Learning Item-Interaction Embeddings for User Recommendations

Xiaoting Zhao*, Raphael Louca*, Diane Hu, Liangjie Hong
Etsy, Inc
New York, U.S.A
{xzhao,rlouca,dhu,lhong}@etsy.com

ABSTRACT

Industry-scale recommendation systems have become a cornerstone of the e-commerce shopping experience. For Etsy, an online marketplace with over 50 million handmade and vintage items, users come to rely on personalized recommendations to surface relevant items from its massive inventory. One hallmark of Etsy's shopping experience is the multitude of ways in which a user can interact with an item they are interested in: they can view it, favorite it, add it to a collection, add it to cart, purchase it, etc. We hypothesize that the different ways in which a user interacts with an item indicates different kinds of intent. Consequently, a user's recommendations should be based not only on the item from their past activity, but also the way in which they interacted with that item. In this paper, we propose a novel method for learning interaction-based item embeddings that encode the co-occurrence patterns of not only the item itself, but also the interaction type. The learned embeddings give us a convenient way of approximating the likelihood that one item-interaction pair would co-occur with another by way of a simple inner product. Because of its computational efficiency, our model lends itself naturally as a candidate set selection method, and we evaluate it as such in an industry-scale recommendation system that serves live traffic on Etsy.com. Our experiments reveal that taking interaction type into account shows promising results in improving the accuracy of modeling user shopping behavior.

CCS CONCEPTS

• **Computer systems organization** → **Content ranking**; *Web log analysis*; *Personalization*; *Query representation*; *Document representation*;

KEYWORDS

Embeddings, Candidate Set Selection, Recommendation

ACM Reference Format:
Xiaoting Zhao*, Raphael Louca*, Diane Hu, Liangjie Hong. 2019. Learning Item-Interaction Embeddings for User Recommendations. In *Proceedings of DAPA 2019 WSDM Workshop on Deep matching in Practical Applications (DAPA '19).* ACM, New York, NY, USA, Article 4, 6 pages.

*These authors have equally contributed to this work.

DAPA '19, February 15th, 2019, Melbourne, Australia

1 INTRODUCTION

As online shopping becomes more prevalent, and inventory grows at an exponential scale, customers have come to rely on personalized recommendation systems to understand their preferences and surface relevant items to them. For Etsy, an online, handmade marketplace with over 50 million active items, recommendation systems become even more critical to helping customers identify items of interest amidst Etsy's vast breadth of one-of-a-kind listings.

One hallmark of Etsy's shopping experience is the multitude of ways in which a user can interact with an item they are interested in: they can view it, favorite it, add it to a collection, add it to cart, or purchase it. We hypothesize that the different ways in which a user interacts with an item indicates different kinds of intent. For example, a user who views an item must have different intent than a user who adds the same item to their cart. Thus, the two users should be shown different recommendations, despite the fact that they both interacted with the same item. Figure 1a shows an example target item, with potential recommendations for the user who *viewed* that item (Figure 1b) versus for the user who *carted* that item (Figure 1c). Not only are the recommendations different, but the first shows recommendations that look more like substitutes to the target item, while the second shows recommendations that are more complementary.

In this paper, we propose a novel method for learning interaction-based item embeddings that encode the co-occurrence patterns of not only the item itself, but also the way in which a user interacts with them. In contrast to previous applications of embedding models, we learn multiple embeddings for each item, one for every possible item-interaction pair. These learned embeddings give us a convenient way of approximating the likelihood that one item-interaction pair would co-occur with another during a shopping session by way of a simple inner product. As such, we can predict not only *which* items a user may be interested in, but also *how* they will interact with them. Because of its computational efficiency, our model lends itself naturally as a candidate set selection process: we can generate user-specific candidate sets by finding items that lie closest to the user's past item-interaction activity in the embedded space, as visualized in Figure 2.

