Query-based Music Recommendations via Preference Embedding

Chih-Ming Chen[1,2], Ming-Feng Tsai[1], Yu-Ching Lin[3], Yi-Hsuan Yang[2]
[1]Department of Computer Science, National Chengchi University, Taipei, Taiwan
[2]Research Center for IT innovation, Academia Sinica, Taipei, Taiwan
[3]KKBOX Inc., Taipei, Taiwan
{104761501,mftsai}@nccu.edu.tw, aaronlin@kkbox.com, yang@citi.sinica.edu.tw

ABSTRACT

A common scenario considered in recommender systems is to predict a user's preferences on unseen items based on his/her preferences on observed items. A major limitation of this scenario is that a user might be interested in different things each time when using the system, but there is no way to allow the user to actively alter or adjust the recommended results. To address this issue, we propose the idea of "query-based recommendation" that allows a user to specify his/her search *intention* while exploring new items, thereby incorporating the concept of information retrieval into recommendation systems. Moreover, the idea is more desirable when the user intention can be expressed in different ways. Take music recommendation as an example: the proposed system allows a user to explore new song tracks by specifying either a track, an album, or an artist. To enable such heterogeneous queries in a recommender system, we present a novel technique called "Heterogeneous Preference Embedding" to encode user preference and query intention into low-dimensional vector spaces. Then, with simple search methods or similarity calculations, we can use the encoded representation of queries to generate recommendations. This method is fairly flexible and it is easy to add other types of information when available. Evaluations on three music listening datasets confirm the effectiveness of the proposed method over the state-of-the-art matrix factorization and network embedding methods.

Keywords

recommender systems; query-based recommendation; heterogeneous preference embedding

1. INTRODUCTION

Recommender systems typically generate a list of recommendations without taking any input from users, but relying on merely user preference histories. In this way, users can only *passively* receive recommended results without any chance to alter or adjust the results. However, in practice, users usually have multiple intentions and may like to *actively* input their interests to obtain different recommendations. To meet this demand, this paper proposes an idea called "query-based recommendation" and presents a novel approach called "heterogeneous preference embedding" to carry out the idea.

The proposed query-based recommender system can greatly increases the quality of recommendations. The rational is that, in addition to user preferences, the system can know more about the query intention, and then find out those items from the user preferences that might be relevant to the query. The idea of query-based recommendation is quite similar to that of context-aware recommendation, the goal of which is to incorporate the contextual information of a user to recommend specific items [1]. However, in the literature, most of existing context-aware recommender systems mainly focus on how to collect or utilize certain contextual information such as time [11] and location [3] into their recommendation models, and suggest some fine-tuned models for the recommendation problems under different circumstances, such as the point-of-interest recommendation.

For the query-based recommendation, it is highly desirable when the system can utilize heterogeneous relevant information for recommendation and allow a user to specify his/her query intention in different ways. Accordingly, we propose the technique of *Heterogeneous Preference Embedding* (HPE) to achieve this. In specific, the proposed method constructs a user-item preference network from user access logs, in which the vertices represent users and various items and the edges are the relationships between the users and the items. After the construction, an edge-sampling technique is applied to embed the large information network into low-dimensional vector spaces, in which the proximity information of each vertex is encoded into its learned vector representation [10]. Afterwards, we can use the learned representations of vertices with some simple search methods or similarity calculations to conduct the task of query-based recommendation. A major advantage of the proposed method is that it is flexible and it is easy to incorporate different information into the recommendation model when available.

In the experiments, three music listening datasets are employed to assess the performance of our proposed method, including the Last.fm-1K dataset, the Million Song Dataset (MSD), and one from a regional music online streaming service provider KKBOX Inc. The proposed method is compared against the state-of-the-art recommendation algorithms using either matrix factorization or network embedding techniques DeepWalk [7] or LINE [10]. The experimental results show that the proposed method outperforms the prior methods significantly in the three datasets.

