] used coordinate descent. Thirdly, we use pairwise ranking objective and show its benefits.

3 LEARNING OF WORD REPRESENTATIONS

The popular word2vec model [16, 17] in skip-gram variant learns vector representations of words which are useful for predicting the context of the word in a document. More formally, let $\{w_1, \ldots, w_T\}$

[5] https://github.com/IlyaTrofimov/bb2vec

be a sequence of words, W - their vocabulary, the context - a c-sized window of words surrounding w_t. The likelihood of such model is

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\le j\le c,\, j\ne 0}\log P(w_{t+j}|w_t) \tag{1}$$

The conditional distribution $P(w_O|w_I)$ is defined as

$$P(w_O|w_I) = \frac{\exp\left(\mathbf{v}'_{w_O}{}^{T}\mathbf{v}_{w_I}\right)}{\sum_{w\in W}\exp(\mathbf{v}'_{w}{}^{T}\mathbf{v}_{w_I})} \tag{2}$$

where $\mathbf{v}_w, \mathbf{v}'_w \in \mathbb{R}^d$ are input and output representations of words. Together they form $|W|\times d$ matrices V, V'. Vectors $\mathbf{v}_w, \mathbf{v}'_w$ are also known as *word embeddings* and have many applications in natural language processing, information retrieval, image captioning, etc. Words of similar meaning typically have similar vector representations compared by cosine similarity.

The exact minimization of (1) is computationally hard because of the softmax term. The `word2vec` algorithm is only concerned with learning high-quality vector representations and it minimizes the more simple *negative sampling* objective

$$Q^{SG}(V',V) = \frac{1}{T}\sum_{t=1}^{T}\sum_{-c\le j\le c,\, j\ne 0} L_{neg}(w_t, w_{t+j}) \tag{3}$$

$$L_{neg}(w_I, w_O) = \log\sigma(\mathbf{v}'_{w_O}{}^{T}\mathbf{v}_{w_I}) + \sum_{i=1}^{k}\mathbb{E}_{w_i\sim N(w)}\log\sigma(-\mathbf{v}'_{w_i}{}^{T}\mathbf{v}_{w_I})$$

where superscript SG stands for "skip-gram".

In the negative sampling objective, positive examples (w_I, w_O) come from the training data, while in negative samples (w_I, w_i) output words w_i come from a noise distribution $N(w)$. The noise distribution $N(w)$ is a free parameter which in the original `word2vec` algorithm is the unigram distribution to the 3/4rd power - $P^{3/4}(w)$.

The `word2vec` model in skip-gram formulation when $N(w) = P(w)$ implicitly factorizes the matrix of shifted pointwise mutual information (PMI) [12]:

$$\mathbf{v}'_{w_i}{}^{T}\mathbf{v}_{w_j} = PMI(w_i, w_j) - \log k \tag{4}$$

$$PMI(w_i, w_j) = \log\left(\frac{n_{ij}/T}{(n_i/T)(n_j/T)}\right) \tag{5}$$

where n_i is a number of times which a word w_i occurred in a document, n_{ij} - number of times which two words w_i, w_j occurred together in a c-sized window, T is length of a document. Let

$$Q^{MF}(V',V) = \frac{1}{2}\sum_{i,j}(PMI(w_i,w_j) - \log k - \mathbf{v}'_{w_i}{}^{T}\mathbf{v}_{w_j})^2$$

where superscript MF stands for "matrix factorization". Solutions of these two problems

$$\operatorname*{argmin}_{V',V} Q^{SG}(V',V) \qquad \operatorname*{argmin}_{V',V} Q^{MF}(V',V) \tag{6}$$

are equal for large enough dimensionality d of vector representations [12].

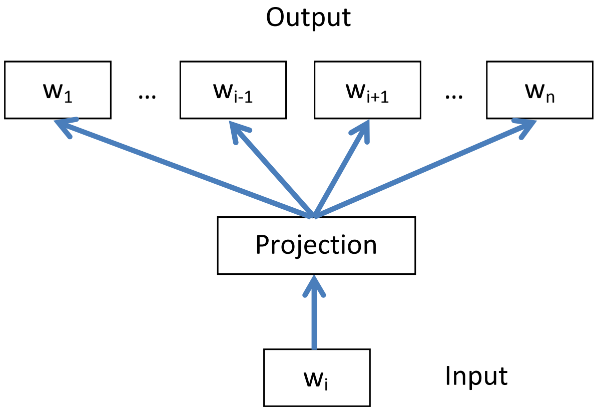


Figure 1: Skip-gram model, basket/session = $\{w_1, w_2, \ldots, w_n\}$

4 THE PROPOSED MODEL

In this section, we describe the proposed `BB2vec` model. It is a combination of `prod2vec` models which are fitted with partially shared parameters (multi-task learning).

