Distributed Representation-based Recommender Systems in E-commerce

Van-Thuy Phi† Liu Chen‡ and Yu Hirate‡

† Nara Institute of Science and Technology 8916-5 Takayama, Ikoma, Nara, 630-0192 Japan

‡ Rakuten Institute of Technology 1-14-1 Setagaya-ku, Tokyo, 158-0094 Japan

E-mail: † phi.thuy.ph8@is.naist.jp, ‡ {chen.liu, yu.hirate}@rakuten.com

Abstract Recommender system plays an important role in many e-commerce services, such as in Rakuten. In this paper, we focus on the *item-to-item* recommender and the *user-to-item* recommenders, which are two most widely used functions in online services for presenting relevant items given an item, or a particular user. We use a large amount of log data from one of Rakuten markets, and apply distributed representation method to that data for developing two types of recommender systems. The key idea of our approach is treating items as words, and users' sessions as sentences, then training the Word2vec model and Doc2vec models based on those items and user's information. Resulting item vectors from the Word2vec model can be used to calculate the cosine similarity between items, and find the similar items given an item. Similarly, Doc2vec model helps users find relevant items that might interest them using similarity between items and vectors. We also use the item vectors from both embedding models to build an additional user-to-item recommender, namely *Item Vector-based* system. The experiments show that our best system achieved a hit-rate of 24.17% for recommending items to users in testing data, which outperformed conventional approaches to a significant extent.

Keyword Recommender System, Distributed Representation, Item Vector-based

1. Introduction

Recommender system plays an important role in many e-commerce services, such as Rakuten[1]. Its goal is to offer relevant items given an item, or a particular user. If the system suggests similar items given an item, it provides *item-to-item* recommendation. In *user-to-item* case, this system helps users to find relevant items that might interest them. We focus on these two kinds of recommender systems, because they are commonly used in real-time services.

Conventional approaches for building recommender systems are divided into three classes: collaborative filtering methods, content-based methods, and hybrid methods. Collaborative filtering (CF) methods help users/customers to make choices based on the opinions of other people who share similar interests [2, 7, 8, 9]. They are considered to be the most popular and widely implemented techniques in recommendation systems. Content-based systems recommend items to users by comparing each item's attributes with the user profile so that only items that have a high degree of similarity with the user profile will be recommended [4, 6]. Hybrid recommenders are systems that combine multiple recommendations techniques together.

Recently, Mikolov et al. (2013) have introduced the skip-gram text modeling architecture [5]. It has been shown to efficiently learn meaningful distributed representations of words or phrases (aka word embedding) from un-annotated text. In the distributed representation, similar words are projected into similar vectors. Vectors from Word2vec model conserve some of the semantic characteristics in operations regarding the semantic information that they capture.

Word2vec model is great for capturing meaning of words or phrases. However, it only learns word embeddings based on the words' context. Le and Mikolov (2014) presented a novel method for generating the distributed representations of sentences and documents (aka Doc2vec model, or sentence embeddings) [3]. Doc2vec is an extension of Word2vec that learns to capture not just individual words but entire sentence and paragraph. Traditionally, Word2vec and Doc2vec models are trained on textual corpus data but here we utilize it on the log data of users. We are therefore treating each item as a "*word*" and the "*sentences*" are the ordered actions (e.g., viewing history of users). Then the original Word2Vec method can be applied in the recommendation scenario.

Based on the distributed representation approach, we apply it to the users' behavior data. The details of our contribution are as follows:

- We train the Word2vec model to assign a vector for each item, and build the item-to-item recommender system.

[1] http://global.rakuten.com/corp/worldwide/

- We apply sentence-embedding technique (Doc2vec model) to the data, and build the user-to-item recommender system. Doc2vec model give us a vector for each item, or user.
- We use the item vectors from both embedding models to build an additional user-to-item recommender system, namely *Item Vector-based*. We then compare our results with conventional approaches, e.g., collaborative filtering methods.

2. Applying Distributed Representation Approach to Recommender Systems

Based on the idea of utilizing cosine distance to measure the similarity between items, or users and items, we apply distributed representation approach to build our recommender systems. We use the log data from one of Rakuten markets to generate the item sequences. Each item sequence is a user' session. Therefore, a user may have several sessions, at different dates and times.