RecSys '16, September 15-19, 2016, Boston , MA, USA
© 2016 ACM. ISBN 978-1-4503-4035-9/16/09…$15.00
DOI: http://dx.doi.org/10.1145/2959100.2959169

In summary, the contribution of this paper includes:

- The idea of query-based recommendation is proposed in this paper, which is a hybrid research work of information retrieval and general recommender systems.
- We formulate the query-based recommendation as a preference-embedding problem, and propose a method to construct the preference network between users and various, possibly heterogeneous, items.
- Extensive experiments have been conducted to verify the effectiveness of the proposed method.

2. QUERY-BASED RECOMMENDATIONS

In what follows, we briefly formalize the task of query-based recommendation. The task is to recommend a list of items I to a user u based on her previous preferences and an input query q. The recommendation task can be formulated as $f_\theta(u, I, q)$, which can also be divided into two parts:

$$f_\theta(u, I, q) := \underbrace{\alpha \times f_\theta(u, I)}_{\text{the user preference}} + \underbrace{(1-\alpha) \times f_\theta(I, q)}_{\text{the query intention}}, \quad (1)$$

where $\alpha \in [0, 1]$ is a parameter to weigh the results from the user preference model and the query intention model. In the literature, several models have been proposed to deal with the user preference [2, 9]. For simplicity, we focus on only the query intention part in this paper and set $\alpha = 0$. Our goal is to show that the proposed embedding approach on the user preference network can better incorporate information from a user's query intention into the learned latent representation than a general matrix factorization model does.

From the perspective of query-based recommendation, we attempt to regard various information as queries, so the flexibility of incorporating the arbitrary types of information into the recommendation model is a key point for our development.

3. HPE: HETEROGENEOUS PREFERENCE EMBEDDING

Network embedding technique is an approach to mapping information networks into low-dimensional spaces, in which each vertex is represented as a low-dimensional vector. Such a low-dimensional vector is useful in a variety of applications such as recommendation [10]. In the field of nature language processing, there is a similar technique called word embedding, the goal of which is also to learn the low-dimensional representation of each word. There have been some well-known studies of word embedding [4, 5, 6] in the field. In this work, we attempt to apply the information embedding techniques [7, 10] to deal with the recommendation problem, and propose a method called heterogeneous preference embedding for the task of query-based recommendation for arbitrary value of $\alpha \in [0, 1]$. In the followings, we describe the details of how to apply the embedding techniques for the task.

3.1 Construction of User Preference Network

In a recommendation dataset, there are usually user-to-other-entity pairs. The initial step is to convert these pairs into a user-to-other-entity bipartite graph. Take music recommendation dataset as the example. As shown in Figure 1,

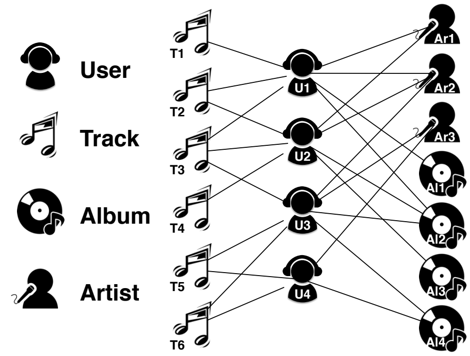

Figure 1: The user preference network. Suppose there are totally four kinds of data attributes in the music listening dataset. In the figure, the user U_4 may frequently listen to the tracks T_5 and T_6, the artist Ar_3 and the album Al_4.

every data entity (i.e. a user, a track, an artist or an album) is treated as an individual vertex in the graph, and they are connected to each other by the observed user preference (i.e. rating or listening times). This convention is applicable to most kinds of recommendation dataset. There are some useful properties for the modeling process:

- **User Preference Edges**: The weight of the edges is determined by the preference strength. These weights are user-specified and can be numerical or binary.
- **Heterogeneous Graph**: There is no limitation on the utilized entity type. The proposed model learns the representation of multiple types of entity in one network so that it is able to compute the similarity between any two entities.
- **Bipartite Graph**: It is a bipartite graph so that the vertices in the graph can be divided into two disjoint sets according to the connections. One of them contains purely users and the other set contains all the other entities. Modeling the direct connected pairs is equivalent to modeling the user preference. This form is suitable to adopt most existing collaborative filtering (CF)-based models as well.