Our proposed method can be seen as a generalization of using co-occurrence counts, a popular approach for candidate set selection in the past [10, 11]. The underlying concept there assumes that if a pair of items has been viewed or purchased together within a short amount of time by the same user, there's a good chance the two items are related. However, this method (1) does not consider the different ways in which a customer can interact with items, usually focusing only on co-purchases, and (2) requires items to have been explicitly co-purchased together, leading to low coverage. Our

proposed method is more flexible and generalizes beyond explicit co-occurrence counts, with the ability to give recommendations along the lines of "Because you X this, you may also want to Y that", where X and Y are any interaction types.

We evaluate our model as a user-specific candidate set selection method in an end-to-end production recommendation system that serves live traffic on Etsy.com. We compare our model against a live production system, which uses a co-occurrence based candidate set and provide both offline *hit rate* metrics, as well as online key business metrics. Our experiments show that encoding interaction type in our item embeddings results in improved modeling accuracy. It also allows for interpretable visualization of a user's shopping behavior and can serve recommendations that offer more explainability. In the following sections, we describe related lines of work (Section 2), describe the proposed model (Section 3), and discuss offline and online experiment results (Section 4).

2 RELATED WORK

There are two broad areas of research that closely relate to our line of work, namely (1) candidate set selection approaches and (2) the application of neural embedding models to search and ranking problems. To confine the scope of this discussion, we focus particularly on the task of recommending items to users.

2.1 Candidate Set Selection

While candidate set selection has always been an important component of large-scale recommendation systems, it is often overlooked in favor of discussing the machine learned re-ranking model. Many previous works depend on basic heuristics that measure goodness of match between the user and items based on product category or taxonomy matching schemes, or simple popularity rankings.

Beyond basic heuristics, co-occurring signals have been a popular method for simple and efficient candidate set selection. Amazon tracks pairs of items that are frequently co-purchased by the same customers and constructs candidate sets by retrieving items that have been frequently co-purchased with a customer's last viewed items [10]. Pinterest's related pin recommendations system selects candidate pins based on *board co-occurrence*, the number of times a pin has been pinned to the same board [11]. Other candidate set models include variations on the random walk [5], as well as collaborative filtering or deep network approaches that incorporate user's historical interactions [4]. Another class of algorithms come from IR, utilizing fast query-document retrieval engines to match user and item features [1, 3]. While these methods are undoubtedly successful, our candidate set selection method explicitly considers the target item to generate candidate sets based on interaction type, whereas other methods do not distinguish between target items in a user's past history. This allows our method to be more transparent and interpretable to the user.

2.2 Neural Language Models

In its original form, neural language models such as continuous bag-of-words (CBOW) and skip-gram (SG) models learn semantically meaningful, low-dimensional representation of words by modeling patterns of co-occurring words in sentences [13]. In recent years, extending these neural embedding models to applications outside of the NLP domain has been gaining in popularity, making appearances in many domains including search, recommendations, and e-commerce [6, 7, 9, 14]. Just as word embeddings can be trained by treating a sequence of words in a sentence as context, item embeddings can be trained in a similar fashion by treating a sequence of user actions as context and learning low-dimensional embeddings for each item. Our proposed method is very similar to this line of work; however, we explicitly model different types of interactions in addition to the item itself.

3 PROPOSED METHOD

In the following sections, we describe a model that allows us to efficiently encode the ways in which users interact with items, and their co-occurrence relationships with other similar items. We show how to learn multiple interaction-based embeddings for each item, generate candidate sets, and produce recommmendations for users.

3.1 Item-Interaction Embeddings

Our proposed model is based on word2vec, a popular method in natural language processing (NLP) for learning a semi-supervised model to discover semantic similarity across words in a corpus using an unlabelled body of text [13]. This is done by relating co-occurrence of words and relies on the assumption that words appearing together are more related than words that are far apart.