4.1 Product recommendations with `prod2vec`

The original `word2vec` model can be easily applied to other domains by redefining the "context" and the "word". Some examples include modeling products in baskets [5], nodes in graph [6, 19], sequences of items selected by users [7], etc.

Particulary, the `prod2vec` model [5] assumes the following likelihood

$$\sum_{B\in\mathfrak{B}}\sum_{w_t\in B}\sum_{-c\le j\le c,\, j\ne 0}\log P(w_{t+j}|w_t) \tag{7}$$

where $\mathfrak{B}$ is a set of baskets.

The context C in the original `word2vec` model (1) is a c-sized windows since it is tailored for long text sequences. In our study, we analyze shorter sequences like baskets or e-commerce browsing sessions. We use a modified likelihood

$$\sum_{B\in\mathfrak{B}}\sum_{\substack{w_I, w_O\in B\\ w_I\ne w_O}}\log P(w_O|w_I). \tag{8}$$

where context is the full basket, except an input product, independently of its size (see fig. 1). This eliminates the need for selecting the window size.

Consider the product k. By definition, complementary products are frequently purchased together and have high conditional probability $P(m\,|\,k)$ [6].

The most simple way to estimate the conditional probability $P(m\,|\,k)$ is by empirical frequencies (co-counting). Let n_k be a number of baskets with item k, n_{mk} - number of baskets with both items k and m. Here and after we assume for simplicity that an item

[6] Some authors [2] propose the modified expression $\frac{P(m\,|\,k)}{P^{\alpha}(m)}$ to reduce the bias towards popular items. In this paper we use $\alpha = 0$, however, the proposed model could by extended to the general case $\alpha > 0$.

Algorithm 1: Multi-task learning of representations

```
Input   :Tasks $T_i$, weights $\lambda_i$, $i = 1, \dots, K$
for $j = 1, \dots, L$ do
    Initialize elements of matrices $V_j, V'_j$ with $N(0, 0.1)$
$W = \sum_{i=1}^{K} \lambda_i$
$\xi$ - random variable having categorical distribution
    $Cat(\frac{\lambda_1}{W}, \dots, \frac{\lambda_K}{W})$
repeat until convergence
    Sample $i$ from $\xi$   (select the task $T_i$)
    Sample object-context pair $(w_I, w_O)$ from the task $T_i$
    for $m = 1, \dots, \mid W \mid$, $n = 1, \dots, d$ do
        $V_{g(i)}^{m,n} \leftarrow V_{g(i)}^{m,n} - \eta^{m,n} \frac{\partial L_{neg}(w_I, w_O)}{\partial V_{g(i)}^{m,n}}$
        $V'^{m,n}_{g'(i)} \leftarrow V'^{m,n}_{g'(i)} - \eta'^{m,n} \frac{\partial L_{neg}(w_I, w_O)}{\partial V'^{m,n}_{g'(i)}}$
        where learning rates $\eta^{m,n}, \eta'^{m,n}$ are calculated by the
        AdaGrad rule [4].
Output:$V_j, V'_j$, for $j = 1 \dots L$
```

might appear in a basket/session only once. Then

$$P(m \mid k) \approx \frac{n_{mk}}{n_k}$$

However, counts n_{mk}, n_k are noisy for products having few purchases.

The prod2vec model can generate recommendations of complementary products without co-counting. Given the vector representations of products V, V', complementary products for the item k could be inferred by selecting items m with a high score $\mathbf{v'}_m{}^T \mathbf{v}_k$ since $P(m \mid k) \sim \exp(\mathbf{v'}_m{}^T \mathbf{v}_k)$ (2). The connection of representations V, V' to the conditional distribution $P(m \mid k)$ in the data is an intrinsic limitation of the prod2vec model. In the next section, we will show how to overcome this limitation by means of the multi-task learning and the specific objective function.

4.2 Multi-task learning

Assume that multiple types of data contain same objects. E-commerce data naturally have such data types: browsing sessions, baskets, product comparisons, search results, etc. Let $\{T_1, \dots, T_K\}$ be a list of such data types. For each data type T_i, one may fit the prod2vec model and infer latent representations of items V_i, V'_i by solving the problem

$$\min_{V'_i, V_i} Q^{SG}_{T_i}(V'_i, V_i)$$

We will refer to each learning problem as a *task*.