3.2 Edge Sampling via Weighted Random Walks

An intuitive way to obtain the neighbor vertices of a vertex is to unfold all the second-order connections, but this greatly increases the memory usage especially for the dataset contained a large item pool. To resolve this, DeepWalk model uses random walk strategy [7] to generate training pairs and LINE model uses edge sampling [10] during the training stage. They are two recent network embedding techniques. From the optimization perspective, the general matrix factorization models consider only the user-to-item pair type, while this approach takes account of every entity-to-entity pair type.

As Figure 2 demonstrates, we choose to adopt the edge sampling to fast obtain the directly connected pairs and then to use a weighted random walk to receive the indirectly connected vertices. Instead of generating the walks uniformly, the weighted random walk generates the walks according to the weights of the observed edges. In our experiments, we consider the proximity information within 2 steps for each

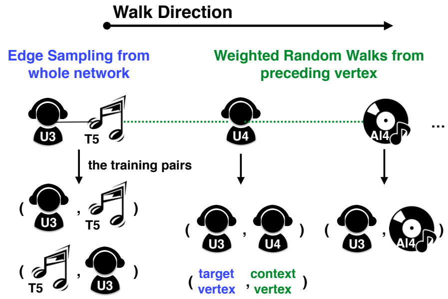

Figure 2: Edge sampling via weighted random walks for training pairs. In our sampling scheme, the first sampling pair is derived from the edge sampling over the whole network. Then, the following training vertices are selected from the weighted random walk on the preceding vertex.

vertex. Note that it is straightforward to enlarge the window size to model higher-order proximity.

3.3 Query Intention Modeling via HPE

To embed the user preference into the vertices v, the proposed HPE model updates the vertex representation Φ according to the sampling proximities derived from the preference network. In other words, we treat those indirect connected vertices as the contextual information of the center vertex, which can be represented as follows:

$$Pr(v_j|\Phi(v_i)) = \begin{cases} 1 & \text{if } v_j \in Context(v_i) \\ 0 & \text{otherwise} \end{cases}. \tag{2}$$

Maximizing the above posterior probability is equivalent to minimizing the negative log likelihood, so the objective function of the second-order proximity can be represented as follows:

$$O = -\sum_{(i,j)\in S} w_{i,j} \log p(v_j|\Phi(v_i)) + \lambda \sum_i \|\Phi(v_i)\|^2 , \tag{3}$$

where S is a set of sampling pairs and w indicates the weight of the edge. In practice, the observed edges are all positive information and thus falls into the one-class prediction problem. We adopt the widely-used solution called negative sampling to sample the additional user-to-other-entity pairs from the unobserved data. This works well in point-wise optimization functions [12].

In the heterogeneous preference embedding, a regularized term is adopted to avoid the over-fitting problem. That is due to the fact that we seek to preserve the inference ability that can match the vertices containing the similar contexts, rather than match those vertices containing exactly the same context. In addition, we also adopt the asynchronous stochastic gradient descent (ASGD) [8] algorithm to optimize Equation 3. The overall procedure is summarized in Algorithm 1.

4. EXPERIMENTS

Three music listening datasets are employed to assess the performance of our proposed method. The first one is the lastfm-dataset-1K dataset,[1] which contains the listening logs

[1] http://www.dtic.upf.edu/~ocelma/

Algorithm 1: Heterogeneous Preference Embedding

Input: User Preference Network: $G(V,E)$,
Walk Steps: w, Sampling Times: n

for $v \in V$ **do**
  Initialize the representation: $\Phi(v)$ and context v
for $i \in \{1,...,n\}$ **do**
  $(v_1, v_2) = EdgeSampling(G)$
  Update $\Phi(v_1)$, context v_2 by minimizing Eq. (3)
  Update $\Phi(v_2)$, context v_1 by minimizing Eq. (3)
  for $v' \in RandomWalk(v_2, w-1)$ **do**
    Update $\Phi(v_1)$, context v' by minimizing Eq. (3)