The same method can be used to model user interactions at Etsy by modeling users' journeys in aggregate as a sequence of user engagements on listings, i.e., viewed, favored, add-to-carted or purchased. Each user session is analogous to a sentence, and each user action is analogous to a word in NLP word2vec parlance. This method of modeling interactions allows us to represent items or

(a) Target item

(b) View-interaction embeddings

(c) Cart-interaction embeddings

Figure 1: Given a target item ℓ shown in (a), we visualize the top 5 nearest neighbors based on ℓ's view-based embedding (b) and ℓ's cart-based embedding (c). One can see that the recommendations for a user who has viewed item ℓ should be different than the recommendations for a user who has already carted item ℓ.

other entities (i.e., shops, users, queries) as low dimensional continuous vectors, where the similarity across two different vectors represents their co-relatedness. Semantic embeddings are agnostic to the content of items such as their titles, tags, descriptions, and allow us to leverage aggregate user interactions on the site to extract items that are semantically similar.

Let $\mathcal{L}$ denote the set of items and $\mathcal{I}$ the set of interaction types a user can have with an item (e.g., view, favorite, add-to-cart, purchase). Each user's visit on Etsy triggers a *session* $S = \{p_1, \ldots, p_k\}$, which consists of a sequence of item-interaction pairs $p_j \in \mathcal{L} \times \mathcal{I}$. For example, the sequence

$$(\ell_1, \text{view}), (\ell_2, \text{favorite}), (\ell_1, \text{purchase})$$

specifies that a user first viewed item ℓ_1, then favorited item ℓ_2, and lastly purchased item ℓ_1. The training data consists of such sequences collected from multiple users over a set period of time.

The skip-gram model uses a single item-interaction pair to predict the output of $2m$ neighboring pairs, where m is a hyperparameter of the model. In particular, given an item-interaction pair $p_i \in \mathcal{L} \times \mathcal{I}$, the probability of observing the pair p_{i+j} is given by

$$\mathbb{P}(p_{i+j} \mid p_i) = \frac{u_{p_i}^\top v_{p_{i+j}}}{\sum_{p=1}^{n} u_{p_i}^\top v_p}, \quad -m \le j \le m, \tag{1}$$

where $u_p, v_p \in \mathbf{R}^d$ are the input and output vector representations of the pair p, repsectively and n is the number of unique item-interaction pairs. Implicit in the skip-gram model is the assumption that the dimension $d << n$. The objective is to maximize the function

$$\sum_{S \in \mathcal{S}} \sum_{p_i \in S} \sum_{-m \le j \le m} \mathbb{P}(p_{i+j} \mid p_i), \tag{2}$$

where $\mathcal{S}$ denotes the set of all sessions in the training data. It follows from (1) and (2) that item-interaction pairs with similar contexts will have "similar" vector representations. The objective function is optimized using stochastic gradient ascent. In practice, however, the computation of an optimal solution, can be computationally expensive because the size of the ambient dimension space $|\mathcal{L}| \times |\mathcal{I}|$ can be prohibitively large. To account for this, we use the negative sampling approach proposed in [13]. We provide more details for the approach we take to generate negative samples in section 4.1.1.

3.2 Candidate Set Selection

A crucial task for industry-scale recommendation systems is its ability to quickly retrieve a small set of relevant items out of a large set (potentially in the range of hundreds of millions) of candidate items. This is necessary as it is computationally infeasible to apply machine-learned models over the entire collection of candidate items. In practice, this is referred to as the *candidate set selection* phase, which aims to quickly prune irrelevant items while retrieving items that are likely to be relevant to the user at low cost. The smaller set is then re-ranked by a (typically, more sophisticated) machine learning model.