Then we assume that some representations are shared and the total number of distinct representations L is less then the number of tasks K. The functions $g', g : \{1, \dots, K\} \to \{1, \dots, L\}$ define which matrices are shared between tasks. For example, if $g(i) = g(j)$ then the input representations V of objects in the tasks T_i, T_j are shared. We come to the following multi-task learning problem

$$\min_{\substack{V'_j, V_j \\ j=1 \dots L}} \sum_{i=1}^{K} \lambda_i Q^{SG}_{T_i}(V'_{g'(i)}, V_{g(i)}) \tag{9}$$

Algorithm 1 describes a procedure for solving the problem (9). At the top level, the Algorithm 1 selects a task T_i with probability proportional to its weight λ_i. Then a random object-context pair is sampled and representations of items $V_{g(i)}, V'_{g'(i)}$ are updated via SGD with AdaGrad learning rates.

4.3 Motivation of multi-task representations learning

Optimizing the objective (9) is an example of *multi-task learning* (MTL). Multi-task learning is about solving several related learning problems simultaneously. The problems with not enough data may benefit from coupling parameters with other problems. However, multi-task learning may worsen a predictive performance. Avoiding it requires choosing a proper way of binding tasks together.

Consider a problem of *complementary* items recommendation in e-commerce. By definition, complementary items are those which could be purchased in addition to each other. As we explained in the section 4.1, the prod2vec model learns representations V'_B, V_B of products by solving the problem

$$V'_B, V_B = \operatorname*{argmin}_{V', V} Q^{SG}_{baskets}(V', V) \tag{10}$$

and products having high score $\mathbf{v'}_{m,B}{}^T \mathbf{v}_{k,B}$ are complementary.

This predictive model can be improved further by analyzing another source of data - browsing sessions. The probability of purchase in a session is typically 1%-5%. Thus, an online shop has much more browsing data then purchasing data. The prod2vec model can learn product representations from browsing data by solving the problem

$$V'_S, V_S = \operatorname*{argmin}_{V', V} Q^{SG}_{brows.}(V', V) \tag{11}$$

It is generally accepted that products which are frequently viewed in the same sessions are *similar.* Let S be a browsing session. Two items m, k having high conditional probability $P(m \in S \mid k \in S)$ are similar.

We conclude that conditional distributions $P(m \in B \mid k \in B)$ and $P(m \in S \mid k \in S)$ are different which leads to different representations V'_B, V_B vs. V'_S, V_S. In the same time, the general property of the prod2vec and other extensions of the word2vec model is that similar objects have similar representations.

In our research, we use the specific objective function for learning product representations where input V_B and output V'_B representations are shared separately between tasks

$$\min_{V'_B, V_B, V'_S, V_S} Q^{SG}_{bask.}(V'_B, V_B) + \lambda \left(Q^{SG}_{browse}(V'_S, V_B) + Q^{SG}_{browse}(V'_B, V_S) \right) \tag{12}$$

Addition of $\lambda Q^{SG}_{browse}(V'_S, V_B)$ forces V_B to be a good input representation for modeling similar products, particularly forcing representations of similar products V_B to be close. The same holds for output representation V'_B because of the term $\lambda Q^{SG}_{browse}(V'_B, V_S)$.

4.4 Ranking

Learning of vector representations in skip-gram variant (3) is considered to be a binary classification problem when positive object-context examples are drawn from the training dataset, and negative examples are drawn from the noise distribution. Since our final goal is to use representations for generating top-N recommendations one may reformulate the problem as learning to rank. We can easily transform a skip-gram negative sampling problem (3) into the pairwise ranking problem.

