



## An Improved Recommender for Travel Itineraries

---

Yajie Gu, Jing Zhou and Shouxun Liu

EasyChair preprints are intended for rapid dissemination of research results and are integrated with the rest of EasyChair.

June 20, 2018

# An Improved Recommender for Travel Itineraries

Yajie Gu, Jing Zhou, and Shouxun Liu

School of Computer Science  
Communication University of China, Beijing, 100024, China  
gu\_yj@cuc.edu.cn

**Abstract.** Given the vast amount of information, including the numerous points-of-interest (POIs) and the various hotels, available on travel websites such as tripadvisor or booking, a recommender system would help users, who are planning their next trip, filter out unnecessary information based on their requirements. We improved our previous work on a recommendation system that was intended to facilitate the generation of daily travel itineraries. We used the X-Means clustering algorithm to divide all attraction sites and hotels into groups according to geographical location. Meanwhile, a Word2Vec model was trained using the Wikipedia text corpus to obtain similar tags of specific ones. A tag-based mapping algorithm was applied to create a list of candidate attractions that best match with the user's favorite spots. Finally, by taking into account the weather information, our recommender can further refine the list of candidate attractions and work out a daily itinerary that involves desirable hotels and attractions. The shortest itinerary (SI) and the itinerary with the highest performance/price ratio (MEI) will then be produced for user selection. The results of a series of experiments demonstrated that, compared to others, our personalized recommender for travel planning can provide a more appealing and detailed travel plan containing daily itineraries for users.

**Keywords:** daily itineraries, evaluation, improvements, tag similarity, X-Means clustering algorithm.

## 1 Introduction

It is time-consuming for users to pick up a desirable travel itinerary on various travel websites. So, a recommendation system is helpful for them to generate personalized results and to save their time. Personalized recommenders in tourism can be mainly divided into three categories: (1) providing popular itineraries for a user without taking into consideration the user's specific requirements (for instance, mafengwo.cn<sup>1</sup>); (2) indicating all POIs (Points of Interests, or attractions) that match with user's preference from which users have to choose and

<sup>1</sup> [https://www.mafengwo.cn/mdd/cityroute/10065\\_5934.html](https://www.mafengwo.cn/mdd/cityroute/10065_5934.html)

figure out a plan on their own (e.g. [elong.com](http://www.elong.com)<sup>2</sup>); (3) accepting information on the origins, destinations, and travel date from users and in return, offering the driving routes and hotels (e.g. [trippy.com](https://www.trippy.com)<sup>3</sup>). These systems are rarely seen to provide customized travel itineraries. Hence, we proposed a personalized itinerary recommender in [18] that leaves room for improvement. We enhanced that system by taking into account a few more factors that would potentially lead to a better itinerary. Initially, we allow for a flexible number of daily recommended POIs. Next, we improve our technique for selecting hotels and attractions so as to generate routes that involve shorter travel distance in total. Moreover, we consider the weather information and provide indoor POIs to cater for user needs for adverse weather. With all available routes, we can also provide users with information on the time that they would spend on commuting between sites.

A generic neural social collaborative ranking (NSCR) solution via the so-called "bridge" users (social media users who have accounts on two or three social networks) was proposed in [17], which seamlessly integrates user-item interactions of the travel domain and user-user social relations. Farseev et al. [5] proposed a recommendation framework C3R (cross-network collaborative recommendation framework) that utilizes both individual (user has visited in past) and group (Foursquare venue categories among user community members) knowledge to solve a task of venue category recommendation. Abel et al. [1] investigated the characteristic of tag-based profiles that resulted from tagging activities in social network websites such as Flickr, and cross-system user modeling strategies had significant impact on the performance of the recommendation quality within the scope of tag and resource recommendations in cold-start settings. The PERSTOUR algorithm was proposed in [9], which can reflect levels of user interest based on visit duration and demonstrate how POIs visit duration can be personalized using this time-based user interest.

The remainder of the paper is organized as follows. In Section 2, we briefly revisit our previous work and then examine the supporting techniques for our proposed recommender. This is followed by Section 3 in which we introduce the method for working out improved travel routes that for users. Evaluation metrics and experiments on the proposed recommendation technique are described in Section 4. Section 5 concludes the paper and identifies our future work.

