Purchase Prediction by Learning Representations From Interactions and Attributes With Heterogeneous Graph

Yujia Ding
Computational Science, Mathematics, and Engineering
y5ding@ucsd.edu

Jiayun Zhang
Department of Computer Science and Engineering
jiz069@ucsd.edu

Heng Zhang
Department of Electrical and Computer Engineering
hez009@ucsd.edu

ABSTRACT

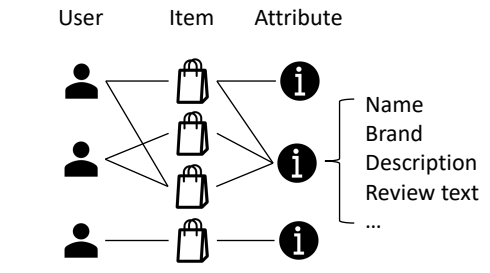
Purchase prediction is an important task in e-commerce. Successfully predicting whether a user would buy an item could give references for online stores to make decisions on what to recommend to the users in order to increase the sales and improve user experience. Many similarity-based, clustering and classification algorithms have been proposed to solve this task. However, one main challenge faced by many algorithms is that if a new item has never shown up in the dataset, it is hard to predict whether the users would buy it. In this project, we propose a new method, called RecHG, for purchase prediction task. We construct a heterogeneous graph based on the purchasing records (user-item interactions) and the attributes of the items extracted from their metadata and review texts. We apply node embedding techniques to learn the representations for users and items. Then the prediction of the purchasing behavior is based on the similarity of the representations. RecHG is capable of predicting the purchase of new items. We compare RecHG with several baseline models. According to our experimental results, RecHG can predict the purchasing behavior with an accuracy of 0.7461, which greatly outperforms the baseline models.

1 INTRODUCTION

Predicting the purchasing behavior of the users is of great importance for the operation and management of online shopping platforms. It provides useful analysis for a wide range of applications such as product recommendation and user behavior modeling. The online shopping platforms can utilize massive information of shopping records to predict users' purchasing behaviors. Traditional recommendation algorithms like collaborative filtering [17] can be applied to deal with this task. In addition, many efforts have been spent on designing new algorithms to do the purchase prediction including supervised learning with feature engineering [15] and neural network-based framework [9, 12].

With the emerge of new items, challenges arise as it is hard to predict a new item that has never appeared in the dataset before. Methods such as collaborative filtering fail to address this issue since no user has ever bought the newly appeared item. To tackle this challenge, in this project, we propose a new method, called RecHG, to predict if a user would buy an item by learning representations for users and items from the user-item interactions and item attributes with heterogeneous graph. An example of the heterogeneous purchasing graph is shown in Figure 1. There are three types of nodes (users, items and attributes which describe the items). The interactions between users and items (whether the user purchased the item) are captured in the graph with the user - item edges. Besides, the attribute nodes such as brand and phrases extracted from the description text are connected with associated item nodes, so that the semantic meaning of the attributes can be embedded in the items' representations, which helps the learning of new items. Moreover, since the metadata of the items are sometimes incomplete or missing, we further use the phrases in review text as complementary information. Phrases in review can provide sentiment information (whether the user like the item or not), as well as the additional attributes that may not be stated by the metadata (such as the style of the music).

We apply node embedding algorithms metapath2vec [7] to learn the representation for users and items. Then we calculate the interaction of the representations of the user and the item by multiplying them and input the result to the Logistic Regression model to make predictions.



(a) Purchasing Graph

(b) Meta Relations

Figure 1: The purchasing graph and its meta relations.

We conduct experiments to compare the performance of RecHG with several baseline models, including popularity-based prediction, similarity-based prediction, SVD++ and Item2Vec. According to our experimental results, RecHG can outperform these baseline models by almost 10% of the accuracy. We also compare the performance of RecHG and RecHG without the item - attribute edges. It shows that adding the attribute nodes can greatly improve the performance of the model.

