PKE: A Model for Recommender Systems in Online Service Platform

Yun-Chien Tseng
yctseng0304@cs.nctu.edu.tw
National Chiao Tung University
Hsinchiu, Taiwan

ABSTRACT

Graph embedding is a technique that has grown attention in recent years. Apart from mining implicit information in a graph representation data, graph embedding can be used in recommender system. Recommender system is a tool that can help e-sellers collect users' information more easily. This is beneficial for platform providers to mine users' interests with more data. However, it is easy to be distracted by other unrelated information when too many data are collected. In this paper, we proposed an idea called information matrix to combine information from different data. This idea considers data as graphics; hence, connects data with common nodes and extends to vectors through keyword embedding. In addition, we constructed a model that illustrates the efficiency and ability of the information matrix. This model was tested on data provided by e-sellers. Our aim was to combine browsing data and order data, and transfer these data into information matrix with high accuracy. Although the accuracy rate drops after transferring information into embedded type, it provides an idea for potential solution of cold start, a common problem in recommender systems.

CCS CONCEPTS

• **Information systems** → *Document structure*; • **Computer systems organization** → **Embedded systems**; *Redundancy*; Robotics; • **Networks** → Network reliability.

KEYWORDS

Recommendation, Graph embedding, Keywords analysis

ACM Reference Format:
Yun-Chien Tseng. 2020. PKE: A Model for Recommender Systems in Online Service Platform. In *Companion Proceedings of the Web Conference 2020 (WWW '20 Companion), April 20–24, 2020, Taipei, Taiwan.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3366424.3382090

1 INTRODUCTION

Recommender systems are widely used by e-commerce corporations. It has been applied to various genres of recommendation, such as recommending books (e.g. amazon)[2], music (e.g. iTunes)[16], video (e.g. youtube) [2] and a variety of products. An efficient recommender system will find out items that users have browsed and identify new items users might have interests in. To make a recommender system more efficient, we uncover the hidden information from the dataset by applying graph embedding techniques.

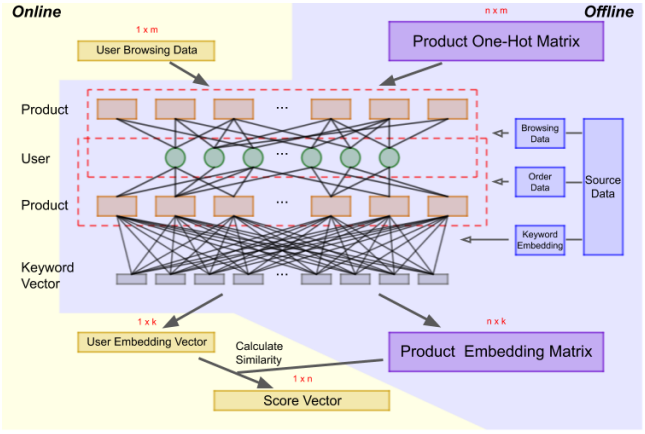

Figure 1: The full structure of our proposed model PKE.

Besides learning and observing on purchasing history log, online purchase platform contains a useful dataset: browsing data. We can learn information from all users' browsing data to find possible products that most users would accept; moreover, to mine a single user's data and guess his or her preference. However, some studies suggested that there are still room for improvement on "Machine Reasoning" and "Explainable Recommendation" when using graph embedding and machine learning on recommender system [17] [5]. This means that embedding method can help rise a recommender's hit rate, but sometimes without reasoning, or could not explain with straight forward reasons (needing some more social studies). For example, the famous diaper-and-beer phenomenon. Although it makes sense after some social study explanation, we can not expect all indirectly-reasonable recommendation have such relation.