## 2 Related Work

### 2.1 Previous Work

We proposed a personalized recommender for travel itineraries [18], which uses the K-Means clustering and tag-based recommendation algorithm. Once the user's travel plan is obtained, the recommender selects hotels according to user budget. Meanwhile, based on the favorite POIs specified by users, tags of all POIs are collected from travel websites and the recommender calculates the tag

<sup>2</sup> <http://trip.elong.com/tags/>

<sup>3</sup> <https://www.trippy.com/drive/>

similarity (between all POIs harvested from websites and user’s favorite attractions) to suggest POIs that users may be keen on. When selecting attractions, the user ratings are taken into account. Then, the recommendation system clusters these selected POIs and calculates distances between POIs and hotels. Finally, daily itineraries are generated and evaluated. The SI (the Shortest Itinerary), MEI (the Most Effective Itinerary), and MAI (the Most Appealing Itinerary) are then displayed on map for user selection.

## 2.2 Word2Vec

Word2Vec is a series of models that are used to produce word embedding, including the Skip-gram Model [6], Neural Net Language Model (NNLM), Continuous Bag-of-Words Model (CBOW) and Recurrent Neural Net Language Model (RNNLM) [11]. It takes as its input a large corpus of text and produces a vector space, typically of several hundred dimensions, with each unique word in the corpus being assigned a corresponding vector in the space. The techniques are proposed for measuring the quality of the resulting vector representations, with the expectation that not only will similar words tend to be close to each other, but that words can have multiple degrees of similarity [12]. However, the amount of relevant information in domain topics for automatically recognized is limited [11].

## 2.3 X-Means Clustering Algorithm

The X-Means clustering algorithm was proposed to solve the three major shortcomings of K-means [3][16] and it can estimate the value of K quickly. The steps of X-Means are described as follows [14]:

1. Pick one centroid, and then produce a new centroid nearby via running K-means to completion;
2. Calculate the resulting model scores (BIC: Bayesian information criterion [8] or MDL: Minimum description length [15]) and see if the scores are better;
3. If it does, accept the new centroid; otherwise, return to the previous structure;
4. Repeat the former steps until the rounds reaching KMax (Maximum for numbers of clusters).

In our paper, the X-Means clustering algorithm has been adjusted in Step 1 and 2. In Step 1, the method K-Means++ (see below) can be used for calculation of the initial centers. Moreover, in Step 2 we need to set a specified splitting criterion (tolerance) to control the process of splitting clusters. We measure the BIC as our scores. The steps of K-Means++ are presented as follows [2]:

1. Randomly select a data object as a center point from the data set;
2. Calculate the distance between all the data objects and the center points, respectively;
3. And then use the distance from object to the closest center to calculate the probability for each point, then select a data object according to the probability as another center point;
4. Repeat step 2 and 3, until required number of centers is initialized.

BIC is a likelihood criterion: let  $X$  be the dataset we are modeling; let  $M_i$  be the candidates of desired parametric models calculated as follows [14][8][2]:

$$BIC(M_i) = \widehat{l}_i(X) - \frac{p_i}{2} * \log N \quad (1)$$

where  $\widehat{l}_i(X)$  is the log-likelihood of the data according to the  $i$ th model and taken at the maximum-likelihood point, and  $p_i$  is the number of parameters in  $M_i$ .  $N$  is the sample size.

The maximum likelihood estimate (MLE) for the variance, under the identical spherical Gaussian assumption, is:

$$\widehat{\sigma}^2 = \frac{1}{N - K} \sum_i (x_i - \mu_i)^2 \quad (2)$$

The log-likelihood of the data is as follows:

$$l(X) = \sum_i \left( \log\left(\frac{1}{\sqrt{2\pi\sigma M}}\right) - \frac{1}{2\sigma^2} \|x_i - \mu_i\|^2 + \log \frac{N_i}{N} \right) \quad (3)$$

## 2.4 Tag-based Recommendation Algorithm

Being increasingly used in various networks, tags are seen as a potential source of user interest, preference, and user profile construction [8]. For example, a user may indicate her travel preferences for luxuries and arts. Meanwhile, the Marina Bay Sands happens to have been tagged with "luxury travel, family travel, nightlife". A match can therefore be made [17]. To some extent this helps alleviate the cold start problem for recommender systems. Users tag an item when they develop an interest in it. These tags serve as resources for describing such items. We term this associated information as attributes, most of which are discrete categorical variables for the web domain [4]. Others can also retrieve the item via its tags [7]. By calculating the similarity (use the cosine similarity equation [10]) between user tags, recommender system can predict the potential rating that a user would give to items, thus eventually being able to recommend the top-N items to her [13].