The learned embeddings described above now give us a convenient way of encoding co-occurrence patterns between items and the way users interact with them. A nice property is that the inner product between two such embeddings should approximate the likelihood that one item-interaction pair would co-occur with another. Because of its computational efficiency, it is easy to approximate the affinity between tens of millions of pairs of embedding vectors, lending itself naturally as a candidate set selection solution. For example, to answer a question such as "since a user viewed on item A, what is an item they may add to cart next?", we can simply find the nearest "cart" embedding to item A's "view" embedding. A user-specific candidate set can then be generated by finding the closest listings to each of the user's past item-interaction pairs in the embedded space. Figure 2 explains this idea visually.

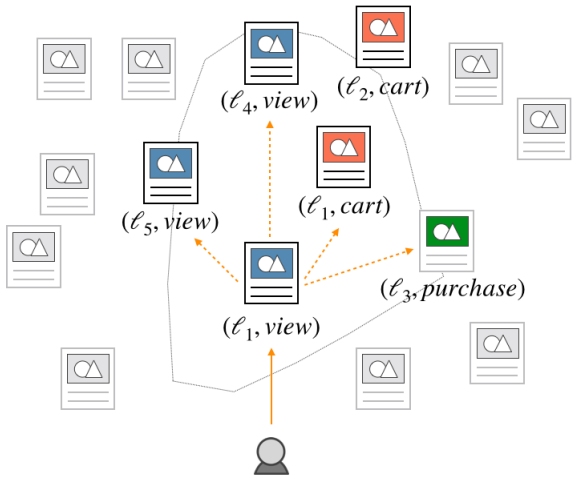


Figure 2: Example of the item-interaction pair embedding space. Given a user's last action (e.g. viewed listing ℓ_1), the top K nearest neighbors should indicate *which* item the user is likely to interact with, as well as *how* the user will interact with it (e.g. the user may *view* items ℓ_4 or ℓ_5; *cart* items ℓ_1 or ℓ_2; or *purchase* item ℓ_3)

4 EXPERIMENTS

In this section, we investigate the strengths and limitations of the proposed approach and discuss evaluation results on a dataset of browsing sessions from Etsy.

4.1 Dataset

The training data we use spans a one year period of visit log collected from Nov 2017 to Oct 2018, and is extracted from implicit feedback collected from users visiting Etsy during that period. In particular, each training instance is defined by a user's session and consisted of a sequence of item-interaction pairs sorted chronologically. We restrict attention to sessions which had more than three item-interaction pairs to eliminate bounces. The resulting dataset has about 30 billion words from over 200 millions distinct tokens.

4.1.1 Negative Sampling. As discussed in section 3.1 the model we propose uses the negative sampling approach introduced in [13] to facilitate computational efficiency in model training and improve

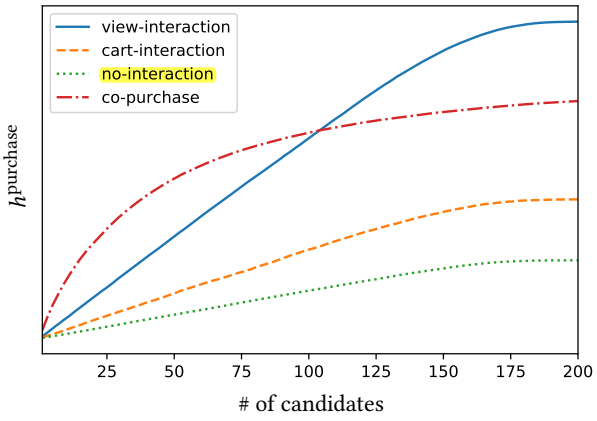


Figure 3: Comparison of average hit rates among different candidate sets with the set of purchased items.

the quality of the low-dimensional vector representations. As the initial step in the implementation of negative sampling, we define the following partial order on the set of interactions we consider:

$$\text{purchase} > \text{add-to-cart} > \text{favorite} > \text{view}.$$

For each item-interaction pair $p = (\ell, i)$ of a given session S, we add negative samples $p_j = (\ell, j)$, for all interactions $j > i$, provided that the pair $p_j \notin S$. For example, if an item has been viewed and added-to-cart, we associate with it a *negative purchase*. A natural drawback of this approach is that it precludes the addition of negative samples having *view* as an interaction. To account for this, we include two negative pairs $(\ell_1, \text{view}), (\ell_2, \text{view})$, for each item ℓ that was only viewed. The items ℓ_1, ℓ_2 are drawn uniformly at random from the set of items belonging to the same taxonomy as ℓ in order to capture the user's preference in the item viewed in said taxonomy.