In Word2vec, or Doc2vec model, a document is a sequence of words with their context. In order to build recommender systems, we treat users' sessions as documents, and items as words in those documents/users' sessions. Each user has a sequence of item views with his/her intention. See Figure 1 for our approach.

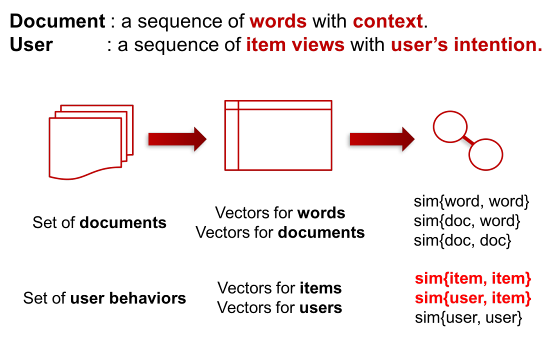

Figure 1: Distributed representation of users and items

3. Item-to-Item Recommender System

Given an item, the target of an item-to-item recommender system is to show relevant items to that item. In this section, we show the results of our item-to-item recommender system using the Word2vec model. We called it *Word2vec-based system*. First, we explain how we generate the data for training Word2vec model. Then, we use the trained model to find the similar items given a particular item.

3.1. Dataset

We collect user log data from one of Rakuten sub markets. It also offers a huge range of products from various merchants. The log data is collected from January till October 2015. It contains *click through* data, and *purchase history* data. Each record in the log data corresponds to a user's action, such as "*search*" or "view " or etc. The click through data is the user's behavior ranging from submitting a search query, to clicking on web pages of items, or scrolling on those web pages; while the purchase history is recorded in the transaction logs. By separating two kinds of data, we can analyze user' behavior effectively and further increase the performance of recommendation results.

Figure 2 shows some properties of the dataset such as length of sessions and number of sessions per user. The left chart is the distribution of sessions' length. In our dataset, more than 60% of users' sessions contain only one single request (user interacted with only one item). The chart on the right indicates the distribution of session count in the dataset. More than 50% of users visited the site once and did not come back in a long time. These important statistics help us understand the data, and give us some hints on how we should generate the users' sequences, and tune the parameters when training the embedding model.

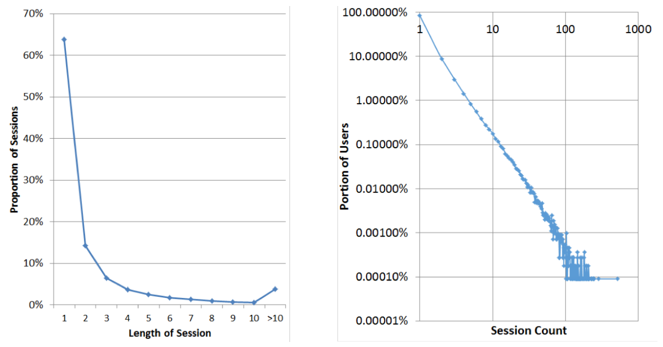

Figure 2: Dataset property

From the original data, we generate the item sequences, and use them to train the Word2vec model. The user's cookie from each record is utilized to identify the unique user. First, we group records with the same user's cookie. Each sequence is a user's session. It is a list of items that the user interacted with. Each session lasts less than 2 hours. It is called the *time interval* between users' sessions, which can reflect user's preference changes over a period of time. Therefore, a particular user may have multiple sessions.

Figure 3 illustrates our method for generating the users' sequences from user log data. We sort the items in each session by the timestamp that users performed their actions, and split users' items into sessions to generate the item sequences. In order to build the item-to-item recommender system, we use only the order of items in sessions, and ignore the user information.

