A roadmap for privacy preserving tourist recommendation system

Alan J. Wecker[1,*,†], Noa Tuval[1,*,†], Alain Hertz[2,*,†], Muhammad Mahamid[1,*,†] and Tsvi Kuflik[1,*,†]

[1]Univeristy of Haifa, 199 Abba Khosuhy Avenue, Haifa, 3498838, Israel
[3]Polytechnique Montreal,, 2500 Chem. de Polytechnique, Montréal, QC H3T 1J4, Canada

Abstract

Users' privacy is one of the main concerns of users who use recommender systems in general and tourist recommender systems in particular, due to the fact that they must share personal information (like preferences and location) with the system in exchange for recommendations. The personal information collected by the system is used for creating user models used for personalization of recommendations, but may be used and / or shared or sold to 3rd parties. Still, when considering content-based recommender systems, the situation may be different if the user's model is built, maintained and stored locally on the user's device/personal cloud. The paper presents a simple yet effective privacy preserving content-based recommender system architecture that uses a hypercube-based model for representing user preferences.

Keywords

Hypercube-based recommender, system, privacy preserving, recommender system, content-based recommender system

1. Motivation

Since the early days of recommender systems, users' privacy became a major concern. Users have to share relevant private information with recommender systems in order to receive personalized service. This can be done explicitly and willingly - by answering a questionnaire, ticking a check-box etc' or implicitly and unknowingly, simply by browsing a website where the recommender system builds a user model based on the user's behavior. The risks for users' privacy in recommender systems attracted considerable research efforts over the years, as can be seen in a recent review [1]. In their review, the authors analyze threats and issues for users' privacy in recommender systems, as they appeared in previous studies and consider three types of solutions that are offered: Architecture based solution for data leakage, algorithm based data protection mechanism and ethical guidelines and regulations for privacy. The authors refer mainly to the prevailing recommender systems technology - collaborative filtering "A RS is designed in such a way that it allows users to give their input for identifying their preferences and mapping these attributes to their neighborhood for an accurate prediction and robust recommendation [2]." With respect to these systems, the authors also discuss the idea of anonymization, but note that it does not guarantee privacy as it can be breached. While this is the main strength as well as limitation of collaborative filtering approach, there are also content-based recommender systems that rely solely on users' preferences for the recommended item's attributes. Here, there is no real need to share any information with the recommender system - all is needed is to build a user model locally, get a database of items and find the most appropriate one without any additional interaction

Workshop on Recommenders in Tourism (RecTour 2024), October 18h, 2024, co-located with the 18th ACM Conference on Recommender Systems, Bari, Italy.
*Corresponding author.
†These authors contributed equally.
ajwecker@gmail.com (A. J. Wecker); noa.tuval@gmail.com (N. Tuval); alain.hertz@gerad.ca (A. Hertz); mohammed.89a@gmail.com (M. Mahamid); tsvikak@is.haifa.ac.il (T. Kuflik)
https://cris.haifa.ac.il/en/persons/alan-wecker (A. J. Wecker); https://www.polymtl.ca/expertises2/hertz-alain (A. Hertz); https://is-web.hevra.haifa.ac.il/index.php/en/prof-tsvi-kuflik (T. Kuflik)
0000-0003-4914-8949 (A. J. Wecker); 0000-0003-0096-4240 (T. Kuflik)

with the recommender system [3]. Still, when a selection/purchase is made through the system, the recommender system may infer the user's preferences. However, if the purchase is made directly from the service provider, the user's privacy is preserved. We follow this line of thought - in order to protect the user's privacy while interacting with recommender systems, we advocate the use of they content-based technique for recommender system, while the purchase is done directly from the provider and not via a recommender system [4]. In this short paper we propose a simple yet effective content-based recommender system architecture, based on a hypercube model [5] and present our current implementation (work in progress) and planned evaluation in the tourism domain. See [6] for examples of such applications in the tourist domain .