In this study, we constructed a model called Probability-Keyword-Embedding (PKE in short). PKE is a model that operates data from "information matrix" by embedding method similar to prod2vec [6] [15]. The information matrix is a collection and combination of dataset we have that based on the idea of "Law of Large Numbers [9]", which believes that enough scale of data could conclude a trend of user preference. "Law of Large Numbers" is a fundamental idea in statistic, but rarely been used directly in such way in data mining or machine learning related paper. The major challenge in generating information matrix was to connect information from different dataset. In the following work, we start from connecting

WWW '20 Companion, April 20–24, 2020, Taipei, Taiwan

ACM ISBN 978-1-4503-7024-0/20/04.
https://doi.org/10.1145/3366424.3382090

two dataset: buying records and the browsing record for the same website.

In general, our contributions can be conclude as follows:

(1) Merged information from different (but related) dataset to one computation structure, which is called information matrix (IM) in this paper. With IM, we are able to simplify complicated information, handle and observe the data with simpler way.
(2) Proposed PKE model, which is able to recommend explainable-related items, that combines pure mathematics and machine learning idea. Case study example will be detailed in experiment section.
(3) Proposed an extended idea: labeling the keywords. By labeling keywords, we can feature a new product, give it a suitable vector that is constructed similar to old products. This can solve cold start problem in recommender systems.

2 MODEL

The PKE model is based on an information matrix constructed by browsing history and order data. We believe that it was possible to aggregate some patterns (or trends) of products by accumulating these data. We fixed a time period T for the model to learn information during this period, and use what it have learnt to predict the next time period T'.

2.1 Information Matrix

Information matrix is built by ratio between past collecting data from two dataset (browsing history and order data), therefore the recommendation will be more reasonable. Our goal for building an information matrix was to combine different data into simple computation node. Our model was tested on data given by a traveling product e-commerce platform. In convenience, we categorized the collected data into user's browsing and order data. This setting can be changed by given data.

The information matrix is noted as I^T, indicates the matrix for time T. I^T is defined as follow:

$$I^T{}_{ij} = \frac{\sum_{k=1}^{n}(P_j^o)_k + \alpha_j}{\sum_{k=1}^{m}(P_i^b)_k + d\alpha_j}, \alpha_j = \sum P_j^b/d \tag{1}$$

P_i^b represents the browsing log of i^{th} product, P_j^o represents the order log of j^{th} product, and α_j is a factor refering Laplace correction design to avoid dividing zero, d is selling days of the product. Newly on stock products and inevitable reasons, such as data collecting problem, etc. could cause its $\sum_{k=1}^{m}(P_i^b)_k$ (browsing record) zero, therefore, α_j is essential to adjust possible data error.

2.2 Offline

We noted that some products have been viewed but never been bought, so we defined the 'seen products' and 'bought products' separately. Suppose there are m seen (browsing) products and n bought (order) products and there are l users. Each user has a browsing log $u_i^b = (w_{1i}^b, w_{2i}^b, \cdots, w_{mi}^b)$ and order log $u_i^o = (w_{1i}^o, w_{2i}^o, \cdots, w_{ni}^o)$. Meanwhile, we can have product vector

$$\begin{aligned} P_t^o &= (\sum_{i=1}^{l} {w_{1i}}^b {w_{ti}}^o, \sum_{i=1}^{l} {w_{2i}}^b {w_{ti}}^o, \cdots, \sum_{i=1}^{l} {w_{mi}}^b {w_{ti}}^o) \\ &\text{for ordered product, and} \\ P_t^b &= (\sum_{i=1}^{l} {w_{ti}}^b {w_{1i}}^o, \sum_{i=1}^{l} {w_{ti}}^b {w_{2i}}^o, \cdots, \sum_{i=1}^{l} {w_{ti}}^b {w_{ni}}^o) \\ &\text{for browsed product.} \end{aligned} \tag{2}$$

In general, we noted $(P_t^o)_r = \sum_{i=1}^{l} {w_{si}}^b {w_{ti}}^o$ and $(P_t^b)_s = \sum_{i=1}^{l} {w_{ti}}^b {w_{1i}}^o$ where $r \in \{1, 2, \cdots, m\}.s \in \{1, 2, \cdots, n\}$. We used users u_i to represent the connection between browsing data and order data. This can be used in two sets of data that have the same column. In our case, customer is the common column, therefore, we can connect the browsing data and order data through customer (common column).