4.2 Implementation Details

In order to support custom negative sampling, we used the *fastText* library [2] developed by Facebook's AI research lab, to train our embedding models with extended functionalities built on top of the existing library. The primary innovation from *fastText* is that it enriches the word2vec model [13] with subword information by adding a bag of character ngrams. We chose the framework in favor of its easiness in extensibility as well as its efficiency in training. We experimented with tunning several hyperparameters of the model and eventually chose to set the context window to $m = 5$ and the embedding dimension to $d = 100$ (cf. section 3.1). In addition to the aforementioned custom negative samples, we also added five random negative samples in each of our sequences.

After training, we use the approximate K-nearest neighbor search algorithm, *Hierarchical Navigable Small World* [12] from Faiss (HNSW) [8], in order to get the top k similar items for each learned item-interaction pair. To balance efficiency with accuracy, we set the following hyperparameters in HNSW (*efSearch* = 256, *efConstruction* = 128, *linksPerVector* = 64) to scale up the approximated neighborhood search over hundreds of millions item-interaction pairs from our embedding training.

4.3 Offline Experiments

We test the performance of our model on data collected from shopping sessions that start in the 24 hour window following the last day of training.

4.3.1 Evaluation Methodology. We evaluate our candidate set selection method by computing a *hit rate*, a metric similar to recall for the candidate sets we generate. (We use hit rate as a metric instead of the the more commonly-used AUC because the candidates we generate are not necessarily present in historical data.) To do so, we first fix a target interaction i, and an item-interaction pair p and define the set

$$\mathcal{H}_p^i := \left\{ (\ell, q) \;\middle|\; p \neq (\ell, i) \in S,\; q = \sum_{S \in \mathcal{S}} q_S(\ell, i) \right\},$$

where $q_S(\ell, i) \in \{1, 2, \dots\}$ denotes the number of times item ℓ had interaction i in session S. For example, if $p = (\ell, \text{view})$ and i= purchase, then $\mathcal{H}_p^i$ contains all items (with associated quantities) purchased across all sessions after item ℓ was viewed. The hit rate of the p^{th} item-interaction pair is given by

$$h_p^i := \sum_{j=1}^{|\mathcal{H}_p^i|} q_j \mathbf{1}_{\{\ell_j \in C_p\}} \Bigg/ \sum_{j=1}^{|\mathcal{H}_p^i|} q_j,$$

where, C_p denotes the candidate set of the p^{th} pair produced by our model and $\mathbf{1}$ is the indicator function. The average hit rate across all item-interaction pairs for a target interaction i is given by

$$h^i := \frac{1}{k} \sum_{p \in S} h_p^i, \tag{3}$$

where k denotes the number of disctinct item-interaction pairs in the test set.

4.3.2 Experimental Results. For offline experiments, we choose two main models for candidate set generation for comparison: **View-Interaction**, which generates a candidate set based only on the "view-based" embeddings for a listing, and **Cart-Interaction**, based only on "cart-based" embeddings for a listing. We compare these methods with the following two baselines:

- **Co-Purchase:** This method represents each item as a *co-purchase vector*, a sparse vector indicating other items that have been co-purchased with it, similar to [10]. The candidate set is generated by finding the top K items with the most similar co-purchase vectors as the target item. We chose this as a baseline because it is the dominant candidate set selection method currently used in Etsy's production systems.
- **No-Interaction:** This method learns a single embedding for each item without differentiating between interaction types, similar to [6]. The candidate set is also generated by finding the top K items with the most similar embedding to the target item.