We conclude our contributions as follows:

- We propose a graph-based purchase prediction system RecHG by learning representations for users and items and do purchase prediction. We construct a heterogeneous graph to

learning representations for users and items from the purchasing records and the item attributes extracted from metadata and review text. We integrate techniques from different domains, including phrase mining, node embedding, recommendation, into our system.
- We choose digital music as the target group of items for case study. We apply RecHG to predict whether a user would buy a piece of digital music.
- We conduct experiments to compare our method with several baseline models. According to the result, our model can outperform the baseline models. We also find that by adding item content extracted from its metadata and reviews, the model improves over 9% of the accuracy.

2 RELATED WORK ON PURCHASE PREDICTION AND RECOMMENDATION

Online purchase prediction and item recommendation is an important task for the e-commerce platforms and online retail platform[16][9] to promote their profits[25]. The e-commerce platforms apply recommendation algorithms to recommend a list of products which may interested by customers[22]. There are three common methods for the recommendation problem: CF (collaborative filtering), cluster models, and search-based method[16]. In the matrix factorization methods, such as SVD[14], the user-item matrix is decomposed and the users and items are mapped in two corresponding high-dimension latent factor space. To do the best prediction, the vector components is multiplied by the inverse frequency to make less frequent products more relevant[6]. In cluster models, users are divided into several groups[16]. The main idea of this algorithm is to find the most similar users, and then create clusters for users who have similar behavior. The prediction will be generated from the users' purchased products and ratings in that cluster[5]. In search-based approaches, prediction is generated by searching the related products[2]. By learning about products already purchased, the recommendation is given based on the similarities of the products, such as brand, company. In addition to the three methods mentioned above, graph based recommendation systems are used in these years[21]. The graph based recommendation system can use a variety of auxiliary information to overcome some disadvantages like data sparsity[21].

3 PROBLEM DEFINITION

3.1 Dataset

We use the Amazon Reviews data (2018)[18] for our projects. Amazon Reviews data contains the ratings, reviews and metadata of the products (descriptions, price, brand). The whole dataset contains the data of items from 29 categories. We choose the Digital Music category and use the small subset of the dataset (5-cores and ratings-only data).

3.1.1 Rating Data. There are 1,584,082 rating records in the Digital Music category, with 840,372 unique reviewers and 456,992 unique products. We use the following information in our study:

- *reviewerID*: Each reviewer has a unique ID
- *asin*: Each item has a unique asin ID
- *overall*: Rating of the corresponding item given by the corresponding user
- *unixReviewTime*: Time of the rating (unix time)

3.1.2 Metadata. Metadata is the descriptive information about the items. Table 1 shows an example. The metadata file contains the information of 74,347 items. We use the following three information from the items' metadata:

- *title*: Name of the item
- *brand*: Brand name of item
- *description*: Description of the item

3.1.3 Review Data. we use the 5-cores review data where the users and items have 5 reviews each. There are 169,781 review records in the 5-cores review data. We use the following information:

- *asin*: The asin ID of the item
- *reviewText*: Name of the item
- *format*: Format of the item (e.g. "MP3 Music", "Audio CD")

3.2 Exploratory Analysis

The distribution of the ratings is shown in Figure 2. The average of rating scores is 4.66. The 5-score ratings account for 80.81% of the records, which indicates that users tend to give rating scores to the items they bought.

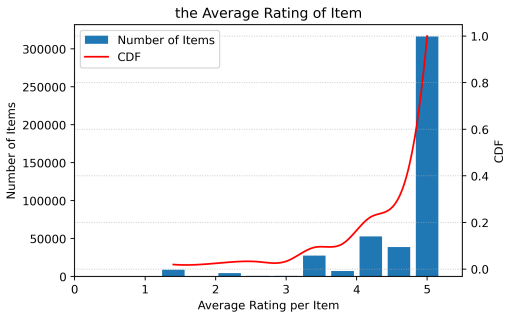


Figure 2: Rating distribution

In practice, we can only use the existing data to train the recommendation system and predict the future purchasing behavior. To simulate the real scenario, we split the dataset for training and testing according to time of rating. For each user, we sort the items he/she purchased according to the rating time. Then, we choose the first two thirds of the data for training, and the last third for validation and testing. We filtered out those users with less than 15 ratings. Finally, our sampled rating dataset contains 216,168 rating records with 6,571 unique users and 109,093 unique items.

