A recommender system for cold start items: A case study in the real-estate industry

Qian Zhang, Di Zhang, Jie Lu, Guangquan Zhang
Decision Systems and e-Service Intelligence Laboratory,
Centre for Artificial Intelligence
University of Technology Sydney, Australia
Email: Qian.Zhang-1@uts.edu.au,
Di.Zhang-7@student.uts.edu.au,
Jie.Lu@uts.edu.au, Guangquan.Zhang@uts.edu.au

Wei Qu, Mark Cohen
Domain Holdings Australia Limited,
NSW, Australia
Email: wei.qu@domain.com.au, mark.cohen@domain.com.au

***Abstract*—The recommender systems provide users with what they prefer and filter unnecessary information. In the fierce marketing environment, it is crucial to recommend items to users in an early stage to keep user's interests and loyalty. With the fast product renewal, classical recommendation methods such as collaborative filtering cannot handle the cold-start item problem. In many real-world applications, content information of items or users is available and can be used to assist recommendation. Besides, user may interact with the items in different behaviors such as view, click or subscribe. How to use the complex content information and multiple user behaviors are real problems that are not well solved in applications. In this paper, we propose a content-based recommender system to deal with the practical problem. Boosting tree model also added to the system to avoid potential Spam. We applied our developed method to real-estate application to recommend new property which just landed into the market to users. Experimental results with three data subsets and three recommendation scenarios demonstrate that the proposed method can outperform the baseline on recommendation accuracy. The results indicate that our method can effectively reduce potential Spam to users, so that user experience will be improved.**

***Index Terms*—recommender systems, cold start, content-based recommender systems, data engineering**

I. Introduction

The rapid development of technology enables data accumulation much faster than before, which brings us huge amount of information whether or not we actively seek it. This is recognized as 'information overload' problem which clearly affect the e-business [1]. In the fierce competition, it is very crucial for the companies to deliver information to the right customers at the right time to keep their loyalty. For example, in e-commerce, especially for the electronic devices such as cellphones or computers, the upgrade of products happens so quickly that customers will lose their interests and the product will be replaced by other new types. Another example is in the real-estate industry, properties last around a month on the market before it is sold, while customers spend a long time searching information before making their decisions. It is better if a new property can be recommended to customers once the property enters the market. Delivering the information in an early stage helps companies win the edge in the dynamic market [2], [3].

A key enabler is to optimize the search results and provide recommendation for customers. The demanding of personalization leads the development of recommender systems to different applications [4]. Recommendation techniques in literatures are divided into four categories: content-based, collaborative filtering (CF)-based, knowledge-based and hybrid methods [5]. Collaborative filtering is one of the most widely used method in recommender systems and received great success in both academia and industry [6]. It is typical to apply CF in mature website or platform such as Netflix. However, the user-item interactions are not always abundant in many business scenarios. Or if recommendation is on new items which just landed in the market, CF will suffer the cold-start problem and the recommendation results are not accurate or even cannot be generated [7]. In this scenario, we should rely more on content-based recommendation techniques [8]. Moreover, unlike recommendation on movies, rich information about users and items is available in many scenarios such as real-estate recommendation, job recommendation or health decision support. The content information matching is very crucial in these recommendation processes mentioned above [9].

There are three challenges when building content-based recommender systems where user-item interactions are not sufficient, such as for the real-estate recommendation: 1) The content information is heterogeneous therefore it is difficult to build user/item profiles. The content information usually contains categorical features, numerical features, free-text features or even images. How to do the data engineering and fuse all these features in to the user/item profile remains one problem to be solved. 2) The interactions between users and items are in multiple types, such as view, click, subscribe or enquiry. How to define the level of preference these behaviors represent is not easy and will affect the accuracy of recommendation. 3) In the real business, the spam of recommendation is taken very seriously. When pushing notifications to the users according to recommendation generated, users will quickly lose interests if they receive irrelevant information. This greatly damages user loyalty to the business.