In Figure 3, we plot the average hit rate defined in (3) for all candidate set methods as a function of the number of candidates in the set number of items in them when the target interaction is a purchase. We observe that the candidate set associated with the view-interaction model outperforms all other candidate sets, including the co-purchase baseline when the number of items it contains

Model	Test Target Interaction		
	View (%)	Cart (%)	Purchase (%)
View-interaction	201.90	198.11	207.10
Cart-interaction	-6.88	34.88	47.74
Co-purchase	76.89	179.70	129.60

Table 1: Percent change in hit rate compared to the model with no-interactions for embedding vectors of dimension $d = 100$ and candidate sets of size 200.

Model	Coverage Rate (% of active listings)	
	Items (%)	Traffic (%)
View-interaction	73.98	85.34
Cart-interaction	70.57	80.13
No-interaction	78.51	96.11
Co-purchase	9.43	42.83

Table 2: Target item coverage rate, based on distinct active items and percent of traffic on one-day visit logs.

is sufficiently large. Although the view-interaction model underperforms the co-purchase model for candidate sets with smaller number of candidates, it is beneficial to explore additional items in candidate selection as this can increase the average hit rate. Additionally, we observe in Figure 3 that the no-interaction model has the smallest average hit rate among all other models. Therefore, learning an embedding for item-interaction pairs instead of a single embedding for each item has resulted in a candidate set which contained more items that were eventually purchased.

In Table 1, we observe that the view-interaction model outperforms the no-interaction model as well as all other models for all target interactions we consider. The cart-interaction model is shown to outperform the no-interaction model only when the test target interaction is either a cart or purchase action. However, it is not shown to outperform the co-purchase baseline. We believe that this is due to sparsity of add-to-cart interactions in our training data, resulting in suboptimal embedding representations.

One disadvantage of relying on historical co-interactions (i.e., co-purchases) for candidate set selection is their relatively low coverage rate. This is evident in Table 2, which shows that the existing *co-purchase* baseline covers only 9.43% of active items. This is due to stringent requirements for constructing such candidate sets. For example, the co-purchase candidate set requires two or more items to be purchased within a small time window. Such critieria is hardly applicable for the majority of items at Etsy due to the one-of-a-kind nature for the majority of the items. In addition, low coverage is also evident in the size of the candidate set for the co-purchase based candidate set (approximately 40 items). As shown in Table 2, both the item-interaction and no-interaction

Model	Embedding Dimension		
	$d = 25$ (%)	$d = 50$ (%)	$d = 75$ (%)
View-interaction	-7.51	-1.31	-1.85
No-interaction	-12.95	-0.19	-0.16

Table 3: Percent change in hit rate for item-interaction embedding models of different dimension compared to the model using $d = 100$. The test interaction used is purchase and the candidate set size is equal to 200.

methods cover at least 70% of distinct active items, which account for more than 80% of Etsy's traffic.

Table 3 shows performance metrics for the item-interaction and no-interaction models as a function of the embedding dimension. The numbers reported are average hit-rates for candidate sets with 200 items. For each embedding dimension, $d \in \{25, 50, 75\}$, the table shows its percent change in hit rate compared to its corresponding model type with $d = 100$. In both model types, we observe a drastic deterioration in performance for smaller dimension, $d = 25$. With Etsy's many one-of-a-kind listings, $d = 25$ is likely not large enough to capture the variance. As a result, we see the hit rate for both models improve as d increases. However, this marginal gain begins to saturate as $d > 100$ as computation feasibility begins to decrease model performance. Offline, we observe an almost linear trend in training time and machine memory in dimension d.