3.3 Purchase Prediction Tasks

Given the purchasing records in training set, the metadata and reviews of the items, and the user-item pair with the ground truth labels (0 indicates user hasn't bought the item, while 1 indicates the user has bought the item) in the validation set. The goal is to predict the labels of user-item pairs in the testing set. Note that

Table 1: An Example of the Metadata and Review of the Item (asin: B001T188FS)

Property	Example	Extracted Attributes
title	Love Story	"love story"
Brand	Taylor Swift	"taylor swift"
Format	Audio CD	"audio cd"
Description	UK three track CD pressing of the second single lifted from her album Fearless...	"uk", "three track"...
Review	It was cool to be able to hear the Digital Dog Radio mix...	"radio mix"...

we generate negative samples in the validation and testing set, we will describe the process in Section 5.1. Denote the user set as U, item set as I, the metadata of the items as M, the reivews as R, and the connections between them as E. we learn the node embedding function g with $\{U, I, Meta, Review, E\}$ and get the representations for the nodes $R = \{r_{node_1}, r_{node_2}, ..., r_{node_n}\}$. Then we will use the representation of r_u and r_i to predict whether the user u has bought the item i.

$$\hat{y} = f(r_u, r_i), \quad \hat{y} \in 0, 1 \tag{1}$$

where f is the function to calculate the interaction of r_u and r_i.

4 PROPOSED METHOD

4.1 Overview

We propose a method to construct a heterogeneous graph to represent the purchasing behaviors of the users and learn the representations of the items and users with node embedding techniques into one feature space. The similarity of the items and users can then be calculated with the representations and used for recommendation.

The workflow of our method is shown as Figure 3. First, we do data preprocessing to extract the descriptive attributes from the metadata and reviews of the item. Then we build a hetergeneous graph based on the purchasing records and the music attributes. After that, we apply node embedding algorithm to learn the representations for each item and user. Finally, we do recommendation tasks with the learned representations.

4.2 Data Preprocessing

We extracted the attributes of the item from its metadata and the reviews from the users. Table 1 shows an example of an item and the attributes we extract. Specifically, for each item, we extract its brand (the brand of the digital music is e.g. "taylor swift") and format (e.g. "audio cd", "mp3 music"). In addition, since phrases are more capable of capturing contextual information comparing to words, we use AutoPhrase [23] to extract the phrases from the text of its name, description and reviews given by the users. In the example in Table 1, we get "uk", "three track", "radio mix", etc. from its description and review text. Note that all the attributes are in lowercase and we use Porter Stemming Algorithm [20] to reduce the word in attributes to their word stems.

4.3 Building Heterogeneous Graph

We use the purchasing records in the training set to form the heterogeneous graph, which we denote as *purchasing graph*. The network schema is shown as Fig. 1. There are three types of nodes in the purchasing network: users, items, and the attributes that describe the items. If user u ever bought item i, there will be an edge between u and i. The items are also connected with their attributes.

4.4 Learning User and Item Representation

The idea of node representation learning is similar to that of word embedding algorithms. In the word embedding scenario, words that appears nearby each other should have relations in their semantic meanings. Based on the assumption, a sliding window is used to extract the co-occurrence relationship of the words from the sentences. Then it uses the target word to predict its context words (or vice versa) to learn the representations of the words. In node representation scenario, the nodes that have co-occurrence relationship in the graph (be reachable within a certain distance) should be similar in their representations. In order to capture the co-occurrence relationship, Random Walk is applied to generate node sequences from the graph. Specifically, given the number of walks, Random Walk starts from a node and randomly selects its connected node as the next node, and repeats the process until it reaches the upper of walks set by the user to generate a node sequence. After generating the node sequences, skip-gram with negative sampling (SGNS) [13] is used to learn the node representations by using the target node to predict its context nodes within a sliding window on the node sequence. Several node representation learning algorithms [8, 19, 24] are proposed based on the idea.