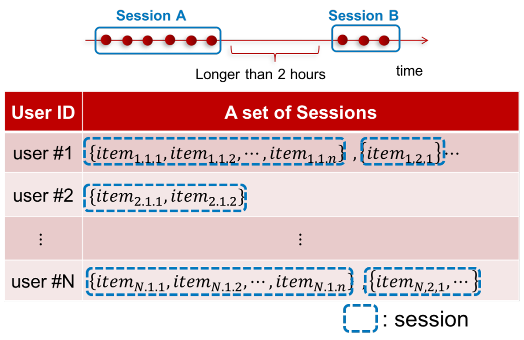
Session A
Session B
Longer than 2 hours
time
User ID
A set of Sessions
user #1
$\{item_{1.1.1}, item_{1.1.2}, \cdots, item_{1.1.n}\}$, $\{item_{1.2.1}\}\cdots$
user #2
$\{item_{2.1.1}, item_{2.1.2}\}$
⋮
⋮
user #N
$\{item_{N.1.1}, item_{N.1.2}, \cdots, item_{N.1.n}\}$ $\{item_{N.2.1}, \cdots\}$
: session

Figure 3: Users' sequences generated from user log data

3.2. Word2vec-based System

Functionally, the item-to-item recommender system takes an item as input, and outputs a set of similar items given the input. In order to give the item candidates, similar items are ranked from highest to lowest score, and top-n items with the highest scores will be stored as the output for the item-to-tem recommender system.

We train the Word2vec model using the item sequences in Section 3.1. Here, each item in sessions corresponds to a word in sentences. We keep only the meaningful sessions that have at least N items, and experimentally set N = 3. To avoid redundancy and reduce the training time, we also remove consecutive duplicate items in sessions, e.g., session "*itemA itemB itemB itemA itemC*" will be converted to "*itemA itemB itemA itemC*".

We use the skip-gram architecture to train the Word2vec model, and experimentally set the parameter value as follows:

Parameter	Values	Explanation
Size	300	Vector dimension
Window	8	Maximum number words/items of context
Negative	25	Number of "noise words/items" should be drawn (train faster)
Sample	1e-4	Sub-sampling of frequent words/items
Min-count	3	Items appear less than this min-count value is ignored
Iteration	20	Training iterations

Table 1: Parameter settings for training Word2vec model

Word2vec model gives us one vector for each item. To measure the similarity between two items, we calculate the cosine distance between their item vectors. Here, the bigger the cosine value is, the more similar the two items will be. Figure 4 shows an example of our Word2vec-based item-to-item recommender system's result for two kinds of data: *click through* data, and *purchase history* data. The first row is the search query, while the others are top-5 similar items to the query.

To evaluate the Word2vec-based item-to-item recommender system, we find top-15 similar items to all the items that have item vectors from Word2vec model, and visualize them by putting the items' images on a 2D plane. Figure 5 illustrates our visualization for some of the items.

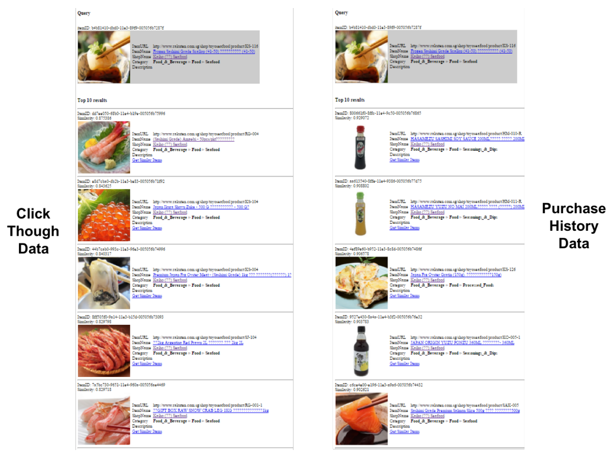

Figure 4: An example of our item-to-item recommender system's results

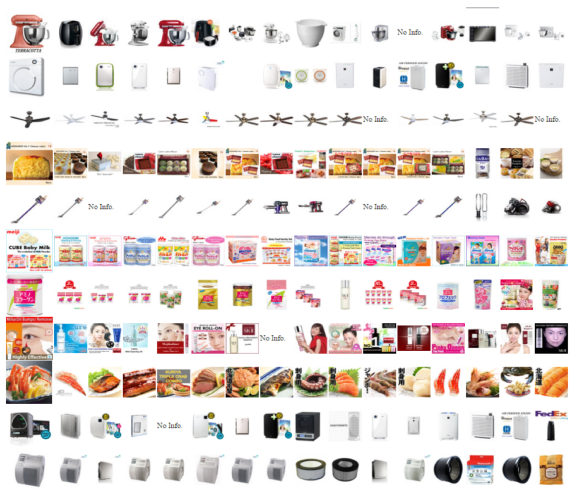

Figure 5: Visualization of our item-to-item recommender system's results, for some items