To deal with the challenges above, we propose a content-

based recommender system for the business scenario where user-item interactions are not sufficient but content information is available and apply it to the area of real-estate recommendation. Although the interactions on those cold-start items are not sufficient, still, we can use the content information to recommend these items to potential users. A data processing and feature engineering method is proposed for dealing with the heterogeneous item content information. As for the multiple user behaviors, we define a neural network to learn the global weights for each type of the user behavior. All these steps above are integrated into a framework of the system is proposed to support generating personalized recommendation and applied to the real-estate business. Moreover, we use the boosting tree model to predict the relevance of items to users to prevent potential spam when we push notifications to users. This paper has contributions on both recommendation theories and practical applications.

The remainder of the paper is as follows. Section 2 reviews related literatures on recommender systems and content-based recommender systems. Section 3 presents the proposed content-based recommender system. Experiments and analysis are demonstrated in Section 4. Finally, conclusion and further study are given in Section 5.

II. Related Work

In this section, we focus on reviewing recommender systems and content-based recommender systems.

A. Recommender systems

The explosive growth in information on the web and the rapid increase in e-services has given users a huge amount of choice, which may make decision making more complex. Recommender systems are primarily devised to assist individuals who are short on experience or knowledge deal with the vast number of choices in relation to items [10]. Recommender systems take advantage of various sources of information to predict preferences of users in relation to different items [11]. This area of research has been the focus of great interest for the past twenty years from both academia and industry. Research in this field is motivated by the potential profit recommender systems have generate for businesses such as Amazon [12]. Recommender systems were first applied in E-commerce to solve the information overload problem caused by Web 2.0 and were quickly expanded to the personalization of e-government, e-business, e-learning, e-tourism [1]. Nowadays, recommender systems are an indispensable part of Internet websites such as Amazon.com, YouTube, Netflix, Yahoo, Facebook, Last.fm, and Meetup.

In brief, recommender systems are designed to estimate the utility of an item and predict whether it is worth recommending. The core part of recommender systems is a function to define the utility of a specific item to a user [13]. A final recommendation list containing a set of items in a ranked order will be provided. This list is ranked according to the utility of all the items the user has not consumed. Basically, the utility of an item is presented as ratings of a user. Recommender systems is to find an item for the user to maximize the utility function. Predicting the utility of items to a particular user varies in different recommendation algorithms. Referencing the classical taxonomies of previous research [1], [13], [14], recommendation techniques are categorized in four types: content-based, collaborative filtering-based, knowledge-based and hybrid approaches.

B. Content-based recommender systems

As the name suggests, content-based recommender systems make use of the content of an item's description to predict its utility based on the user's profile [15]. Content-based recommender systems aim to recommend items that are similar to those in which a specific user was interested previously. The recommendation process of content-based recommender systems is as follows [16]: Item representation, user profiling and filtering and recommendation.

Content-based recommender system has several advantages as illustrated in previous studies [17]. First, content-based recommendation is based on item representation, so it is user independent. As a result, this kind of system does not suffering from the data sparsity problem, a serious problem in collaborative filtering-based recommender systems. Second, content-based recommender systems are able to recommend new items to users so the system can solve the new item cold start problem. Lastly, content-based recommender systems can provide a clear explanation of the recommendation result. The transparency of this kind of system is a great advantage compared to other kind of techniques in real-world applications.

On the contrary, there are several limitations in content-based recommender systems [13], [18]. To begin with, although the new item problem can be solved or alleviated using content-based recommender systems, they suffer from the new user problem as the lack of user profile information will seriously affect the accuracy of the recommendation result. Furthermore, a content-based system will always choose similar items for users, leading to overspecialization in recommendation. A user tends to become bored with similar recommendation lists because most users want to learn about new and fashionable items rather than being limited to items similar to those they have previously used. Finally, items sometimes cannot easily be represented in the specific form required by content-based recommender systems. This kind of system is more suitable for articles or news recommendation rather than images or music.