In order to learn node representations in the heterogeneous graph, we use metapath2vec [7], which consider the types of nodes when generating node sequences. metapath2vec requires an input of metapath schema, which constrains that the node type when selecting the next node. In our task, we define the metapath as user - item - attributes - item - user. The sampling process starts from (1) selecting a user node u_1, (2) find an item i_1 connected with u_1, (3) select a attributes node c_1 connected with i_1, (4) select another item i_2 connected with c_1, (5) find another user u_2 connected with i_2. To generate one node sequence, we repeat the process in (1) - (5) until reaching the number of walks. After the node sequences are generated, we use SGNS to learn the node representation. The objective is to minimize the loss function:

$$L = -\sum_{t \in V} \sum_{m \in C(t)} logP(m|t) \tag{2}$$

where C(t) represents the set of attributes nodes of node t. The probabilities are approximated with negative sampling:

$$logP(m|t) = log(r_m^T u_t) - log \sum_{i \in V} exp(r_i^T u_t) \tag{3}$$

where u and r are the representation vector of node t and its attributes nodes.

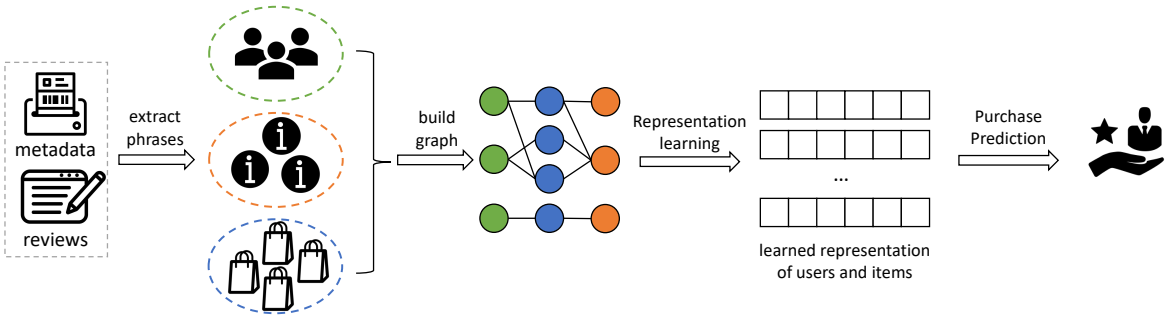


Figure 3: The Overview of Our Method.

4.5 Recommendation

After getting the learned representation for users and items, we can then use the representation to do prediction and recommendation task. We use the representation to perform purchase prediction task. We calculate the product of the representation of user and item, and use it as the input feature to Logistic Regression model.

5 EXPERIMENTAL EVALUATION

5.1 Experimental Settings

The dataset we use is the Amazon Review Data described in Section 3.1 which only consists of the positive samples. For the purchase prediction task, we also need negative samples of user-item pairs which user u has not purchased item i. For each entry user-item pair in validation and testing set, we sample a negative entry by randomly selecting an item that has not been purchased by the user. Finally, we get the training set with 141,758 positive user-item pairs. The validation set and testing set both have 37,205 negative samples and 37,205 positive samples.

5.2 Evaluation Metrics

For purchase prediction task, we use Accuracy to evaluate the performance of the models.

$$Accuracy = \frac{\text{Correctly predicted pairs}}{\text{Total number of testing pairs}} \tag{4}$$

5.3 Baseline Models

We consider the following four baseline models to compare with our model RecHG:

- Popularity-based prediction (popularity): The popularity-based prediction make predictions solely based on the popularity of the items.[11] We compute the percentage of users who bought each item. If the popularity of an item exceeds a threshold θ, it will predict that the item was purchased by the user. θ can be tuned with the validation set.
- Similarity-based prediction (similarity): Given a user-item pair (u, i), the similarity-based prediction make predictions based on the average of the Jaccard coefficients between the item i and the set of items i[1]that u has ever bought. This method does not perform good when "data sparsity" or "cold-start" problem occurs.[26]
- SVD++[4]: SVD++ is a type of latent factor model, which factorizing the matrix of user-item pairs to n dimension of the latent factor space. Each user is represented as a n-dimensional vector p_u and each item is represented as a n-dimensional vector q_i. The interaction of the user and item is calculated with the following equation:

$$r_{ui} = \mu + b_i + b_u + q_i^T (p_u + |N(u)|^{-\frac{1}{2}} \sum_{j \in N(u)} y_j) \tag{5}$$

- Item2Vec [3]: The idea of Item2Vec is derived from Word2Vec in Natural Language Processing. It consider the items that the user bought as the contexts and use SGNS to predict the contexts item with the target item. After getting the representations for each item, we apply similarity-based prediction with the representations. We calculate the average of the cosine similarity between the representations of the items that the user u bought and target item i and find the threshold as described in the similarity-based prediction method.