In this formulation we treat w_I as a *query item* and force our model to give positive example w_O higher rank than negative examples $w_i \sim N(w)$. The value $\mathbf{v}'_{w_i}{}^T \mathbf{v}_{w_I}$ is the ranking score:

$$Q^{SG}(V', V) = \sum_{\substack{w_I, w_O \in B \\ w_I \neq w_O}} \sum_{i=1}^{k} \mathbb{E}_{w_i \sim N(w)} \log(\sigma((\mathbf{v}'_{w_O}{}^T - \mathbf{v}'_{w_i}{}^T)\mathbf{v}_{w_I})) \tag{13}$$

We can show (see Appendix A) that optimizing (13) with respect to V, V' in the general case $N(w) = P^{\alpha}(w)$ leads to the solution

$$(\mathbf{v}'_r - \mathbf{v}'_m)^T \mathbf{v}_k = \log\left(\frac{P(r \mid k)}{P^{\alpha}(r)}\right) - \log\left(\frac{P(m \mid k)}{P^{\alpha}(m)}\right)$$

Thus, if $\mathbf{v}'_r{}^T \mathbf{v}_k > \mathbf{v}'_m{}^T \mathbf{v}_k$ then $\frac{P(r \mid k)}{P^{\alpha}(r)} > \frac{P(m \mid k)}{P^{\alpha}(m)}$. Interestingly, the number of negative samples k vanished and doesn't introduce a shift like in the classification setting (5). For ranking, it can be considered as a hyperparameter of training. We will use further the learning to rank (13) variant of negative sampling problem.

4.5 Improving computational performance of the BB2vec model

Since there are much more browsing data than basket data, solving the skip-gram problems $Q^{SG}_{browse}(\cdot)$ requires much more SGD updates in the Algorithm 1 than in the original problem $Q^{SG}_{bask.}(\cdot)$ and significantly slows down the program.

The software implementation of the BB2vec model solves matrix factorization problems $Q^{MF}_{browse}(\cdot)$ instead of the skip-gram problems $Q^{SG}_{browse}(\cdot)$ for browsing sessions since their solutions are asymptotically equal (6). As a result, the software implementation optimizes the following objective

$$\min_{V'_B, V_B, V'_S, V_S} Q^{SG}_{bask.}(V'_B, V_B) + \lambda \left(Q^{MF}_{browse}(V'_S, V_B) + Q^{MF}_{browse}(V'_B, V_S) \right) \tag{14}$$

The matrix factorization problems are solved by the stochastic matrix factorization [10] with AdaGrad learning rates [4].

The complexity of one epoch for solving $Q^{SG}_{browse}(\cdot)$ is

$$O(\#sessions \cdot avg.session\,size \cdot \#neg.samples \cdot d)$$

while the complexity of one epoch of the stochastic matrix factorization of the shifted PMI matrix is

$$O(\#entries\ in\ SMPI\ matrix \cdot d)$$

For our datasets, the complexity of solving the stochastic matrix factorization of the shifted PMI matrix is roughly 10^3 times less.

5 EXPERIMENTS

In computational experiments, we compared different models for complementary products recommendation. We measured how well each model predicts products which are purchased together in baskets at the hold-out data.

5.1 Datasets

To evaluate the performance of the BB2vec and compare it with baselines we used 4 datasets with user behaviour log on e-commerce web sites (see Table 1):

1. **RecSys'15** - the dataset from the ACM RecSys 2015 challenge [7]. The challenge data has two main parts: clicks on items (equivalent to visiting products web pages) and purchases. We consider a basket to be a set of items purchased in a session. Products with less then 10 purchases in the whole dataset were removed. We randomly split sessions to train, validation and test in the proportion 70% / 15% / 15%.
2. **RecSys'15 - 10%** - is the modification of the **RecSys'15** dataset. The only difference is that the training dataset was replaced by it's 10% subsample, while validation and test were left unchanged.
3. **CIKM'16** - is the dataset from CIKM Cup 2016 Track 2 [8]. We took browsing log (product page views) and transactions (baskets). We randomly split sessions to train, validation and test in proportion 70% / 15% / 15%. All baskets have session identifier and were split accordingly.
4. **CIKM'16, Categories** - is a modifications of **CIKM'16** dataset. Since **CIKM'16** dataset is very sparse, we replaced product identifiers with their category identifiers. The number of distinct categories is much less than the number of products. Thus, dataset becomes denser.

Finally, for each dataset, we left at test and validation parts only products having at least one purchase or a view at the training dataset. Otherwise, methods under considerations can't learn vector representations for such products. All the purchase counts and the view counts were binarized.

In computational experiments, PMI matrices were calculated using browsing data from train datasets only. For RecSys'15, RecSys'15, 10% datasets the cells of PMI matrix having $n_{ij} \geq 10$ were kept. For CIKM'16, CIKM'16 Category the cells of PMI matrix having $n_{ij} \geq 3$ were kept.