4. User-to-Item Recommender System

The goal of a user-to-item recommender system is to suggest relevant items to an individual user based on its knowledge of the user's behavior. That is, given a user, this recommender shows relevant items to the user. In this section, we present the results of two user-to-item recommender systems. The first system is based on the similarity between all user vectors and item vectors from Doc2vec model, whereas the latter system infers new item candidates from user's history items, and ranks their average similarities to the history items set. We also compare our results with conventional approaches, such as collaborative filtering methods.

4.1. Dataset

We split the log data into two separated sets: training and testing data. Training set contains the data from January to August 2015, while testing set is the data in September and October 2015. We generate the users' sequences from the training data. This is similar to Section 3.1. Then, we build the user-to-item recommender systems from training data, and evaluate our new systems on the testing set. Note that, we use both the order of items in sessions, and the user information (their cookies) for building the user-to-item recommender systems.

4.2. Evaluation Method

Our evaluation metric is similar to the one introduced by Deshpande and George (2004). The performance is measured by looking at the number of *hits* within the top-N items that were recommended by our user-to-item recommender system [1]. The number of hits is the number of items in the testing data that were also present in the top-N recommended items returned for each user. If n is the total number of common users in training and testing data, the *hit-rate* of the recommendation algorithm is computed as:

$$\text{hit-rate (HR)} = \frac{\text{Number of hits}}{n}$$

In all of experiments we set N = 20 as the number of items top be recommended by the recommender system. A hit rate of 1.0 indicates that the system was able to always recommend the relevant item, whereas 0.0 indicates that the system was not able to recommend any of the relevant items.

4.3. Doc2vec-based System

We develop the Doc2vec-based user-to-item recommender system from the training data in Section 4.1. Each item in sessions corresponds to a word, and each user's cookie corresponds to a sentence label. In contrast to the item-to-item recommender system, we keep all the items and consecutive duplicate items in sessions to increase the item quantity, and take into account the items that are viewed/purchased multiple times.

We also use the skip-gram architecture to train the Doc2vec model, and experimentally tune the parameter values as in Table 2. The combination of the following parameters is considered: *size*, *window, negative*, *sample*, *min-count*, and *iteration*. We then perform a grid search to find the optimal parameters and the best performance for our user-to-item recommender system.

Parameter	Values
Size	[50, 100, 200, **300**, 400, 500]
Window	[1, 3, 5, **8**, 10, 15]
Negative	[0, 5, **10**, 15, 20, 25]
Sample	[0, 1e-2, 1e-3, 1e-4, **1e-5**, 1e-6, 1e-7, 1e-8]
Min-count	[1, 2, **3**, ..., 20]
Iteration	[10,15, **20**, 25, 30]

Table 2: Parameter settings for Doc2vec model (Bold values are the best settings)

Our Doc2vec-based user-to-item recommender system achieved a hit-rate of 18.21% for 13,995 users with the best settings shown in the above table. After that, we tune each parameter to inspect the best setting for each of them, and find the important parameters that affect recommender's performance. The experiment results are analyzed in Figure 6.