III. Two stage content-based recommender system

In this section, we present a two stage content-based recommender system. It opens with several frequently used notations and the formulation of the recommendation problem we aim to address. Our proposed method, which settles the problem of cold-start item recommendation, is then presented.

A. Notations and Problem Formulation

We use bold uppercase letters, such as $\boldsymbol{X}$, to denote matrices. The i-th row and the j-th column in the matrix are denoted

TABLE I
TYPES OF USER BEHAVIORS.

Level	User Behavior Type	User Behavior
1	Click	AdvertView
2	Detailed View	FloorPlanView
		InteractiveFloorPlanView
3	Bookmarked	Shortlisted
		AutoShortlist
4	Enquiry	EmailEnquiry
		SMSEnquiry

as X_{i*} and X_{*j}, respectively. The (i,j)-th element of matrix $\boldsymbol{X}$ is denoted as $X_{i,j}$. Suppose we have M users and N items in our system, $\boldsymbol{X} \in \mathbb{R}^{M \times N}$. One user i may have interacted with one item j in different behaviors. These behaviors are represented by a set $B = \{b_1, ..., b_p, ..., b_P\}$, in which each of the element represents one kind of interaction behavior such as view, click or enquiry. The types of user behaviors are listed in Table I. And the interaction matrix on user behavior b_p is represented as $\boldsymbol{X}^{b_p}$. The interaction between user i and item j is represented as $X_{i,j}^{b_p}$. If the user i has interacted with item j on the behavior type b_p, $X_{i,j}^{b_p}$ is 1, otherwise 0.

Given an item j, it has L features on the feature set $F = \{f_1, ..., f_l, ..., f_L\}$ according to its content, where L is the number of features in the item profile. The features are extracted from item attributes which are in different values. Some attributes are categorical value, some are numerical values. We will talk more about how to do feature extraction in the next section. All the M items feature vectors are composed as item profile matrix, represented as $\boldsymbol{V} \in \mathbb{R}^{N \times L}$. The jth row $\boldsymbol{V}_{j*}$ represents the content vector of item j.

The problem solved in this paper is based on the assumption that user-item interactions are very sparse and the interaction behavior are in multiple types. According to the descriptions above, we have formulated our problem as follows: Given user-item interaction matrixes on multiple user behaviors: $\boldsymbol{X}^{b_1}$, ..., $\boldsymbol{X}^{b_p}$, ..., $\boldsymbol{X}^{b_P}$ and item profile matrix $\boldsymbol{V}$, the content recommender system is to use the item content information to assist recommendation on new items where they have no interactions with users in the system.

B. Two Stage Recommendation Method

Our method consists of two stages, as shown in Fig. 1:

1) We build content-based recommendation method that can select the top 50 users for each property. This can be used as a final recommendation list or as a set of candidate users, much smaller than the whole user set.
2) For each (candidate-user, item) pair, the probability that candidate-user will interact with this item is learned and this is used for preventing potential Spam when we push notifications to users.

1) Memory-based content recommendation: Different attributes are processed and features are extracted to represent the item. We extract these features and process the values to numerical value 0 and 1. As a result, an item profile encoded with $\{0, 1\}$ has been obtained. For H item attributes, we extract L features. For the categorical values and numerical values in item attributes, we use one-hot encoding to encode the features as categorical. Note that L and H are different since each of the attributes contains various feature dimensions because of the one-hot encoding. The H attributes $A = \{a_1, ..., a_h, ..., a_H\}$ are corresponding to L feature dimensions $D = \{d_1, ..., d_h, ..., d_H\}$ where $\sum_{h=1}^{H} d_h = L$. For $\forall\ V_{j,l}$ in item profile $\boldsymbol{V}$, $V_{j,l} \in \{0, 1\}$. Take the real-estate business as an example, the attribute for one property is the property type. Suppose that there are five property types: {house, unit, apartment, townhouse, villa}. If for item j, it is an apartment. The one-hot encoding vector is $[0, 0, 1, 0, 0]$. This one attribute will be encoded into a five-dimension feature.