Output: Vertex representations Φ

Table 1: Experimental Datasets

Dataset	#Users	#Items	#Logs
lastfm-1k	992	107,528	19,150,868
MSD	1,019,318	384,546	48,373,586
KKBOX	50,000	400,000	220,000,000

from the website of Last.fm. The second one, released by EchoNest,[2] is a music taste profile subset derived from the official user dataset of the Million Song Dataset (MSD). The third one is a dataset with user listening logs provided by KKBOX Inc., which is a regional leading music streaming company. The third dataset covers user listening logs from 2014 to 2015. Table 1 lists the statistics of the three datasets.

For query-based recommendations, most users may only care about the top recommendations. Therefore, we adopt 1) precision at k (P@k) and 2) mean Average Precision at k (mAP@k) as the evaluation metrics, where k indicates the number of the cut-off recommended items. Calculating the precision with a small k is equivalent to examining the possible hit ratio of the users in top recommendations.

In the following experiments, we randomly select 70% of the listening history for each user as the training logs, and put the rest 30% logs in the test set for the off-line evaluation. In the testing stage, we randomly select 5 queries from the test logs of each user, and ask for matching the recommendations to these selected query items.

Note that the ground truth of the query-based recommendations shall depend on both the user and the user's query (even when we set $\alpha = 0$ in this work). Therefore the best returned results are varied from person to person. Treating the query as a context information, we assume that a user may tend to listen to similar songs in a short period, thereby considering the songs *co-listened* within the time period as the ground truth. In the lastfm-1k and KKBOX datasets, we consider the co-listen frequency that is higher than 3 as the ground truth. In MSD, we do not further filter the ground truth because it does not contain the primitive listening logs.

To verify the effectiveness of the proposed method, we compare it with other state-of-the-art approaches, including one simple method (i.e. popularity-based), one CF-based model (i.e. matrix factorization) and two embedding models (i.e. DeepWalk and LINE-2nd). The similarities among the entities are measured by cosine similarity. Below we briefly describe these baseline methods:

[2] http://labrosa.ee.columbia.edu/millionsong/tasteprofile

Table 2: Performance of query-based recommendations. The ground truth of the first two datasets are derived the co-listen frequency within a window size 5 for each user, whereas the MSD uses the original test logs as the ground truth; the * symbol is used when the result is significantly better (p-value<0.01 in a paired t-test) than the result of the others.

P@10									
	lastfm-1k (window=5)			**KKBOX** (window=5)			**MSD** (original)		
	$d = 16$	$d = 32$	$d = 64$	$d = 64$	$d = 128$	$d = 256$	$d = 64$	$d = 128$	$d = 256$
Popularity	2.66%	2.66%	2.66%	4.32%	4.32%	4.32%	0.92%	0.92%	0.92%
MF	3.02%	3.93%	4.22%	7.11%	8.49%	8.93%	1.37%	1.79%	2.00%
DeepWalk	3.18%	3.55%	3.54%	11.61%	12.55%	13.08%	1.71%	1.95%	1.95%
LINE-2nd	3.44%	3.74%	4.10%	12.79%	13.47%	12.77%	1.62%	1.60%	1.14%
HPE	**3.54%**	***4.22%**	**4.51%**	**12.95%**	***13.74%**	***14.20%**	***2.08%**	***2.15%**	***2.19%**
mAP@10									
	lastfm-1k (window=5)			**KKBOX** (window=5)			**MSD** (original)		
	$d = 16$	$d = 32$	$d = 64$	$d = 64$	$d = 128$	$d = 256$	$d = 64$	$d = 128$	$d = 256$
Popularity	**3.27%**	**3.27%**	**3.27%**	5.03%	5.03%	5.03%	1.04%	1.04%	1.04%
MF	1.87%	2.34%	2.60%	4.65%	5.85%	6.16%	1.88%	2.44%	2.81%
DeepWalk	1.82%	2.10%	1.99%	8.73%	9.47%	10.01%	2.66%	2.70%	2.55%
LINE-2nd	2.00%	2.10%	2.38%	9.95%	10.64%	10.09%	1.84%	1.60%	1.44%
HPE	2.08%	2.55%	2.71%	**10.14%**	**10.86%**	***11.31%**	**2.86%**	***3.09%**	***3.12%**