## 2.5 Summary

We combine the use of Word2Vec (w2v for short) with that of the X-Means clustering algorithm in our improvement. Firstly, we use the Chinese Wikipedia corpus to train our word embedding model and determine parameters, including the minimum count and the value of window, as 1 and 5, respectively. Before that, "mecab"<sup>4</sup> has been used to tackle the Chinese corpus. We need the w2v model to obtain similar tags of POIs. And then, the X-Means algorithm is applied for clustering sites and hotels based on their geolocation, which is intended for

<sup>4</sup> "mecab" is a tool that has been used natural language processing to segment words.

minimizing distances between each location in real world. After getting all tags of attractions that travelers may be interested in, the tag-based recommendation algorithm is used for generate the final recommended candidate list. In fact, we also use an existing tool — "Synonyms", which is utilized in obtaining near-synonym. And we compared it with the w2v model we trained and eventually we chose one with the better effect.

### 3 An Itinerary Recommender

In this section, we introduce our improvement on the previously proposed recommender system. Users inform the recommender of their preferences, including their destination, planned travel date, hotel budget, favorite attraction sites, and etc. According to the requirement, itineraries with recommended hotels and POIs are then presented to the users.

At first, the recommender collects hotels, POIs, and tags from tripadvisor.com, and then uses X-Means to cluster of the hotels and POIs according to their geolocations. Meanwhile, a trained w2v model is applied to calculate word similarity, which helps get similar tags of specific ones. In the phase of tag cleaning, the recommendation system divides all POIs into two categories, indoor POIs and outdoor POIs, based on their tags. According to user's favorite sites, interested tags and similar tags are used in tag-based recommendation algorithm, combined with the number of POIs' reviews, the rating of POIs, and the suggested visiting time, in order to produce a candidate sites list. Next, different clusters of all sites in the candidate list are counted. To recommend neighboring hotels with those sites, the recommender needs to calculate the distance between clusters of hotels and sites, and then selects the closest hotel cluster and the first 3 hotels from this cluster are picked up, with the highest rating and the price that suits user's hotel budget. The weather information is then taken into account for working out a daily itinerary. With all available routes, the recommender also provides users with information on the time that they would spend on commuting between sites. The workflow is shown in Fig. 1.

#### 3.1 Selection of Attractions

Having obtained user's favorite attraction sites, the recommender calculates the tag similarity<sup>5</sup> between the user's favorite sites and all the other attractions in the same city. Attractions are ranked in reverse order of their tag similarity and a threshold of the number of reviews is set. Because the number of travel days ( $d$ ) is provided by users, the recommender can select the first  $d$  attractions. With the same similarity, a site with a higher rating is chosen. If chosen attraction sites do not belong to different clusters, we continue to add the first  $d$  (the number of clusters) attractions to the candidate sites list, until all sites on the list can be divided into  $d$  clusters or the length of the list is equal to  $4*d$  (The default number of recommended daily attractions is 4).

<sup>5</sup> The cosine similarity equation is used to calculate tag similarity.



### 3.2 Planning of Daily Itineraries

After a candidate sites list is generated, weather information is taken into consideration for producing daily itineraries. According to user's travel date, weather information is harvested using the Xinzhi API<sup>6</sup>. However, the validity of weather information is time-dependent.

If adverse weather is expected during the travel period, for example a storm is on the way, the weather will be given top priority by our recommender when it works on generating daily itineraries. In the Sec. 3.1, two categories of attractions are labeled. If the number of days with adverse weather is equal to the number of clusters of indoor attractions, the recommender plans daily itineraries for indoor sites and outdoor sites, respectively. Finally, the system combines them in a whole daily itinerary. If the number of days with adverse weather is less than the number of clusters of indoor attractions, the same method is applied. In the final step of combining routes, the system needs to arrange indoor attractions on some days with fine weather.