Content-based recommender systems profile a user's preference from items in a user's consumption records. In our developed method, user profile is built according to users' historical data. For each user, their historical data contains difference user behaviors as user-item interaction matrixes $\boldsymbol{X}^{b_p}$. We firstly consider one user behavior denoted as $\boldsymbol{X}$, and user profile is generated as follows:

$$\boldsymbol{U} = \boldsymbol{X} \cdot \boldsymbol{V}. \tag{1}$$

For user profile $\boldsymbol{U}$, it has the same structure as item profile $\boldsymbol{V}$. The difference is that the user profile is an aggregation result where for the entry $U_{i,l}$ in $\boldsymbol{U}$, $U_{i,l} \in \mathbb{N}^+$. To normalize values in $\boldsymbol{U}$ to $[0, 1]$, $\boldsymbol{U}$ can be divided into H parts according to H attributes $\{a_1, ..., a_h, ..., a_H\}$. As a result, $\boldsymbol{U}$ is divided to $\{\boldsymbol{U}_1, ..., \boldsymbol{U}_h, ..., \boldsymbol{U}_H\}$ where $\boldsymbol{U}_h \in \mathbb{R}^{M \times d_h}$. The normalization of user profile is:

$$(\boldsymbol{U}_h)_{i,l} = \frac{(\boldsymbol{U}_h)_{i,l}}{\sum_{l=1}^{d_h} (\boldsymbol{U}_h)_{i,l}}. \tag{2}$$

To take each type of the user behaviors $\boldsymbol{X}^{b_p}$ into consideration, we assign weights to indicate the importance of the behaviors. These weights can be trained by the data, or they can be assigned by experienced domain experts. For each type of user behavior, we have a user profile $\boldsymbol{U}^{b_p}$, and assign the weight $\boldsymbol{W}^{b_p}$. We introduce one possible weight learning strategy through logistic regression. The probability of whether the user i will apply the property j is given by

$$p_{ij} = p(y_{ij} = 1) = \sigma[\boldsymbol{\theta}(\boldsymbol{W}_i^{b_p} \boldsymbol{U}_i^{b_p} + \boldsymbol{V}_j) + \boldsymbol{b}] \tag{3}$$

where $\boldsymbol{\theta}$ and $\boldsymbol{b}$ are parameters of the logistic regression.

Item profiles and user profiles are constructed in the same feature space and ready for distance measurement. There are many suitable choices for performing these calculations, such as cosine similarity, Pearson's similarity, Euclidean measurement, or the RBF measurement. The choice depends on the situation and the characteristics of the domain. For example, cosine similarity is very popular and effective for word count and text similarity measurements due to the advantages of using angles rather than distance. Pearson's measurement tends to be more effective in memory-based collaborative filtering methods owing to its emphasis on averages. In this problem,

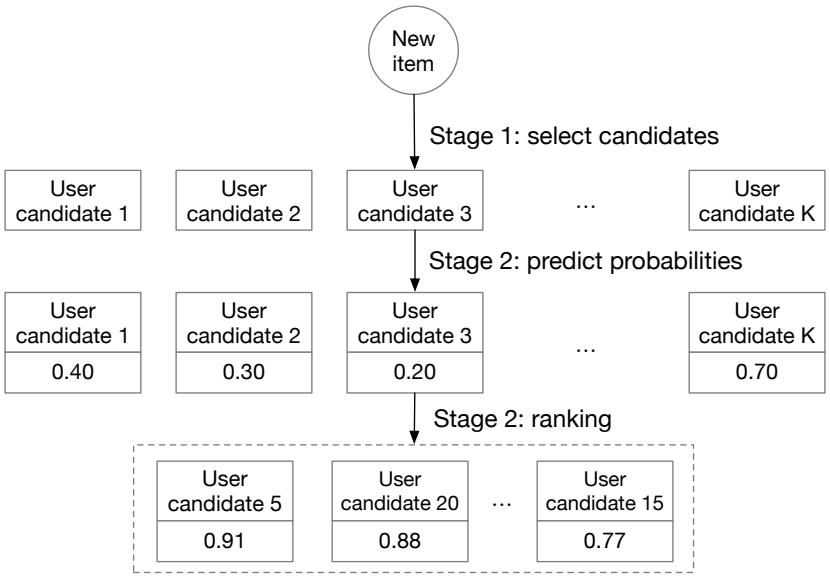

Fig. 1. Two stage recommendation method.