2. System overview

The system consists of two parts. The first part sits on a server(s) and is concerned with generating the non-personalized (generic) information about the restaurants and constructing the hyper-cube. The second part is located on the user's device and is concerned with the user interface, construction of the user profile, and using the user profile to search the hypercube. The server contains a json file which describes each of the restaurants' features and additional information about the restaurant which could be useful for the user (but not necessarily for the collaborative filtering).The features are translated into a binary vector. Thus the multi-valued cuisine type (Turkish, Chinese, Fast Food...) feature is translated into a number of binary (Yes, No) features (isTurkish, isChinese, isFastFood ...). Once a restaurant is translated into a feature vector, a hypercube is built and the original information about the restaurant is sent to the user. The user device application consists of the user interface which shows the user a ranked list of restaurants (sorted by hypercube recommendations, with a rank score), which he can filter for location and features they are presently interested in (this fine tunes the recommendations by allowing the user to express contextual concerns which are appropriate for this particular visit). They can select a restaurant to get more detailed information than is contained in the listing. In addition, they can evaluate a restaurant that they like by giving it a score. These scores are used to determine the user profile as described in the next sections,

3. Recommendation engine description - Hypercube

The hypercube-based model for CB recommender systems is a graph-based approach with multiple Boolean features. The system is represented by a hypercube-graph Qn in which edges exist between nodes that differ in one bit (Hertz et al., 2021). The graph vertices represent the items in the system which are characterized by n attributes. Each item is associated with a vector of attributes indicating for each attribute whether the item has the attribute or not, as shown in Figure 1 below, for n=3. For example, for a set of tourist attractions with n = 3 Boolean attributes, the first one being 'low-price', the second one 'unique landscapes', and the third one 'accessible', then vector (1,1,0) is associated with low price tourist attractions located in unique landscapes but are not accessible to the disabled.

Each user has his preferences, represented by the user model wu which is a vector of attributes associated with the user. wu indicates for each attribute whether the user likes the attribute, doesn't care about it, or dislikes the attribute. So, the attributes have 3 possible values: Yes, Don't care (no opinion), No (or 1, 0, -1, respectively). For the above example, the vector (0,1,1) would be associated with all users who do not care about the price of a tourist attraction, if it is in unique landscapes and accessible to people with disabilities.

4. Recommendation engine implementation

The user model is inferred by the system based on the user's past ratings. A maximum score given by a user to an item i means that he or she likes all attributes in i, where the minimum score means that the user does not like the attributes in the item. To estimate the user model, we first need to transform a

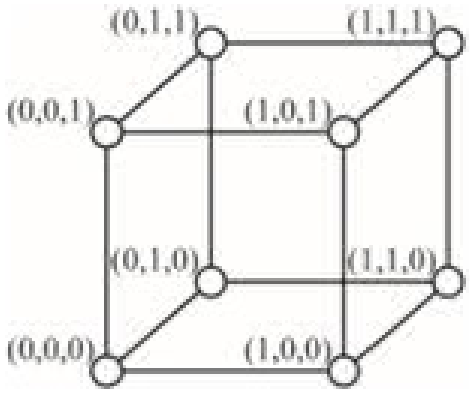

Figure 1: The hypercube graph Q3

rating into a number of attributes that do not fit with the user preferences. In the formula below, the rating ri of item i which is given using an s-star scale (i.e., a rating in 1, 2, …., s) is converted by function τ into a distance δi ∈ [0,n] to the user preferences, where n is the number of attributes characterizing an item [5]. For example, for a 5-star rating of items with n = 20 Boolean attributes, we have τ(1) =