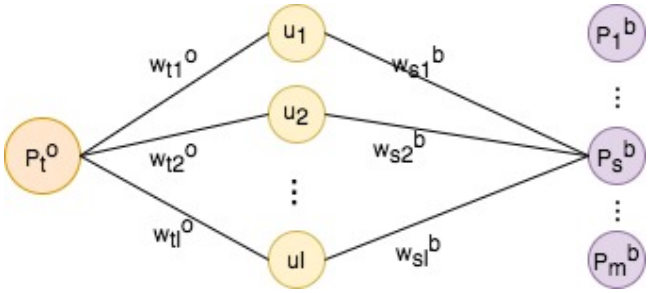

Figure 2: Construction of the s^{th} dimension for order vector P_t^o

In Fig. 2, the middle layer $(u_1, u_2, \cdots, u_l)$ represents the connection between two datasets. Any column that repeatedly appear in both datasets can be the connecting bridge. In our model, the goal is to mine information from products then recommend to users, therefore, our connecting bridge for information matrix are users and both side are products.

2.3 Keyword Embedding

Keywords were generated by items in the side layer. We selected the item name of browsing product and used Jieba [12], a Chinese word segmentation Python library, to generate keywords.

After generating keywords, we selected continuous k keywords as vertex to connect with the vertex that represents the browsing nodes. An input ordered product P_t^o is correspond to a list of browsing product $(P_1^b, P_2^b, \cdots, P_m^b)$ through user's records $(u_1, u_2, \cdots, u_l)$. Through the information matrix calculation, the user layer have been embedded, browsing products $(P_1^b, P_2^b, \cdots, P_m^b)$ are directly connected to keywords. The connection between browsing products P_x^b and keywords κ_y is defined by following equation 3:

$$e_{xy} = \begin{cases} 1 & \text{if } \kappa_y \text{ is in } P_x^b, \\ 0 & otherwise \end{cases} \tag{3}$$

Through the network heritage, vector of a given product P_t^o is defined as follow:

$$\hat{P_t^o} = (\sum_{i=1}^{m}(P_t^o)_i e_{i1}, \sum_{i=1}^{m}(P_t^o)_i e_{i2}, \cdots, \sum_{i=1}^{m}(P_t^o)_i e_{ik}) \quad (4)$$

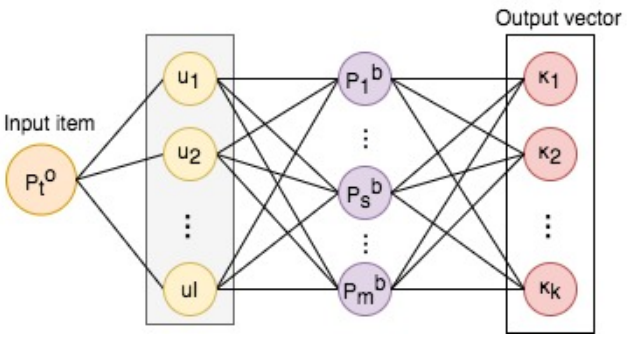

Figure 3: The input item and output vector $(\kappa_1, \cdots, \kappa_k)$

We could note $\kappa_s = \sum_{i=1}^{m}(P_t^o)_i e_{is}$. According to Fig.4, it is suggested that we should select the top k keywords instead of s to $s+k$. Since the value of s is to large, we would lose information for each order product if s were selected.