5.2 Evaluation

We evaluated the performance of the models by predicting items which are purchased together in baskets at hold-out data. We used two measures: average HitRate@K and NDCG@K. For each basket B, consider all distinct pairs of items $k, m \in B$. The goal is to predict the second item m in the pair given the first one k (query item). Denote L_k, a list of length K with complementary items recommendations for the query item k.

[7] http://2015.recsyschallenge.com/challenge.html

[8] https://competitions.codalab.org/competitions/11161

Table 1: Datasets.

Dataset	Sessions	Baskets			Items	
		train	val/test	w/ ≥ 2 items	total	w/o purch. at train
RecSys'15	9,249,729	346,554	74,161	49%	7226	0%
RecSys'15 - 10%	9,249,729	34,753	74,161	49%	7226	17%
CIKM'16	310,486	9380	2063	19%	11244	11%
CIKM'16, Categories	310,486	9380	2063	17%	750	5%

Then

$$\text{HitRate@K} = [m \in L_k]$$

$$\text{NDCG@K} = \frac{[m \in L_k]}{\log_2(pos(m, L_k) + 1)}$$

where $[\cdot]$ is an indicator function, $pos(m, L_k)$ is the position of item m in the list L_k. These performance measures are averaged

- over all distinct pairs of items (k, m) in all the baskets;
- or over a subset of pairs of items (k, m) with query item k having a particular number of purchases at the training dataset.

5.3 Models

In computational experiments, we evaluated the performance of the proposed BB2vec model. This model learns vector representations of products from baskets and browsing sessions by optimizing the objective (12). We evaluated two variants: with classification (`BB2vec, class.`) and ranking (`BB2vec, ranking`) objectives.

We compared the proposed BB2vec model with the following baselines:

(1) `Popularity`: items are sorted by purchases count at train.
(2) `Co-counting`: for the query item k top items with joint purchases n_{km} are selected. If for two items $m, r : n_{mk} = n_{rk}$ then the item with larger purchase count had higher rank.
(3) `prod2vec`. We implemented the `prod2vec` algorithm [5] and fit item representations from basket data. The context in the skip-gram objective always was all items in a basket except the input item. It is equivalent to setting window size in `prod2vec` to the size of the largest basket. Thus the model optimizes the objective (10).

Validation datasets were used for early stopping. The initial learning rate in AdaGrad was set to 0.05. The number of negative samples was set to 20. All hyperparameters of the methods (latent vectors dimensionality, mixing parameters λ) of were tuned for maximizing HitRate@10 at validation datasets and final results in the table 2 are calculated at test datasets (see Appendix B). In the `prod2vec` and `BB2vec` models negative items were sampled uniformly.

For the predictive models `prod2vec` and `BB2vec` we used matrices V'_B, V_B for generating recommendations. Given a query item k, top N items by the score $\mathbf{v}'_{m,B}{}^T \mathbf{v}_{k,B}$ are selected.

5.4 Results

Firstly, we analyze the overall performance of the methods under evaluation. Table 2 show the average HitRate and NDCG over test datasets. We conclude that the proposed `BB2vec` model has the best predictive performance among almost all the datasets and performance measures. The difference between the models `BB2vec, class.` vs. `prod2vec` is the effect of learning of shared product representations for both purchases and browsing sessions, instead of representations for purchases only. The variant with the learning-to-rank objective `BB2vec, rank.` is better than the classification one `BB2vec, class.` The gap between `BB2vec` and other models is the most sound at sparse datasets CIKM'16, CIKM'16 Categories, RecSys'15 10%.

Secondly, we analyze in how the `BB2vec` model alleviates the cold start problem. Fig. 2 shows the breakdown of average HitRate@10 by product purchase count at train dataset. The `BB2vec` model performs better for products having few or no purchases. In the same time, for products having enough purchases (≥ 4 for CIKM dataset, ≥ 16 for other datasets) the `BB2vec` does not generate better recommendations than other models. The notorious feature of the `BB2vec` model is that is can make recommendations for products with no purchases at the training set. We conclude that the improvement of the overall performance comes from better recommendations for products having few or no purchases at the training dataset.

6 CONCLUSIONS

In this paper, we proposed the `BB2vec` model for complementary products recommendation in e-commerce. The model learns product representations by processing simultaneously two kinds of data - baskets and browsing sessions which are the very basic ones and always available to any e-commerce company. Despite the large amount of browsing sessions, the learning algorithm is computationally scalable.