5.4 Implementation Details

- Popularity: We first calculate the popularity of each product within the training set. Then, we sort out a group of most popular items by tuning a threshold θ with the validation set. We try different value for the θ by iterating from the range (0.1 0.9) and observing the accuracy of prediction in the validation set. Finally we set the θ as 0.5.

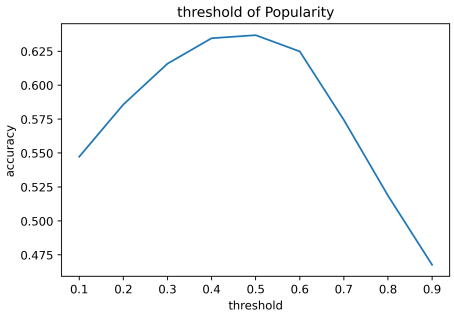


Figure 4: The thresholds of popularity

- Similarity: First, we collect all training items i' that user u has bought and all training users u' who have bought given item i for every unique user and item in the train set. Then for each given pair (u, i) we computed the average Jaccard similarity between i and i', which refers to users (in the training set) who have bought i and users who have bought i'. We predict that the user will buy items when the similarity exceeds a threshold. We chose the threshold from [0.1, 0.9] and the best threshold is 0.01.

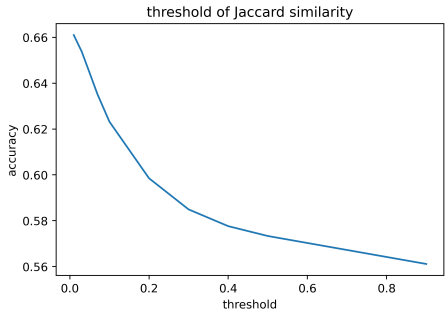


Figure 5: The thresholds of similarity

- SVD++: In order to implement SVD model, we used the surprise[10] library. We do grid search to find the best hyper-parameters for SVD++ algorithm. The best hyper parameters: n-factors=25, lr-all=0.01, reg-all=0.6, n-epochs=20. Then we used the latent factors learnt by the SVD++ model to predict the probability of whether the . We use the validation set to find a threshold for model to predict 0 or 1. The threshold is 0.61. Finally, using test dataset to evaluate the performance of this model.

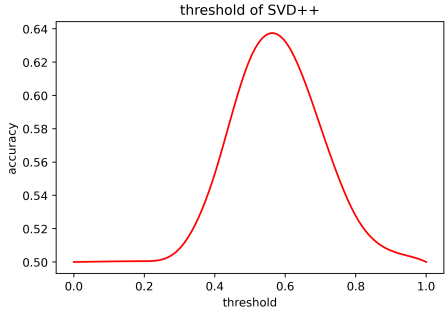


Figure 6: The thresholds of SVD++

- Item2Vec: To create a vocabulary of items, we first find the equivalent of "words" and "sentences": every unique item is treated as a "word" and all items that receive similar ratings from users are in the same "sentence".We classify all items with ratings below 4 as a "disliked" list and others as a “liked” list. We use Word2Vec from gensim library. 1. we set the maximum iteration as 50, the size of embedding as 64, negative sampling as 5. 2. Recommendation model: (1) Combing similarity between item-to-item and threshold to make prediction. We calculate the average cosine similarity of item-to-item. However, the accuracy of this model is only 0.5208. (2) Combing logistic regression and Item2Vec. We extract vectors of items appearing in train set and make prediction based on logistic regression. This model can highly improve the accuracy to 0.685. 3. Optimize parameter: (1) Find the best threshold between [0.1,0.99], and the highest accuracy is 0.5208. (2) Modify size to 64,128. The size of hidden layers makes little difference on accuracy. (3) Modify iteration to 50,100. The 50 epoch is better.