After taking weather information into consideration, the recommender is expected to yield daily itineraries for users. If there is only one cluster in the candidate sites list, the recommendation system recommends four sites to users every day (the order of recommended sites is based on their geo distance), until all sites in the list are planned. If the number of clusters is equal to the number of travel days, the system plans itineraries for each cluster respectively, until all clusters are planned. If the number of clusters is greater than the number of travel days, excludes  $n^7$  clusters that not include user's favorite sites. However, if all clusters include user's favorite sites, excludes  $n$  clusters randomly. And then the method of planning itineraries is same to the above condition. If the number of clusters is less than the number of travel days, the system selects the first  $n$  clusters with the most number of attractions and plans itineraries. Finally, a whole daily itinerary is generated.

### 3.3 Selection of Hotels

In Sec. 3.2, the recommender generates a complete daily itinerary for users. Next, the system calculates distance between different clusters of attractions in the itinerary and all clusters of hotels, respectively. Clusters of hotels are ranked in order of their average distance between each hotel cluster and all clusters of POIs in the route, and the cluster with the smallest average value is chosen. In this cluster, the recommender selects hotels that their price is within user's hotel budget and the number of reviews is greater than a threshold, and then they are ranked in reverse order of ratings. The first three hotels are recommended. If there are less than three hotels, the system recommends all hotels that meet the requirements to users.

---

<sup>6</sup> <https://www.seniverse.com/>

<sup>7</sup>  $n$  = the number of clusters - the number of days



## 4 Experiments

We anticipated to recommend different kinds of itineraries that best meet the user’s travel needs. A series of experiments were therefore carried out to demonstrate the efficiency of the proposed recommender for itinerary planning.

### 4.1 Evaluation Metrics

We have used the same evaluation metrics with our previous work [18], including the tag similarity ( $tag\_sim_{rou}$ ), the travel distance ( $dis\_sim_{rou}$ ), and the efficiency ( $comp_{rou}$ ) that takes into account the hotel price, tag similarity, and the travel distance. The most Appealing itinerary is formed when  $tag\_sim_{rou}$  reaches its maximum. The shortest itinerary and the most effective one are formed when  $dis\_sim_{rou}$  and  $comp_{rou}$  reach their minimum, respectively.

### 4.2 Methodologies, and Experimental Settings

We used a desktop Dell with a 64-bit operating system. The CPU is intel(R) Core(TM) i7-7700, and the software supporting our experiments includes Python3.5 and MongoDB. We also utilized the package googlemap to generate distances and commuting time between all sites and hotels. The Xinzhi Weather API was used to acquire weather information around the world.

We simulated the requirements of users Alva, Bob, Carlo, Peter, and Tom, and different travel questionnaires are filled out for them, which are shown in Tab. 1 (BJ: Beijing, HZ: Hangzhou, WH: Wuhan, SZ: Suzhou, QZ: Quanzhou). Information on 2017 sites and 8885 hotels was harvested from tripadvisor.cn and then saved in MongoDB. "Synonyms" was a python package that was used to get similar tags. In addition, we applied X-Means clustering the hotels and sites that were distinguished with cities. After clustering several times, the best parameters of each city have been picked up, which have shown in Tab. 2. And Fig. 2 - Fig. 6, illustrate clustering results.

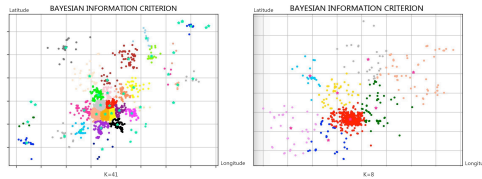


Fig. 2. BJ: the clustering result of hotels (left) and of attractions (right)

### 4.3 Results and Analysis

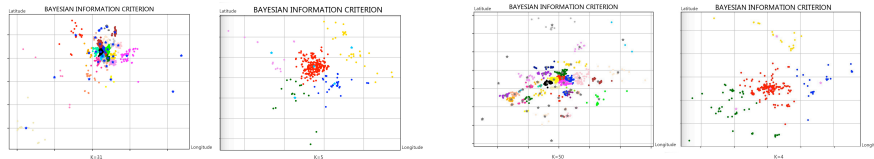
Taking Hangzhou as an example, the itineraries for Bob have been displayed in Tab. 3. We also evaluated them in terms of the metrics introduced in Sec. 4.1. The result is shown in Tab. 4.