The predictive performance of the `BB2vec` model is better than the performance of state-of-the-art models which rely solely on basket data. We show that the improvement comes from better recommendations for products having few or no purchases. Thus, the proposed model alleviates the cold start problem. Unlike other models, the `BB2vec` model can generate recommendations for products having no purchases.

Our model can be extended in two ways. Firstly, one can use other algorithms as tasks in the multi-task objective, for example the `P-EMB` model. Secondly, our model can be extended to handle items metadata.

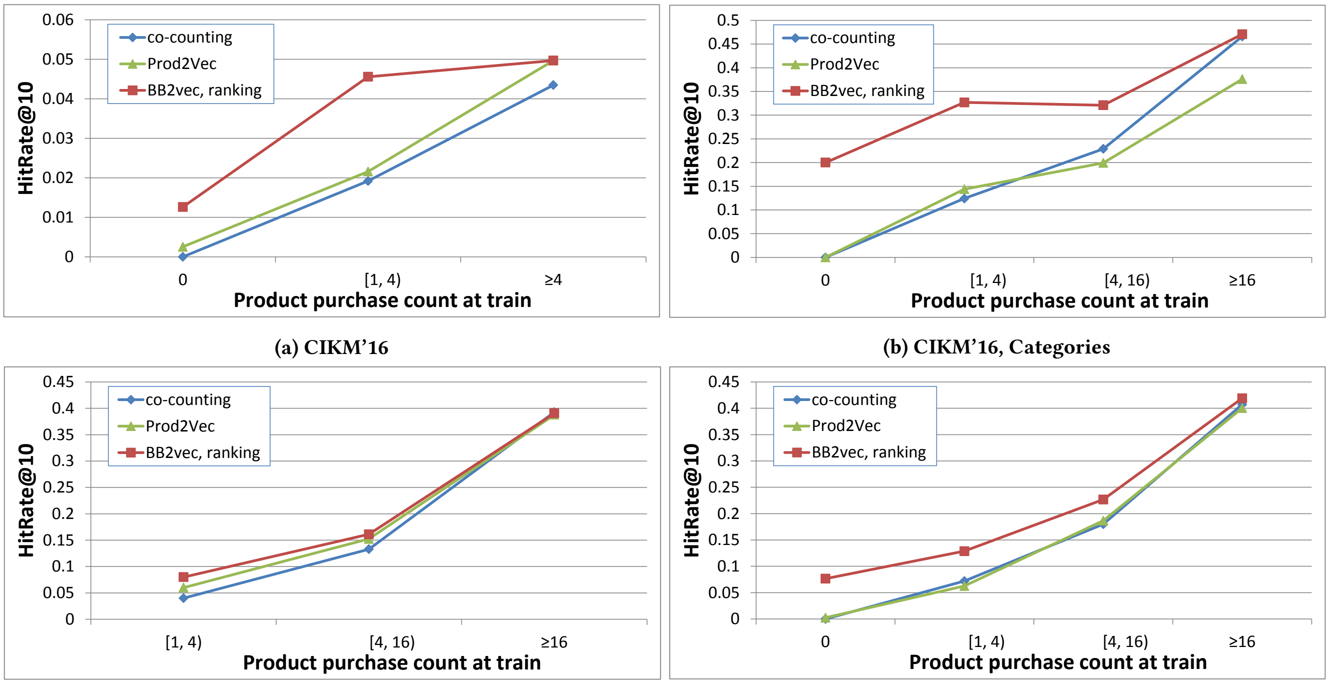
ACKNOWLEDGMENTS

The author thanks Konstantin Bauman and Pavel Serdyukov for their useful comments and proofreading.

Table 2: Experimental results.

Method	CIKM’16				CIKM’16, Categories			
	HitRate@		NDCG@		HitRate@		NDCG@	
	10	50	10	50	10	50	10	50
Popularity	0.006	0.047	0.006	0.024	0.201	0.482	0.174	0.305
Co-counting	0.015	0.016	0.023	0.024	0.391	0.477	0.531	0.574
Prod2Vec	0.018	0.021	0.025	0.026	0.322	0.510	0.467	0.547
BB2vec, class.	0.027	0.039	**0.036**	0.042	0.354	0.609	0.483	0.598
BB2vec, ranking	**0.029**	**0.077**	0.029	**0.051**	**0.430**	**0.659**	**0.559**	**0.665**