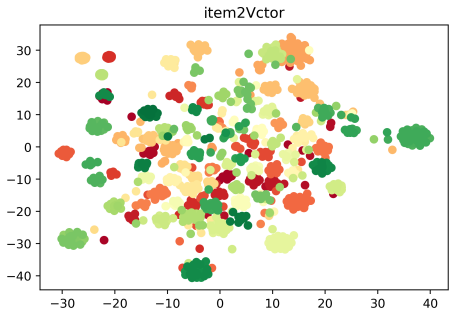


Figure 7: Item2Vec

- RecHG (our model): We use pytorch and pytorch-geometric library to build the metapath2vec model. The embedding size is set to be 64. The maximum iteration is set to be 150. The number of walks is 40. For each node, we generate 100 walk sequences.

5.5 Result

The performance in test set of all six models are shown in Table 2. Note that the fifth model (RecHG w/o attributes) is RecHG without the item - attribute edges. Popularity model achieves an accuracy of 63.95%, which meets our expectations because popularity is intuitively important for the purchase of digital music category. The popularity model is the simplest model, but the gap between popularity model with four other complex models is very small, which shows that this data set is dominated by popularity. Our model has the best performance in all those models with the accuracy of 74.61%. It is 9% higher than RecHG w/o attributes which shows the metadata and review information contribute a lot to the performance.

6 DISCUSSION

6.1 Advantages of our method

There are several advantages of our method compared to the baseline models.

- First, RecHG is capable for making predictions for new items. With the introduction of attribute nodes into the heterogeneous purchasing graph, the new items are connected with the previously existed item nodes through the common attributes and the representations of the new items can be learned.

- Second, RecHG make use of the rich text information from the metadata and the reviews which greatly improves the performance from purely relying on user-item interactions.
- Third, the graph-based structure of RecHG makes it robust to data sparsity problem. It can handle with the missing or incompleteness of the data.

6.2 Limitations and Potential Solutions

One limitation of RecHG is that it does not consider the temporal information of the user-item interactions. If item A was usually bought shortly after item B, item A and B should have stronger connections than other normal item pairs. One potential solution is that we can assign edge weight according to the time interval and co-occurrence frequency when building the purchasing graph, then generate node sequences regarding to the edge weight. Then the sampled node sequences will contain more samples from those frequently connected nodes and the learned representations will reflect the temporal information.

6.3 Future Work

There are several future directions about this project. First, as discussed above, we can take the frequency of co-occurrence of the items into consideration to improve the performance of the model. Second, we can extend the model to recommend items for new users who has not appeared in the dataset by adding the demographic attributes of the users into the heterogeneous graph. Although these new users have no purchase records, their representations can be learned with their displayed demographics. Third, the RecHG system can be further applied to the recommendation tasks such as recommending top-k items for a user. With the representations of the users and the items, we can calculate the cosine similarity of the user and the items and return the top k items with the largest similarity.

REFERENCES

[1] Hyung Jun Ahn. [n.d.]. Utilizing popularity characteristics for product recommendation. *International Journal of Electronic Commerce* 11, 2 ([n. d.]).

[2] Marko Balabanović and Yoav Shoham. 1997. Fab: content-based, collaborative recommendation. *Commun. ACM* 40, 3 (1997), 66–72.

[3] Oren Barkan and Noam Koenigstein. 2016. Item2vec: neural item embedding for collaborative filtering. In *2016 IEEE 26th International Workshop on Machine Learning for Signal Processing (MLSP)*. IEEE, 1–6.

[4] Robert M Bell, Yehuda Koren, and Chris Volinsky. 2008. The bellkor 2008 solution to the netflix prize. *Statistics Research Department at AT&T Research* 1 (2008).

[5] PS Bradley, U Fayyad, and C Reina. 1998. Scaling clustering algorithms to large databases. Knowl Discov Data Min.

Table 2: Purchase Prediction

Method	Accuracy
Popularity	0.6395
Similarity	0.6610
SVD++	0.6303
Item2Vec	0.6852
RecHG w/o attributes	0.6507
RecHG	**0.7461**

[6] JS Breese, David Heckerman, and Carl Kadie. 1908. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*. 43.