**Table 1.** Travel Questionnaire

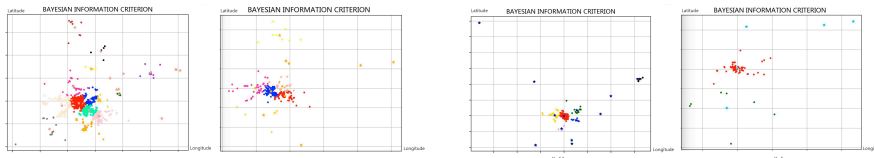
Name	City	No. of Days	Travel Dates	Hotel Budget	Favorite Sites	Weather
Alva	BJ	5	2018/5/24	350	Forbidden City Summer Palace Temple of Heaven	Sunny, Cloudy, Thundershower, Sunny, Sunny
Bob	HZ	3	2018/5/24	200	West Lake Zhejiang University Lingyin Temple	Cloudy, Overcast, Shower
Carlo	SZ	3	2018/5/24	200	Lingering Garden Lion Forrest Humble Ad's Garden	Cloudy, Moderate Rain, Overcast
Peter	WH	2	2018/5/24	200	Yellow Crane Towel Wuhan University	Cloudy, Moderate Rain
Tom	QZ	2	2018/5/24	180	Guandi Temple East Lake Park	Cloudy, Sunny

**Table 2.** Parameters in the clustering process for hotels and POIs

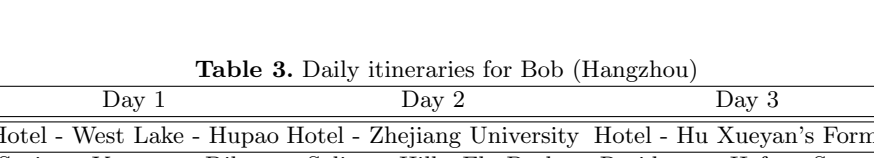
Datasets	Hotels					POIs				
City	BJ	SZ	QZ	HZ	WH	BJ	QZ	SZ	HZ	WH
KMax	100	50	100	100	100	100	100	100	100	100
tolerance	0.01	0.01	0.01	0.001	0.1	0.75	0.0001	0.09	0.25	0.5



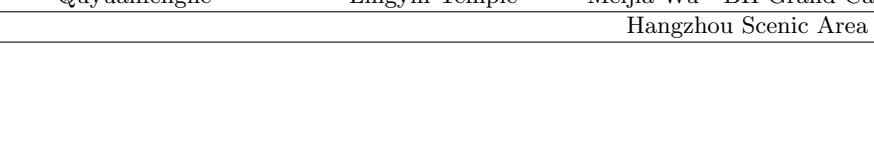
**Fig. 3.** HZ: the clustering result of hotels (left) and of attractions (right)



**Fig. 4.** SZ: the clustering result of hotels (left) and of attractions (right)



**Fig. 5.** WH: the clustering result of hotels (left) and of attractions (right)



**Fig. 6.** QZ: the clustering result of hotels (left) and of attractions (right)

**Table 3.** Daily itineraries for Bob (Hangzhou)

Day 1	Day 2	Day 3
Hotel - West Lake - Hupao Spring - Yanggong Dike - Quyuanfenghe	Hotel - Zhejiang University - Solitary Hill - Fly Peak - Lingyin Temple	Hotel - Hu Xueyan's Former Residence - Hefang Street - Meijia Wu - BH Grand Canal Hangzhou Scenic Area

**Table 4.** The result of evaluation for Bob (Hangzhou)

Hotel	$Tag\_sim_{rou}$	$Dis\_sim_{rou}$	$Comp_{rou}$
Hofang Youth Hostel	5.70514	71.4	1.16293
Friendship Hotel	5.70514	82.7	1.21442
Silver Bridge Hotel	5.70514	83.2	1.21715

As shown in Tab. 4 we acquired a shortest itinerary for Bob, which sets off from Hofang Youth Hostel and a highest performance/price ratio itinerary that also sets off from Hofang Youth Hostel. Finally, we compared these itineraries for Bob given by our proposed recommender with the routes presented in [18]. Only the first 2 daily itineraries of Bob’s recommendation results are considered, because in the previous experiment of only 2 daily itineraries were provided. From Tab. 5, we can tell that the itinerary provided by this recommender is shorter in distance and higher efficiency.