we are measuring user-item similarity from two vectors containing weights on each features, which is very similar to the situation of measuring the weight between two documents with weights on words. Therefore, in our developed method, cosine similarity is used to measure the distance between user i and item j as follows:

$$W_{(i,j)}^{cos} = \frac{\sum_{l=1}^{L} \boldsymbol{U}_{i,l} \times \boldsymbol{V}_{j,l}}{\sqrt{\sum_{l=1}^{L} \boldsymbol{U}_{i,l}^2} \times \sqrt{\sum_{l=1}^{L} \boldsymbol{V}_{j,l}^2}} \tag{4}$$

where $\boldsymbol{U}_{i,l}$ is the lth dimension in the user profile of user i, $\boldsymbol{V}_{j,l}$ is the lth dimension in the profile of item j, L is the total number of dimension in for both the user profile and the item profile.

After calculating the cosine similarities, the users are ranked by cosine similarity scores. Top-K users will be recommended to send notification for the cold-start items.

2) Boosting tree-based spam filtering: Pushing notifications to users can be risky since users will get tired and bored if the recommendation is not accurate or if users receive too many notifications. The second stage of the method is to compute the probability that user will interact with this item for a given user-item pair (i, j). In order to estimate the probability, we use XGBoost[1] to implement boosting tree(also known as GBDT, GBM) and rank the candidates chosen by our method in stage 1 to filter the possible spam notifications.

For training the model, we take users and items from training set as positive examples. Meanwhile, we sampled negative user-item pairs. Specifically, to be consistent to the cold-start scenario on items, we took an item-oriented random sampling strategy for each item from the entire user set (exclude users that already interacted with the item in the training set). Random negative sampling is appropriate since both user and item sets are large enough to ensure the randomness of the selection.

The top-K candidates from stage 1 content-based recommendation are ranked in stage 2. The candidate-users that have lower probability to interact with the items are filtered out. In this way, possible spam notification is limited.

[1]https://github.com/dmlc/xgboost

IV. Experiments and Analysis

In this section, the evaluation metrics will first be introduced, followed by dataset description on different setting. Then how to do feature engineering and feature selection of the data are introduced. Finally the experimental results on these dataset and comparison results are presented.

A. Evaluation Metrics

We use three evaluation metrics: precision, recall and MAP (Mean Average Precision) to evaluate our proposed method. They are defined as follows:

$$Precison@K = \frac{\#\ of\ recommended\ items\ @K\ that\ are\ relevant}{\#\ of\ recommended\ items\ @K} \tag{5}$$

$$Recall@K = \frac{\#\ of\ recommended\ items\ @K\ that\ are\ relevant}{total\#\ of\ relevant\ items} \tag{6}$$

$$AP@K = \frac{1}{|itemset|} \sum_{k=1}^{K} (P(j)\ if\ item\ j\ is\ relevant) \tag{7}$$

MAP is the average of AP on all the users.

B. Dataset Description

Our experiments are conducted on real-estate data in Australia. The attributes of properties vary in different areas. For example, the price of listings from Sydney and Melbourne can be quite different. And the preferences of customers who are renting a house are quite different from those who are buying a house. We divide the data according to geographic information

and the property market purpose information (sale/rent/invest). Our experiments are mainly based on the sale data in Sydney area. We suggest that recommendation engines should be built separately in different areas and on different market purpose.