$$\delta_i = \tau(r_i) = n - \frac{n(r_i - 1)}{s - 1}.$$

20, τ(2) = 15, τ(3) =10, τ(4) = 5 and τ(5) = 0. The distance between an item and a user model is defined as the number of attributes that the item has and the user does not like, plus the number of attributes that the item does not have and the user likes, while ignoring the attributes that the user does not care about. For a vector wu corresponding to the user model of u and a vector vi associated with an item i, this distance is denoted d(vi,wu). For example, for wu=(0,1,1) and vi=(1,1,0), we have d(vi,wu)=1 since the first attribute is ignored and the user likes the third attribute that is not present in the item. Now, if a vector x corresponds to the user model wu of user u, then the rating ri of each item i rated by the user u (converted to distance δi) should be equal to the distance between x and the vector vi associated with item i. Therefore, to estimate the user model, we determine a vector x that minimizes which is the

$$\sum_i \left| d(\boldsymbol{v}^i, \boldsymbol{x}) - \delta_i \right|$$

cumulative error over the rated items, that results from imprecise user's ratings. We model this problem as an integer linear program. The optimal solution can be obtained in one or two seconds (Hertz et al., 2021). In order to recommend to the user items that suit her or his preferences, i.e. closest to the user model, we estimate the rating ri given by u to item i by computing a predicted rating ρi as follows: where vi is the vector associated with item i, and x is the vector associated with our estimate of the

$$\rho_i = s - \left\lfloor \frac{d(\boldsymbol{v}^i, \boldsymbol{x})(s-1)}{n} \right\rfloor$$

user model of u. For example, for a 5-star rating, x = (0,1,1) and vi = (1,1,0) we get ρi = 5 - 1 = 4. Finally, the hypercube-based recommender system offers the user the items with the highest predicted ratings.

5. Hypercube Advantages

Graphs are applied for user modeling as they provide convenient structures for representing complex relationships among users and items, integrating entities and links without loss of information [7] [8]. Our model uses the hypercube graph that allows defining distances to indicate the similarity between items, specifying the difference between the item's features and the features desired by the user, and predicting rating of an item. The hypercube-based model is suitable for application in areas such as restaurants and tourism, that allow the representation of an item through many features. As shown in [5], using a hypercube-based model is helpful in obtaining accurate predictions, even when the number of user ratings is small. Moreover, as a CB recommender system, the hypercube model does not require users to share their information with a system in order to generate recommendations for other users, so user privacy is preserved.

The hypercube differs from other distributed systems such as[9] in that the user's data does not need to be shared. This derives from the hypercube being a CB system as opposed to the distributed collaborative filter system (In fact [9] disputes our basic assumption and claims that users do not care about sharing tourist preferences, since they are seen in these public places)

6. Practical considerations to work on mobile device

At present, the user profile construction and the search for a match on the hypercube are done on the server. The major problems in migrating them to a less powerful user device are discussed below:

6.1. ILP on mobile

The setup heavily relies on integer linear programming and uses a python package (GLPK) to solve problems involving user profile construction and ranking the user preferences as discussed above. We are presently looking for a package to run on Android or we may port the GLPK to something that can run on Android.

6.2. Feature Reduction

Clearly, a server is much more powerful than a mobile device, therefore computational complexity is of importance. In addition, since many multi-varied features get translated to a number of binary features, there is an explosion of features. In the restaurant example we are working on we have approximately 165 features, 48 of which have only one item matching them (while this can be useful for filtering to find unique restaurants, they are useless in collaborative filtering to find similar items. Thus by removing them from the hypercube we can speed up computations. If we consider the group of features from among 450 restaurants that have a count of less than 5 we can reduce by 50 percent, but this maybe a too radical reduction

7. Problems

7.1. How can we do personalized ads

What is the economical model that would make someone want to provide this service? Normally personalized ads and exposures pay for these kinds of services. One could do a model of paying to be part of the hypercube but that doesn't reflect real value to the restaurant. A possible solution is to send a generic package of 10-100 adds, the user-side application knowing the user profile could pick appropriate ads (the ads could have vectors similar to restaurants How can we order directly or how do we know the system is effective If as discussed in the introduction the user orders directly to the restaurant, how does the service know how effective it has been and hence its value. The restaurant can know the value as a direct result of the order (maybe containing a value of who the referral is from.