2.4 Online

The most complicated part of building the model is keyword embedding and information matrix calculation. Products that have been bought by users will outcome a vector P_t^o. When a user enters our system online, they will leave browsing record. If they bought products during their first browsing, we can mine their potential interests and recommend more related products. Moreover, even if users didn't buy anything after first browsing, we can still predict what they would be interested in. We generated user vector through user's browsing history as follow:

$$f(u) = \frac{\sum_{i=1}^{m} \hat{P_i^o} a_i}{\sum_{i=1}^{m} a_i} \quad (5)$$

u represents a user's browsing log, the $f(u)$ outcomes a vector represents the user. $a_i = 0$ if the user never browses i^{th} product in the past given period, otherwise a_i represents the time input user browses i^{th} product.

3 EXPERIMENT

Our data were provided by KKday, one of the leading e-commerce platform in Asia that provides traveling related product, such as amusement park ticket, one day tour, train tickets reservation, etc. Table 1 shows the statistic of data we used in the experiment.

Table 1: Statistic of our data

	2018 whole year	2019 January
member*	387,669	145,353
browsing	31,807,436	4,280,959
order	2,188,231	273,935

The '2018 whole year' information were use to generate information matrix (IM for short), which is our training data. '2019 January' is the testing data. Product vectors were generated by 2018 information, thus product vectors were limited by old products in 2018. The goal of our PKE model is to break this kind of limitation. We want to generate vectors for new products that have not been browsed or ordered in the past. Further works need to be done on keyword label extension.

3.1 Variable Setting

We were inspired by traveling products, which contains seasonal influence (eg, tourists can only go skiing in Korea during winter, around December and January, but not in March.). To include all seasons in a year, we defined our time period T as 1 year. Setting $T = 1$ year includes all season, also were more likely to cover full information of seasonal products. The information matrix I^T was constructed by 2018 whole year order, browsing, product and member data. Finally, the testing data were tested on 2019 January.

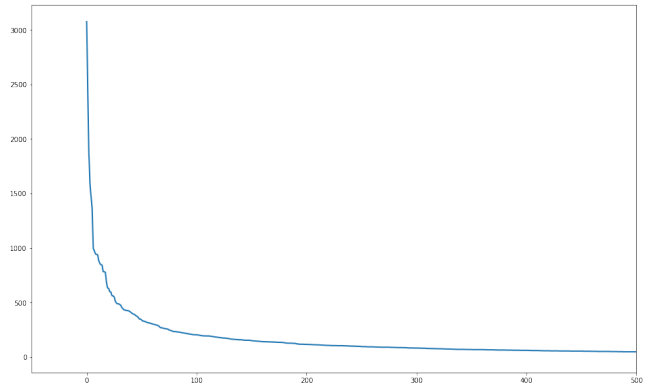

Figure 4: Keywords appearance ranking. The keywords were ranking in their appearance times between different product names.

In Fig. 4, we can see top keywords appear significantly higher rate in product names. The top frequent keywords appear in product names over 3000 times, and only top 27 words appears more than 500 times. We set to collect keywords appear in more than 10 products, all of those keywords (appear more than 10 times keywords) have mean appearance times 328 and std 363. Accordingly, we can conclude that keywords appearance has a very wide gap. High appear rate could lead to both positive and negative influence on embedding to product vector. For instance, selecting only highest rate keywords could let system only focus on the most popular products. Therefore we choose to runs result on the 10^{th} 210^{th} keywords. The selected 10^{th} keyword is keyword with the appearance time lower than $(average + std/2)^{th}$. We also selected $k = 200$ as the dimension to compare with LINE, which model default is embed data into 200 dimension.