We made subsets of the dataset on several different settings. For preprocessing, we filter out the items that have less than two interactions with users. This is because we are going to randomly split the dataset into training set and test set, partition as 80% training set and 20% test set with item-oriented method. If the item is checked less than two times, it will be either in training set or test set. Thus, the item will not be tested or cannot be recommended which makes it meaningless to be contained in the dataset. Data1 in Table II contains one week data while data2 contains two weeks data. These two subsets are made for testing the stage 1 of our proposed method. We accumulate three months data to generate some user attributes for stage 2 of our proposed method. And we test the performance of stage 2 on data3.

TABLE II
DATA DESCRIPTION OF DIFFERENT SETTINGS.

	User No.	Item No.	Train No.	Test No.
Data1	88818	50972	1024823	280587
Data2	77053	31902	946678	237456
Data3	76583	24907	935748	226782

The purpose of choosing different data settings are as follows: First, feature selection and feature engineering are processed to extract related attributes that are important to make recommendations. This work is mainly focused on data1 and also verified on data2. Second, the length of the time period of the dataset is another interesting factor we need to take into consideration. In this paper, we presented our experiment results on one week and two weeks data. We also tried four weeks data, but it did not make a difference. Hence, here we just present the results on two data settings.

We designed three scenarios on data2 to test our proposed method. Scenario 1 is randomly splitting the data to 80% and 20% to training set and test set. Scenario 2 is to split the training set and test set according to the time window so that it is close to real situation. Scenario 3 is to predict the new item which has never seen by any user in the training set. This scenario is to simulate the cold-start item scenario when it come freshly into the market.

C. Feature Selection and Feature Engineering

We have tried different combination of features and found that geographical feature and price are especially important. We fix our features to the following: property type, property area, property region, property suburb, number of bedroom, number of bathroom, number of car space, property price, train station information.

For all these features, we use one-hot encoding to align them to a vector where values are in $\{0, 1\}$. For the property price, we use two methods to encode it in one-hot vectors. One is to uniformly divide it according to its distribution. The other is to divide the price in the following range: $\{10K, 100K, 200K, 300K, 400K, 500K, 600K, 700K, 800K, 900K, 1M, 2M, 3M, 4M, 5M\}$. There is no big difference on the two methods above in our experiments. So we choose to use the second strategy in our proposed method. The price of the property which is missing is completed with the average price in that suburb where the property is in. The price of the property which is below $10K$ and above $5M$ are set to be $10K$ and $5M$ because price that high or that low is not unconvincing. The train station information is collected from NSW train station open data set. This feature is created as whether the property is within 1.5 km to the train station or not and whether the train station has express train or not.

Attributes used in stage 2 of our method is different from those in stage 1. Both user and item attributes are also in one-hot encoding and we tried different combinations of user and item attributes. Our method in the end contains the user attributes as follows: persona, engagement level, value segmentation, average price, median bathroom, median bedroom, median car space, favor property type. These features are accumulated in three months data. The first three user attributes are set according to some business rules.

D. Baselines

The baseline used in this paper is an existing method in the real-estate company. Concretely, in this baseline user profiles are firstly built according to their historical click records. The attributes they are using are nearly the same to our choice. They use the median number for the numerical attributes for bedroom number, bathroom number, car space number and price. And they calculate the choose the suburb and property type that appeared the most in users' click records as their profile. The query-based method that uses the profile built as above to match to the attributes of properties. The query rules are as follows:

1) The attributes of the property on bedroom number, bathroom number and car space number should be around the user profile within a range of less than one;
2) The property suburb should be in the same suburb of the user profile;
3) The property type should be in the same property type of the user profile;
4) The property price should be around the price in the user profile within a range of less than 15K.

E. Comparison Results and Analysis

To compare the performance of our proposed method with the existing query-based recommendation method, we evaluate them with the same evaluation metrics on the same dataset. The results on data1 are in Table III. We compare these two methods on two different scenarios on data2 in Table IV.

We have tested our proposed method on the three scenarios that described in the last section in Table V. The results suggest that our proposed method can recommend the listings to users even for cold-start listings (scenario 3) while the query-based method failed on the recommendation.