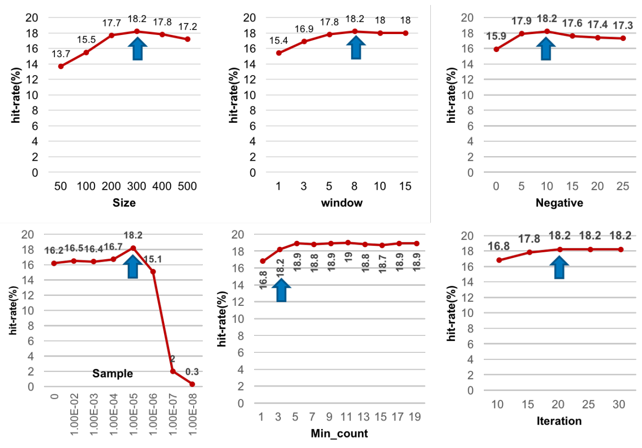

Figure 6: Parameter optimization for Doc2vec model

All the best settings for parameters result in the best performance (hit-rate), except *min-count*. If min-count value is increased, e.g., min-count=5, the hit-rate will be higher. However, the item quantity will be significantly reduced. We therefore, set min-count value to 3 for keeping the balance between the number of items and the system's performance. Figure 6 shows the impact of parameters Doc2vec model. Interestingly, *min-count* had little effect on our recommender system's hit-rate, while *sample* and *size* are two most important parameters that control the performance.

We compare our results with conventional approaches, including item-similarity based similarity [10] method and matrix factorization [11]. The item similarity based model first computes the similarity between items using the observations of users who have interacted with both items. Given a similarity between item *i* and *j*, it scores an item *j* for user *u* using a weighted average of the user's previous observations. There are server choices of similarity function to use, e.g., 'Jaccard', 'cosine' or 'pearson'. In our case, we choose 'Jaccard' to compute the item similarities. Matrix factorization method is another well-developed method in recommendation scenario. This model tries to learn latent factors for each user and item and then uses them to make recommendations.

The first three columns in Figure 8 show the hit-rate of two collaborative filtering based systems, and our Doc2vec based system. It indicates that our Doc2vec-based user-to-item recommender system performed significantly better than those two collaborative filtering based recommender systems, with a hit-rate of 18.21% compared to 3.42%, and 8.02%. Our new system has proved to be effective by significantly improving the hit-rate score.

4.4. Item Vector-based System

We also use the item vectors from Word2vec/Doc2vec model to build an additional user-to-item recommender, namely *Item Vector-based* system. Figure 7 provides an example of our item vector-based system. We develop a system based on the assumption that each user has a list of items that he/she interacted with in the training data, called *history items*, and each item is represented as an n-dimension vector from Word2vec/Doc2vec model. Then, our recommender system will find the similar items given history items calculate and rank their scores to keep the most confident items for a particular user. The details of our approach are described in the following:

- Each user has history items in the log data. They are used to infer item candidates for recommendation, which are similar items given those history items. We get item vectors from Word2vec/Doc2vec model, choose top-n similar items, and add them to the candidates set. In our case, we set n to 20.
- Next, we calculate the score for each pair of user and item, denoted by *SCORE(U, i)*, where j is an item in the candidates set. SCORE(U, i) is computed as:

$$\text{SCORE(U, i)} = \frac{\sum_{j\in U_{items}} \boldsymbol{Sim(i,\ j)}}{|\boldsymbol{U_{items}}|},$$

  where *i* is an item candidate, j is an item in history items set, and $|U_{items}|$ is the number of history items. This score is the average cosine similarity between each item in the candidates set and history items.
- Finally, we rank items' scores, and keep top-N similar items that have highest scores. We set N to 20 and use these items as the recommended items for that user. For example, in figure 7, only items in the red eclipse will be recommended for the user.

We leverage the item vectors from pre-trained Word2vec model in Section 3.2 for calculating items' scores. Testing data and test users are the same with the data used to evaluate our Doc2vec user-to-item recommender system; therefore we can compare their results in terms of hit-rate score.

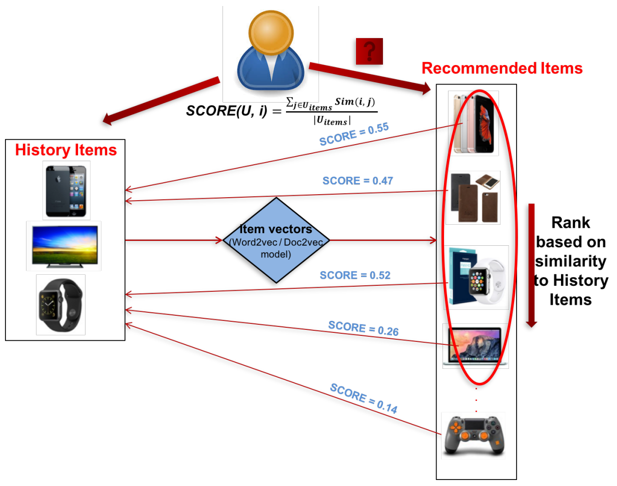
?
Recommended Items
SCORE(U, i) = $\frac{\sum_{j\in U_{items}} Sim(i,j)}{|U_{items}|}$
SCORE = 0.55
History Items
SCORE = 0.47
Item vectors (Word2vec / Doc2vec model)
SCORE = 0.52
SCORE = 0.26
SCORE = 0.14
Rank based on similarity to History Items