- **Popularity-based**: This is a baseline approach that always recommends the most popular songs to users. The popularity is estimated by the number of the users who favor/listen to the song in the training data.
- **Matrix Factorization** (MF): The model is a well-develop technique for making recommendations on a user-to-item data structure. Given the entities in the built preference newtwork can be seperated into two disjoint sets, we can utilize the benchmark from MyMediaLite [3] to generate the recommendations.
- **DeepWalk**[7]: DeepWalk uses local information obtained from truncated random walks to learn latent representations of nodes in a graph. It is an extended application of word2vec-based model.
- **LINE-2nd**[10]: We adopt the LINE second-order (2nd) version in order to make the comparison to our proposed context embedding model.

Table 2 lists the experimental results in terms of $Precision$@10 (upper table) and mAP@10 (bottom table). We can find that the three embedding models, including the proposed HPE model, achieve a comparable performance to the MF model for the three datasets, indicating that the embedding models can capable of capturing the query intentions form the preference network. Note that the recommendation performance drops for the DeepWalk and LINE methods degrades when the dimensions of the vectors become higher. In contrast, the proposed preference embedding with regularized terms can prevent the learned representation from over-fitting, yielding better recommendation performance in most cases.

5. CONCLUSION

In this paper, we propose the HPE model that is designed towards the application of "query-based recommendation." The HPE exploits not only the observed preference but also the possible relevant entities via the user preference network. Therefore, by embedding such proximity information, the learned representation can better reflect the query intention in comparison to the CF-based model. Moreover, for the proposed preference embedding, we adopt the regularized learning to alleviates the over-fitting issue while learning the embedded representations. Our experiments confirm the effectiveness of the proposed method in making recommendations for the task of query-based recommendation.

[3] http://www.mymedialite.net

6. REFERENCES

[1] G. Adomavicius and A. Tuzhilin. Context-aware recommender systems. In *Proc. ACM RecSys*, 2008.

[2] Y. Hu, Y. Koren, and C. Volinsky. Collaborative filtering for implicit feedback datasets. In *Proc. IEEE ICDM*, 2008.

[3] J. J. Levandoski, M. Sarwat, A. Eldawy, and M. F. Mokbel. Lars: A location-aware recommender system. In *Proc. IEEE ICDE*, 2012.

[4] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *ICLR Workshop*, 2013.

[5] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26*. 2013.

[6] J. Pennington, R. Socher, and C. D. Manning. Glove: Global vectors for word representation. In *Proc. EMNLP*, 2014.

[7] B. Perozzi, R. Al-Rfou, and S. Skiena. Deepwalk: Online learning of social representations. In *Proc. ACM SIGKDD*, 2014.

[8] B. Recht, C. Re, S. Wright, and F. Niu. Hogwild: A lock-free approach to parallelizing stochastic gradient descent. In *Advances in Neural Information Processing Systems 24*. 2011.

[9] S. Rendle. Factorization machines with libFM. *ACM Trans. Intell. Syst. Technol.*, 2012.

[10] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei. LINE: large-scale information network embedding. *CoRR*, 2015.

[11] Q. Yuan, G. Cong, Z. Ma, A. Sun, and N. M. Thalmann. Time-aware point-of-interest recommendation. In *Proc. ACM SIGIR*, 2013.

[12] W. Zhang, T. Chen, J. Wang, and Y. Yu. Optimizing top-n collaborative filtering via dynamic negative item sampling. In *Proc. ACM SIGIR*, 2013.