Method	RecSys’15				RecSys’15, 10%			
	HitRate@		NDCG@		HitRate@		NDCG@	
	10	50	10	50	10	50	10	50
Popularity	0.036	0.127	0.031	0.071	0.036	0.128	0.031	0.072
Co-counting	**0.383**	0.569	**0.475**	0.561	0.333	0.453	0.419	0.475
Prod2Vec	0.379	0.585	0.461	0.557	0.329	0.411	0.400	0.440
BB2vec, class.	**0.383**	0.593	0.464	0.562	0.351	0.505	0.420	0.493
BB2vec, ranking	**0.383**	**0.597**	0.465	**0.564**	**0.356**	**0.559**	**0.425**	**0.519**



(a) CIKM’16

(b) CIKM’16, Categories

(c) ACM RecSys’15

(d) ACM RecSys’15, 10%

Figure 2: Model performance vs. product purchase count

A RANKING LOSS FUNCTION

Consider a basket B, input item $k \in B$, output item $m \in B$ and negative samples $r \sim P^{\alpha}(w)$. Then the negative sampling objective for learning-to-rank variant equals

$$Q^{SG}(V, V') = \sum_{B \in \mathcal{B}} \sum_{\substack{k, m \in B \\ k \neq m}} \mathbb{E}_{r \sim P^{\alpha}(w)} \log(1 + \exp((\mathbf{v}'_r - \mathbf{v}'_m)^T \mathbf{v}_k))$$

Let $P(k, m)$ be the empirical distribution of item pairs (k, m) in the baskets $\mathcal{B}$, $a_{rmk} = (\mathbf{v}'_r - \mathbf{v}'_m)^T \mathbf{v}_k$, $N = \sum_{B \in \mathcal{B}} |\{k, m \in B, k \neq m\}|$.

Then

$$\begin{aligned}
\frac{1}{N}Q^{SG}(V, V') &= \sum_{k,m,r} P(k,m)P^{\alpha}(r)\log(1 + \exp((\mathbf{v}'_r - \mathbf{v}'_m)^T \mathbf{v}_k)) \\
&= \sum_{k,m,r} P(k)P(m|k)P^{\alpha}(r)\log(1 + \exp((\mathbf{v}'_r - \mathbf{v}'_m)^T \mathbf{v}_k)) \\
&= \sum_{k,m,r} P(k)P(m|k)P^{\alpha}(r)\log(1 + \exp(a_{rmk})) \\
&= \frac{1}{2}\sum_{k,m,r} P(k)P(m|k)P^{\alpha}(r)\log(1 + \exp(a_{rmk})) \\
&\quad + P(k)P(r|k)P^{\alpha}(m)\log(1 + \exp(-a_{rmk})) \\
&= \frac{1}{2}\sum_{k,m,r} P(k)f(a_{rmk})
\end{aligned}$$

here we defined the function

$$f(a) = P(m|k)P^{\alpha}(r)\log(1+\exp(a)) + P(r|k)P^{\alpha}(m)\log(1+\exp(-a))$$

and used the identity $a_{mrk} = -a_{rmk}$.

The derivative of $f(a)$ is

$$f'(a) = \frac{1}{1 + \exp(a)}(P(m|k)P^{\alpha}(r)\exp(a) - P(r|k)P^{\alpha}(m))$$

By solving the equation $f'(a_{rmk}) = 0$ we obtain

$$a_{rmk} = \log\left(\frac{P(r|k)}{P^{\alpha}(r)}\right) - \log\left(\frac{P(m|k)}{P^{\alpha}(m)}\right)$$

Recalling than $a_{rmk} = (\mathbf{v}'_r - \mathbf{v}'_m)^T \mathbf{v}_k$ we conclude that ranking by the score $\mathbf{v}'^{T}_r \mathbf{v}_k$ is equivalent to the ranking by $P(r|k)/P^{\alpha}(r)$.

B BEST HYPERPARAMETERS OF THE MODELS

Dataset / Model	prod2vec	BB2vec
CIKM'16	$d = 400$	$d = 100, \lambda = 2$
CIKM'16 Categories	$d = 200$	$d = 100, \lambda = 8$
RecSys'15	$d = 100$	$d = 200, \lambda = 8$
RecSys'15, 10%	$d = 100$	$d = 400, \lambda = 32$

REFERENCES

[1] Antoine Bordes, Nicolas Usunier, Jason Weston, and Oksana Yakhnenko. 2013. Translating Embeddings for Modeling Multi-Relational Data. *Advances in NIPS* 26 (2013), 2787–2795. arXiv:arXiv:1011.1669v3

[2] Mukund Deshpande and George Karypis. 2004. Item-based top-n recommendation algorithms. *ACM Transactions on Information Systems (TOIS)* 22, 1 (2004), 143–177.