[7] Yuxiao Dong, Nitesh V Chawla, and Ananthram Swami. 2017. metapath2vec: Scalable representation learning for heterogeneous networks. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*. 135–144.

[8] 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*. 855–864.

[9] Chao Huang, Xian Wu, Xuchao Zhang, Chuxu Zhang, Jiashu Zhao, Dawei Yin, and Nitesh V Chawla. 2019. Online purchase prediction via multi-scale modeling of behavior dynamics. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2613–2622.

[10] Nicolas Hug. 2020. Surprise: A Python library for recommender systems. *Journal of Open Source Software* 5, 52 (2020), 2174. https://doi.org/10.21105/joss.02174

[11] Nirmal Jonnalagedda, Susan Gauch, Kevin Labille, and Sultan Alfarhood. 2016. Incorporating popularity in a personalized news recommender system. *PeerJ Computer Science* 2 (2016), e63.

[12] Mandy Korpusik, Shigeyuki Sakaki, Francine Chen, and Yan-Ying Chen. 2016. Recurrent Neural Networks for Customer Purchase Prediction on Twitter. *CBRecSys@ RecSys* 1673 (2016), 47–50.

[13] Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *International conference on machine learning*. 1188–1196.

[14] Omer Levy and Yoav Goldberg. 2014. Neural word embedding as implicit matrix factorization. In *Advances in neural information processing systems*. 2177–2185.

[15] Dancheng Li, Guangming Zhao, Zhi Wang, Wenjia Ma, and Ying Liu. 2015. A method of purchase prediction based on user behavior log. In *2015 IEEE International Conference on Data Mining Workshop (ICDMW)*. IEEE, 1031–1039.

[16] G. Linden, B. Smith, and J. York. 2003. Amazon.com recommendations: item-to-item collaborative filtering. *IEEE Internet Computing* 7, 1 (2003), 76–80. https://doi.org/10.1109/MIC.2003.1167344

[17] Andreas Mild and Thomas Reutterer. 2003. An improved collaborative filtering approach for predicting cross-category purchases based on binary market basket data. *Journal of Retailing and Consumer Services* 10, 3 (2003), 123–133.

[18] Jianmo Ni, Jiacheng Li, and Julian McAuley. 2019. Justifying Recommendations using Distantly-Labeled Reviews and Fine-Grained Aspects. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Association for Computational Linguistics, Hong Kong, China, 188–197. https://doi.org/10.18653/v1/D19-1018

[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*. 701–710.

[20] Martin F Porter et al. 1980. An algorithm for suffix stripping. *Program* 14, 3 (1980), 130–137.

[21] Chuan Qin, Hengshu Zhu, Fuzhen Zhuang, Qingyu Guo, Qi Zhang, Le Zhang, Chao Wang, Enhong Chen, and Hui Xiong. 2020. A survey on knowledge graph-based recommender systems. *Scientia Sinica Informationis* 50, 7 (2020), 937–956.

[22] J. Ben Schafer, Joseph A. Konstan, and John Riedl. 2001. E-commerce recommendation applications. *Data Mining and Knowledge Discovery* 5, 1-2 (1 Jan. 2001), 115–153. https://doi.org/10.1007/978-1-4615-1627-9_6

[23] Jingbo Shang, Jialu Liu, Meng Jiang, Xiang Ren, Clare R Voss, and Jiawei Han. 2018. Automated phrase mining from massive text corpora. *IEEE Transactions on Knowledge and Data Engineering* 30, 10 (2018), 1825–1837.

[24] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. Line: Large-scale information network embedding. In *Proceedings of the 24th international conference on world wide web*. 1067–1077.

[25] Liang Wu, Diane Hu, Liangjie Hong, and Huan Liu. 2018. Turning clicks into purchases: Revenue optimization for product search in e-commerce. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. 365–374.

[26] Jinbo Zhang, Zhiqing Lin, Bo Xiao, and Chuang Zhang. 2009. An optimized item-based collaborative filtering recommendation algorithm. In *2009 IEEE International Conference on Network Infrastructure and Digital Content*. IEEE, 414–418.