**Table 5.** A comparison between itineraries provided by our previous recommender and the current one

Itinerary	$Tag\_sim_{rou}$	$Dis\_sim_{rou}$	$Com_{rou}$
previous system	3.90305	53.1	1.21715
improved system	3.73395	28.3	1.01377

## 5 Conclusions

We have improved our recommender of personalized itineraries in the sense that more factors could now be taken into account, such as the suggested visiting time for each POI and future weather data. An X-Means clustering algorithm was used to cluster all sites and hotels to help minimize the entire distance involved in the travel among them. Then a Word2Vec model was applied to obtain several similar tags of specific ones. In addition, a tag-based algorithm was utilized to create a list of candidate attractions that best match with the user’s favorite sites. Eventually the itineraries that would meet different user needs were created and the results were displayed in a format with complete distances and commuting time for user selection.

We have performed experiments with a few cities in China, including Beijing, Hangzhou, Suzhou, Wuhan, and Quanzhou, and found out some parameters for clustering to help produce the shortest itinerary. In our server (118.89.196.180:8888), we will publish a simple application of travel recommender for tests as soon as possible.

In the future, we plan to enhance accuracy in finding similar tags by training corpus in travel domain. Some conditions related to weather should be detailed in order to obtain more flexible itineraries.

## References

1. Abel, F., Araújo, S., Gao, Q., Houben, G.J.: Analyzing cross-system user modeling on the social web. In: International Conference on Web Engineering. pp. 28–43. Springer (2011)
2. Andrei, N.: Center initializer. [https://github.com/annoviko/pyclustering/blob/master/pyclustering/cluster/center\\_initializer.py](https://github.com/annoviko/pyclustering/blob/master/pyclustering/cluster/center_initializer.py) (2015)
3. Bishop, C., Bishop, C.M., et al.: Neural networks for pattern recognition. Oxford university press (1995)
4. Duraao, F., Dolog, P.: A personalized tag-based recommendation in social web systems. *Adaptation and Personalization for Web 2*, 40 (2009)
5. Farseev, A., Samborskii, I., Filchenkov, A., Chua, T.S.: Cross-domain recommendation via clustering on multi-layer graphs. In: Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval. pp. 195–204. ACM (2017)
6. Goldberg, Y., Levy, O.: word2vec explained: Deriving mikolov et al.’s negative-sampling word-embedding method. arXiv preprint arXiv:1402.3722 (2014)
7. He, X., Chua, T.S.: Neural factorization machines for sparse predictive analytics. In: Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval. pp. 355–364. ACM (2017)
8. Kass, R.E., Wasserman, L.: A reference bayesian test for nested hypotheses and its relationship to the schwarz criterion. *Journal of the american statistical association* 90(431), 928–934 (1995)
9. Lim, K.H., Chan, J., Leckie, C., Karunasekera, S.: Personalized trip recommendation for tourists based on user interests, points of interest visit durations and visit recency. *Knowledge and Information Systems* 54(2), 375–406 (2018)
10. Marlow, C., Naaman, M., Boyd, D., Davis, M.: Position paper, tagging, taxonomy, flickr, article, toread. In: In Collaborative Web Tagging Workshop at WWW’06. Citeseer (2006)
11. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781 (2013)
12. Mikolov, T., Yih, W.t., Zweig, G.: Linguistic regularities in continuous space word representations. In: Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. pp. 746–751 (2013)
13. Nakamoto, R., Nakajima, S., Miyazaki, J., Uemura, S.: Tag-based contextual collaborative filtering. *IAENG International Journal of Computer Science* 34(2) (2007)
14. Pelleg, D., Moore, A.W., et al.: X-means: Extending k-means with efficient estimation of the number of clusters. In: *Icml*. vol. 1, pp. 727–734 (2000)
15. Rissanen, J.: Modeling by shortest data description. *Automatica* 14(5), 465–471 (1978)
16. Ro, D., Pe, H.: *Pattern classification and scene analysis* (1973)
17. Wang, X., He, X., Nie, L., Chua, T.S.: Item silk road: Recommending items from information domains to social users. In: Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval. pp. 185–194. ACM (2017)
18. Yajie, G., Jing, Z., Hanwen, F., Anying, C., Shouxun, L.: A recommender for personalized travel itineraries (in press). In: Proceedings of the 4th International Conference on Cloud Computing and Security (2018)