Figure 7: Illustration of item vector-based system

Figure 8 shows the performance of various user-to-item recommender systems. Our new item vector-based system achieved of 21.04% when recommending for 13,907 / 13,995 (99.4% user coverage), with the Word2vec's item vectors.

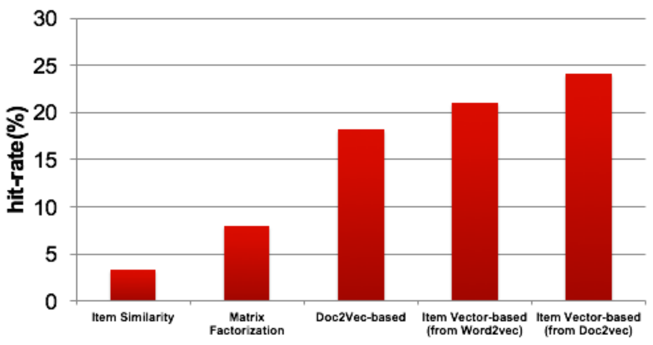

Figure 8: Performance of user-to-item recommender systems

We also use the item vectors from Doc2vec model instead of Word2vec model. The experiments showed that our proposed systems achieved a hit-rate of 24.17% for recommending items to 13,981 / 13,995 users (99.9% user coverage) in testing data, and outperformed conventional approaches.

5. Conclusion and Future Work

In this paper, we focused on item-to-item and user-to-item recommender systems. We developed the distributed representation-based recommender systems, and applied that approach to the dataset from one of Rakuten markets. Experiment results illustrated that our proposed systems achieved a hit-rate of 24.17% for recommending items to users in testing data, and outperformed collaborative filtering methods.

In future work, we plan to evaluate the distributed representation-based recommender systems based on other datasets, such as from Rakuten Ichiba, one of the largest e-commerce site in Japan.

References

[1] Deshpande, Mukund, and George Karypis. "Item-based top-n recommendation algorithms." ACM Transactions on Information Systems (TOIS) 22.1 (2004): 143-177.

[2] Konstan, Joseph A., et al. "GroupLens: applying collaborative filtering to Usenet news." Communications of the ACM 40.3 (1997): 77-87.

[3] Le, Quoc V., and Tomas Mikolov. "Distributed representations of sentences and documents." arXiv preprint arXiv:1405.4053 (2014).

[4] Lops, Pasquale, Marco De Gemmis, and Giovanni Semeraro. "Content-based recommender systems: State of the art and trends." Recommender systems handbook. Springer US, 2011. 73-105.

[5] Mikolov, Tomas, et al. "Distributed representations of words and phrases and their compositionality." Advances in neural information processing systems. 2013.

[6] Pazzani, Michael J., and Daniel Billsus. "Content-based recommendation systems." The adaptive web. Springer Berlin Heidelberg, 2007. 325-341.

[7] Resnick, Paul, et al. "GroupLens: an open architecture for collaborative filtering of netnews." Proceedings of the 1994 ACM conference on Computer supported cooperative work. ACM, 1994.

[8] Sarwar, Badrul, et al. "Item-based collaborative filtering recommendation algorithms." Proceedings of the 10th international conference on World Wide Web. ACM, 2001.

[9] Schafer, J. Ben, et al. "Collaborative filtering recommender systems." The adaptive web. Springer Berlin Heidelberg, 2007. 291-324.

[10] Francesco Ricci, Lior Rokach, and Bracha Shapira. Introduction to recommender systems handbook. Springer US, 2011.

[11] Koren, Yehuda, Robert Bell and Chris Volinsky. “Matrix Factorization Techniques for Recommender Systems.”Computer Volume: 42, Issue: 8 (2009): 30-37.