[3] Nemanja Djuric, Hao Wu, Vladan Radosavljevic, Mihajlo Grbovic, and Narayan Bhamidipati. 2015. Hierarchical neural language models for joint representation of streaming documents and their content. In *Proceedings of the 24th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 248–255.

[4] John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research* 12, Jul (2011), 2121–2159.

[5] Mihajlo Grbovic, Vladan Radosavljevic, Nemanja Djuric, Narayan Bhamidipati, Jaikit Savla, Varun Bhagwan, and Doug Sharp. 2015. E-commerce in Your Inbox : Product Recommendations at Scale Categories and Subject Descriptors. In *SIGKDD 2015: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 1809–1818. arXiv:1606.07154

[6] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 855–864.

[7] Elie Guàrdia-Sebaoun, Vincent Guigue, and Patrick Gallinari. 2015. Latent trajectory modeling: A light and efficient way to introduce time in recommender systems. In *Proceedings of the 9th ACM Conference on Recommender Systems*. ACM, 281–284.

[8] Yifan Hu, Yehuda Koren, and Chris Volinsky. 2008. Collaborative filtering for implicit feedback datasets. In *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*. Ieee, 263–272.

[9] Alexandre Klementiev, Ivan Titov, and Binod Bhattarai. 2012. Inducing crosslingual distributed representations of words. (2012).

[10] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 42, 8 (2009).

[11] Qv Le and Tomas Mikolov. 2014. Distributed Representations of Sentences and Documents. *International Conference on Machine Learning - ICML 2014* 32 (2014), 1188–1196. arXiv:1405.4053

[12] Omer Levy and Yoav Goldberg. 2014. Neural Word Embedding as Implicit Matrix Factorization. In *Advances in Neural Information Processing Systems (NIPS)*. 2177–2185. arXiv:1405.4053

[13] Dawen Liang, Jaan Altosaar, Laurent Charlin, and David M. Blei. 2016. Factorization meets the item embedding: regularizing matrix factorization with item co-occurrance. In *ACM RecSys*. arXiv:arXiv:1602.05561v1

[14] Greg Linden, Brent Smith, and Jeremy York. 2003. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing* 7, 1 (2003), 76–80. arXiv:69

[15] Julian McAuley, Rahul Pandey, and Jure Leskovec. 2015. Inferring networks of substitutable and complementary products. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 785–794.

[16] Tomas Mikolov. 2013. Learning Representations of Text using Neural Networks (Slides). In *NIPS Deep Learning Workshop*. 1–31. arXiv:arXiv:1310.4546v1

[17] Tomas Mikolov, Greg Corrado, Kai Chen, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. *Proceedings of the International Conference on Learning Representations (ICLR 2013)* (2013), 1–12. arXiv:arXiv:1301.3781v3

[18] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2011. A three-way model for collective learning on multi-relational data. In *Proceedings of the 28th international conference on machine learning (ICML-11)*. 809–816.

[19] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 701–710.

[20] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian Personalized Ranking from Implicit Feedback. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*. 452–461. arXiv:1205.2618

[21] Maja Rudolph, Francisco Ruiz, Stephan Mandt, and David Blei. 2016. Exponential Family Embeddings. In *Nips*. 478–486. arXiv:1608.00778

[22] J Ben Schafer, Joseph Konstan, and John Riedl. 1999. Recommender systems in e-commerce. In *Proceedings of the 1st ACM conference on Electronic commerce*. ACM, 158–166.

[23] Ajit P Singh and Geoffrey J Gordon. 2008. Relational learning via collective matrix factorization. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 650–658.

[24] Flavian Vasile, Elena Smirnova, and Alexis Conneau. 2016. Meta-Prod2Vec - Product Embeddings Using Side-Information for Recommendation. 0–7. arXiv:1607.07326

[25] Konstantin Vorontsov, Oleksandr Frei, Murat Apishev, Peter Romov, Marina Suvorova, and Anastasia Yanina. 2015. Non-bayesian additive regularization for multimodal topic modeling of large collections. In *Proceedings of the 2015 Workshop on Topic Models: Post-Processing and Applications*. ACM, 29–37.

[26] Chong Wang and David M Blei. 2011. Collaborative topic modeling for recommending scientific articles. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 448–456.