However because of privacy concerns this information is withheld from the service. Thus this is a price negotiation problem where only one side has true knowledge. Perhaps by limiting the number of places in the cube and having restaurants bid for a place on the cube, they have incentive to ask for the true worth..

7.2. How do we share data among multiple personal devices

Today many people have multiple devices, how can user profiles be shared among different devices? We don't want this information on a central server; so perhaps some form of encrypted peer to peer communication would be useful in allowing users to share profiles and migrate from and to different devices,

7.3. Cold Start problem

How do we get a user profile when the user has not rated any restaurants. Our present solution involves having the user rate 6 restaurants that they are familiar with before beginning to work with the system. The user profile is then built up over time as the user visits more and more restaurants.

8. Evaluation Plan

We plan to evaluate our system by giving two systems to the user to evaluate (the order being balanced so to counter first use effects). The first gives the restaurants in a generic order as provided by the services' generic ranking; the second system gives them the personalized hypercube order. After experiencing each system we ask questions about satisfaction and to evaluate 6 of the recommendations for accuracy and usefulness At the end we ask some comparative questions. This is to determine whether the system brings reasonable results, maybe not as precise as some of the state of the art techniques, but with the advantage of preserving the user's privacy. We also will try to examine users attitudes towards privacy to determine if this is a needed consideration.

We also plan on comparing the methods computational complexity to other RS algorithms to show that the hypercube has a comparatively low overhead. In particular we will focus on content based methods.

9. Conclusion

The Hypercube-based content-based recommendation model enables users to preserve their privacy while seeking recommendations for services/items to purchase. It was already shown that it may be effective with a small number of ratings. Using this approach, a user profile is kept locally on the user's device (or private cloud) and there the section of item/service is performed. The hypercube-based model provides a compact and effective representation for content based recommendations. The planned study will focus on demonstrating its effectiveness and will seek to understand users' perceptions.

Acknowledgments

This study is partially supported by the Israeli science foundation, grant number 216/23

References

[1] Y. Himeur, S. S. Sohail, F. Bensaali, A. Amira, M. Alazab, Latest trends of security and privacy in recommender systems: a comprehensive review and future perspectives, Computers & Security 118 (2022) 102746.

[2] M. P. O'Mahony, N. J. Hurley, G. C. Silvestre, Recommender systems: Attack types and strategies, in: AAAI, 2005, pp. 334–339.
[3] S. Puglisi, J. Parra-Arnau, J. Forné, D. Rebollo-Monedero, On content-based recommendation and user privacy in social-tagging systems, Computer Standards & Interfaces 41 (2015) 17–27.
[4] T. Kuflik, K. Poteriaykina, User model on a key, in: Proceedings of the 20th ACM conference on Hypertext and hypermedia, 2009, pp. 371–372.
[5] A. Hertz, T. Kuflik, N. Tuval, Resolving sets and integer programs for recommender systems, Journal of Global Optimization 81 (2021) 153–178.
[6] D. Gavalas, C. Konstantopoulos, K. Mastakas, G. Pantziou, Mobile recommender systems in tourism, Journal of network and computer applications 39 (2014) 319–333.
[7] C. C. Aggarwal, et al., Recommender systems, volume 1, Springer, 2016.
[8] A. Tiroshi, S. Berkovsky, M. A. Kaafar, D. Vallet, T. Kuflik, Graph-based recommendations: Make the most out of social data, in: User Modeling, Adaptation, and Personalization: 22nd International Conference, UMAP 2014, Aalborg, Denmark, July 7-11, 2014. Proceedings 22, Springer, 2014, pp. 447–458.
[9] F. Beierle, S. Egger, Mobrec—mobile platform for decentralized recommender systems, IEEE Access 8 (2020) 185311–185329.