3.2 Results

We tested our result with existing historical data, then planned to work online if the result meets the efficiency. Based on some case

Table 2: Results of January order data testing by our idea.

model	Precision@5	NDCG@5	Coverage@5
IM	0.856	0.755	2912/4640
IM+PKE	0.294	0.139	143/4640
IM+PKE+N	0.535	0.371	1019/4640
LINE	0.375	0.272	546/4640

studies, users started to browse their target products 15 days before ordering in average. In order to gather more information, we set to collect user's browsing log by 1 month before the user's first order log (in January and February, separately). Since we tested our data on users that have order records, it is hard to define recall. Therefore, our evaluation is precision, normalized discount cumulative gain (for short, NDCG), and coverage. Coverage is a method to evaluate the diversity of our recommend result since traditional recommender systems, such as content-based filtering [14] [1], collaborative filtering [7] [10], tend to recommend popular products and ignore new products, which leads to the cold start problem.

The recommend result was ranked by using cosine similarity to judge the similar level between product and user. Note that "IM+PKE+N" means normalizing Keyword vectors before evaluating. LINE[13] has normalized its vector when embedding. As can be seen by Table2, the best result is by pure implying IM. It is reasonable because IM is a buying-browsing ratio matrix, but there is no 'embedding' technique. We first try to embed user vector and product vector separately by LINE, but the result shows low efficiency (worse than random guess); therefore, the result is not listed in Table2. After experiencing failure in separately embedding, we generated the user vector by weighted averaging the products that have been browse.

3.3 Case Study

Our model PKE was originally planned to compare with LINE[13] BiNE[4] and metapath2vec[3], and yet the result was surprisingly unable to compare. We supposed the reason is that some essential information that BiNE and metapath2vec needed were not preserved when generating the IM, our first step of embedding. Although LINE and metapath2vec do not shows its benefits in recommender system, we done some case study to show the result.

We chose product id 7781 to do the case study, which is "Universal Studios Japan Ticket Express Pass[1]". This product was chosen because it is the most popular browsing and order product on KKday website.

For instance, in our case, we considered LINE more likely to learn information from our information matrix, therefore we chose to list only LINE comparing with PKE to make fairness.

However, applying pure IM makes an surprisingly result in high precision. As a result, we also made a case study for IM result. We chose a user randomly to see what our model will recommend to the user, in this case the user is A, . In Table4, the product in

[1]Products in case study are not list in the full name, but enough to recognise a specific products.

Table 3: Case study on product 7781: Top 3 similar product

model	1st	2nd	3rd
PKE	Osaka Amazing Pass (18418)	ICOCA Card	Universal Studio Express Pass
meta[3]	Golden Reel Ferris Wheel Ticket (5092)	Shoryudo Highway Bus Pass	Tokyo Disneyland E-Ticket
LINE[13]	Leofoo Village Theme Park Admission Ticket (6370)	Universal Express Pass: Special Edition	Nankai Rapi:t Airport Express Ticket

Table 4: Top 5 recommend product for user A

rank	product name
1	**Hong Kong Disneyland tickets**
2	Hong Kong Airport Express tickets
3	**Hong Kong Ngong Ping 360 Ticket**
4	Protable Wifi for Hong Kong, Macau and China
5	Hong Kong Ocean Park ticket

boldface was purchased by user A. According to this example, we can find explainable recommendation by our structure PKE.

4 FUTURE WORK

4.1 Keyword Label Extension

Keyword labeling is one of our focus future work. Further work on keyword labeling could be done by using the existing way of word segmentation in Chinese to categorize the collected keywords into p labels. Hence, we would be able to find the clique[8] between keywords to generate the vector for product. The illustrating blueprint is referred in Fig.5,

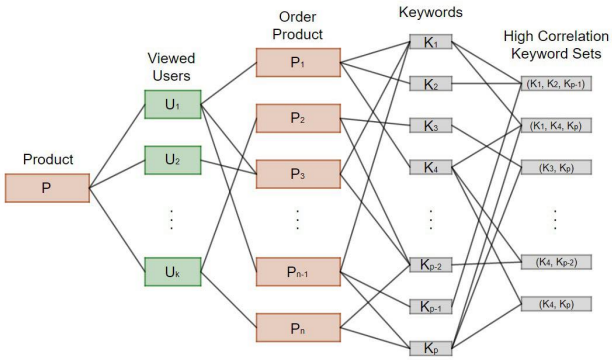

Figure 5: Keyword Label extension

5 CONCLUSION

In conclusion, we proposed an idea to concatenate information between dataset through the construction of an information matrix. A single original data generates a higher dimension of vectors