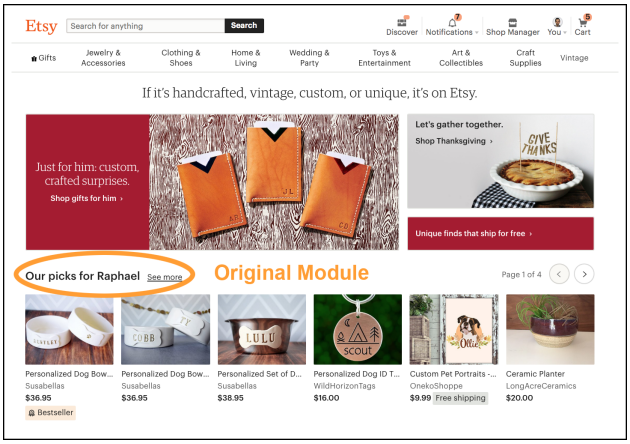
4.4 Online Experiments

To further evaluate our candidate set selection methodology, we implement and deploy the algorithm to a production user recommendation module on the homepage of Etsy.com. We train an embedding model based on Section 4.1.1 and generate candidate sets for each user as described below.

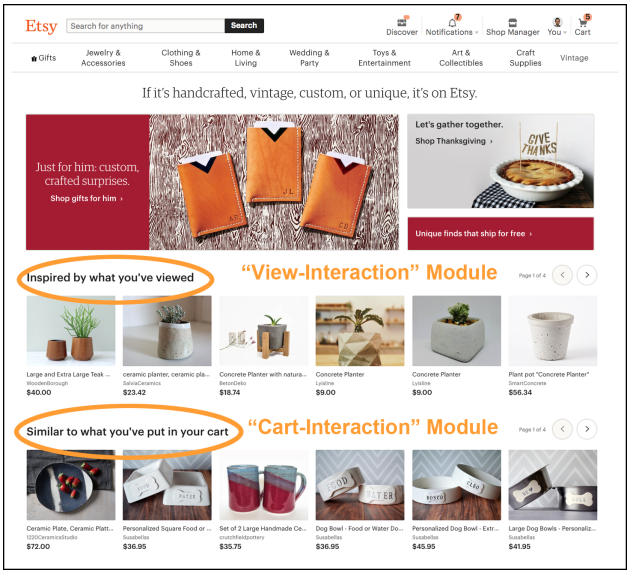
A live A/B test was run for 7 days in which we bucketed 50% of signed-in users into the *control* group, and the remainder into the *treatment* group. The *control group* received a single module that recommends items using a candidate set that matched items based on the last 100 items that the user interacted with, regardless of the interaction type. The *treatment group* received two separate modules: The first module finds the closest item-interaction embeddings based on the last 4 items that the user had viewed; the second module is computed similarly but is based on the last 4 items that the user has added to cart. Figure 4 shows a visual description of what the two variants look like.

Both the control and treatment groups used the same re-ranking model on top of the candidate set selection phase, which has an objective function that is optimized for predicting purchases, and uses a variety of historical user and listing rates and content features. We also note that while we prepared additional modules based on favoriting and purchasing behavior, they were not deployed to the live A/B test due to their lower-than-desired offline metrics.

The online A/B test evaluates an end-to-end system, which includes a combination of several factors, not only the choice of the candidate set selection algorithm. These additional factors include:



(a) Control Group



(b) Treatment Group

Figure 4: Visual examples of the recommendations received by users in the two buckets of our live A/B test, including: (a) the control group, which used a co-purchase based candidate set, and (b) the treatment group, which used the proposed item-interaction embedding model. The single recommendation module was split into two separate modules based on the way the user interacted with past listings. This was preferred as it offers more explainability to the user.

the design for breaking up the original module into two separate modules, the change of copy on the new modules, as well as the effect of the re-ranker on top of the candidate set. As such, we look at high-level business metrics to evaluate an end-to-end system with potentially many confounding factors.