(left part of Fig.2). However, the dimension can be lower when connecting to other data. Further on, we extended a layer that generates vector by keywords. Our model PKE combined two parts and showed outstanding results during the offline test. In our future work, we want to strengthen the keyword part of the PKE model. This could be done by labelling keywords, then referring meta-path[11] or motif-clique [8] to generate vector for new products. By generating vector for new products, the long term problem in recommender system, cold start problem, could be solved.

ACKNOWLEDGMENTS

Special thanks to Professor Wen-Chih Peng's fully support in research and help. Thanks for reviewer's suggestion, and all of the others who help me in the experiment and paper writing.

REFERENCES

[1] Justin Basilico and Thomas Hofmann. 2004. Unifying collaborative and content-based filtering. In *Proceedings of the twenty-first international conference on Machine learning*. ACM, 9.

[2] Chih-Ming Chen, Ming-Feng Tsai, Yu-Ching Lin, and Yi-Hsuan Yang. 2016. Query-based music recommendations via preference embedding. In *Proceedings of the 10th ACM Conference on Recommender Systems*. ACM, 79–82.

[3] 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*. ACM, 135–144.

[4] Ming Gao, Leihui Chen, Xiangnan He, and Aoying Zhou. 2018. Bine: Bipartite network embedding. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. ACM, 715–724.

[5] Palash Goyal and Emilio Ferrara. 2018. Graph embedding techniques, applications, and performance: A survey. *Knowledge-Based Systems* 151 (2018), 78–94.

[6] 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. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 1809–1818.

[7] Jonathan L Herlocker, Joseph A Konstan, Al Borchers, and John Riedl. 1999. An algorithmic framework for performing collaborative filtering. In *22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 1999*. Association for Computing Machinery, Inc, 230–237.

[8] Jiafeng Hu, Reynold Cheng, Kevin Chen-Chuan Chang, Aravind Sankar, Yixiang Fang, and Brian YH Lam. 2019. Discovering Maximal Motif Cliques in Large Heterogeneous Information Networks. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*. IEEE, 746–757.

[9] Kenneth L Judd. 1985. The law of large numbers with a continuum of iid random variables. *Journal of Economic theory* 35, 1 (1985), 19–25.

[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] Changping Meng, Reynold Cheng, Silviu Maniu, Pierre Senellart, and Wangda Zhang. 2015. Discovering meta-paths in large heterogeneous information networks. In *Proceedings of the 24th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 754–764.

[12] J Sun. 2012. 'Jieba'Chinese word segmentation tool.

[13] 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*. International World Wide Web Conferences Steering Committee, 1067–1077.

[14] Robin Van Meteren and Maarten Van Someren. 2000. Using content-based filtering for recommendation. In *Proceedings of the Machine Learning in the New Information Age: MLnet/ECML2000 Workshop*. 47–56.

[15] Flavian Vasile, Elena Smirnova, and Alexis Conneau. 2016. Meta-prod2vec: Product embeddings using side-information for recommendation. In *Proceedings of the 10th ACM Conference on Recommender Systems*. ACM, 225–232.

[16] Jheng-Hong Yang, Chih-Ming Chen, Chuan-Ju Wang, and Ming-Feng Tsai. 2018. HOP-rec: high-order proximity for implicit recommendation. In *Proceedings of the 12th ACM Conference on Recommender Systems*. ACM, 140–144.

[17] Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. 2019. Deep learning based recommender system: A survey and new perspectives. *ACM Computing Surveys (CSUR)* 52, 1 (2019), 5.