For this particular experiment, the metrics that we tracked include site-wide click-through-rate, conversion rate, and add-to-cart rate. Given the preliminary experimental data, we observed promising upward trend in many key metrics, although more data is needed to gain statistical significance. Compared to the *control*, the first module in the treatment ("Inspired by what you've viewed") showed 4.1% improvement in click-through-rate. Additionally, the "treatment" group showed 0.20% and 0.31% increase in conversion rate and checkout rate, respectively.

5 CONCLUSION

In this paper, we describe a model for learning interaction-based item embeddings, which we use for candidate set selection. The proposed method allows us to encode co-occurance patterns between items and the way users interact with them. We train our model on a large production dataset of browsing sessions from Etsy and evaluate its performance both through offline metrics and online experiments. We observe that the canidate sets produced by our model improve upon the current production baselines.

REFERENCES

[1] Nima Asadi and Jimmy Lin. 2013. Effectiveness/efficiency tradeoffs for candidate generation in multi-stage retrieval architectures. In *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*. ACM, 997–1000.

[2] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching Word Vectors with Subword Information. *Transactions of the Association for Computational Linguistics* 5 (2017), 135–146.

[3] Fedor Borisyuk, Krishnaram Kenthapadi, David Stein, and Bo Zhao. 2016. CaSMoS: A framework for learning candidate selection models over structured queries and documents. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 441–450.

[4] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems*. ACM, 191–198.

[5] Chantat Eksombatchai, Pranav Jindal, Jerry Zitao Liu, Yuchen Liu, Rahul Sharma, Charles Sugnet, Mark Ulrich, and Jure Leskovec. 2018. Pixie: A system for recommending 3+ billion items to 200+ million users in real-time. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 1775–1784.

[6] Mihajlo Grbovic and Haibin Cheng. 2018. Real-time Personalization Using Embeddings for Search Ranking at Airbnb. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD '18)*. ACM, New York, NY, USA, 311–320. https://doi.org/10.1145/3219819.3219885

[7] Mihajlo Grbovic, Vladan Radosavljevic, Nemanja Djuric, Narayan Bhamidipati, Jaikit Savla, Varun Bhagwan, and Doug Sharp. 2016. E-commerce in Your Inbox: Product Recommendations at Scale. *CoRR* abs/1606.07154 (2016). arXiv:1606.07154 http://arxiv.org/abs/1606.07154

[8] Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2017. Billion-scale similarity search with GPUs. *arXiv preprint arXiv:1702.08734* (2017).

[9] Krishnaram Kenthapadi, Benjamin Le, and Ganesh Venkataraman. 2017. Personalized Job Recommendation System at LinkedIn: Practical Challenges and Lessons Learned. In *Proceedings of the Eleventh ACM Conference on Recommender Systems (RecSys '17)*. ACM, New York, NY, USA, 346–347. https://doi.org/10.1145/3109859.3109921

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

[11] David C Liu, Stephanie Rogers, Raymond Shiau, Dmitry Kislyuk, Kevin C Ma, Zhigang Zhong, Jenny Liu, and Yushi Jing. 2017. Related pins at pinterest: The evolution of a real-world recommender system. In *Proceedings of the 26th International Conference on World Wide Web Companion*. International World Wide Web Conferences Steering Committee, 583–592.

[12] Yu A. Malkov and D. A. Yashunin. 2016. Efficient and robust approximate nearest neighbor search using Hierarchical Navigable Small World graphs. *arXiv preprint arXiv:1603.09320* (2016).

[13] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*. 3111–3119.

[14] Dongjing Wang, Shuiguang Deng, Xin Zhang, and Guandong Xu. 2016. Learning Music Embedding with Metadata for Context Aware Recommendation. In *Proceedings of the 2016 ACM on International Conference on Multimedia Retrieval (ICMR '16)*. ACM, New York, NY, USA, 249–253. https://doi.org/10.1145/2911996.2912045