TABLE III
COMPARISON RESULTS WITH QUERY BASED METHOD ON DATA1

Method	K	Precision	Recall	MAP
Proposed method	5	0.0165	0.0296	0.0172
	10	0.0090	0.0321	0.0176
	50	0.0018	0.0321	0.0177
Query-based	5	0.0103	0.0098	0.0065
	10	0.0068	0.0120	0.0070
	50	0.0019	0.0146	0.0072

TABLE IV
COMPARISON RESULTS WITH QUERY BASED METHOD ON DATA2

Scenario	Method	K	Precision	Recall	MAP
Scenario 1	Proposed method	5	0.0275	0.0211	0.0093
		10	0.0250	0.0398	0.0121
		50	0.0066	0.0592	0.0138
	Query-based	5	0.0003	0.0002	0.0001
		10	0.0002	0.0003	0.0001
		50	0.0001	0.0004	0.0001
Scenario 2	Proposed method	5	0.0375	0.0082	0.0040
		10	0.0381	0.0159	0.0053
		50	0.0219	0.0471	0.0080
	Query-based	5	0.0228	0.0087	0.0055
		10	0.0161	0.0117	0.0061
		50	0.0053	0.0175	0.0067

Last but not least, the stage 2 of our method is tested on Data3. With the stage 2 spam filter, the precision and recall can be improved as shown in Table VI, indicating that our stage 2 model can help filter the possible Spam when pushing notifications to users.

We have tested multiple user behaviors that happened when users have interactions with properties. For previous experiments, the user behavior is their click on the property. We also considered using multiple user behaviors in the experiments. The other high level user behaviors are in three types: floorplan view (checking the details of the property), shortlist (adding the property to user favorite) and enquiry (including message and email enquiry). We tried different weights on the higher level behavior, but the performance of the proposed method was not improved quite a lot, as shown in Table VII.

TABLE V
RECOMMENDATION RESULTS OF THE PROPOSED METHOD IN THREE SCENARIOS.

Scenario	K	Precision	Recall	MAP
Scenario 1	5	0.0103	0.0141	0.0069
	10	0.0109	0.0286	0.0088
	50	0.0078	0.0995	0.0124
Scenario 2	5	0.0110	0.0037	0.0016
	10	0.0116	0.0081	0.0022
	50	0.0098	0.0345	0.0037
Scenario 3	5	0.0229	0.0052	0.0021
	10	0.0252	0.0112	0.0030
	50	0.0275	0.0576	0.0064

TABLE VI
COMPARISON RESULT OF STAGE 1 AND STAGE 2 OF THE PROPOSED METHOD.

Methods	K	Precision	Recall	MAP
Stage1	5	0.0154	0.0146	0.0067
	10	0.0155	0.0276	0.0085
	50	0.0139	0.0971	0.0124
Stage1+Stage2	5	0.0171	0.0154	0.0072
	10	0.0174	0.0304	0.0094
	50	0.0156	0.0906	0.0130

TABLE VII
EXPERIMENT RESULTS ON MULTIPLE USER BEHAVIORS.

Features	K	Precision	Recall	MAP
2 weeks Click	5	0.0113	0.0042	0.0021
	10	0.0118	0.0084	0.0027
	50	0.0098	0.0347	0.0042
2 weeks Multiple	5	0.0119	0.0045	0.0021
	10	0.0123	0.0090	0.0027
	50	0.0097	0.0336	0.0041
4 weeks Click	5	0.0126	0.0047	0.0024
	10	0.0134	0.0096	0.0030
	50	0.0108	0.0377	0.0047
4 weeks Multiple	5	0.0135	0.0050	0.0025
	10	0.0137	0.0098	0.0032
	50	0.0107	0.0372	0.0048

V. CONCLUSION AND FUTURE WORK

This paper presents a content-based recommender system to solve the cold-start item problem. The user and item contents are used in our proposed method. We use one-hot encoding to process the features after that we calculate similarities between users and items. Different user behaviors are considered in our proposed method and how they affect the performance of the proposed method is analyzed. The stage 1 of our proposed method is able to generate the recommendation to users. Meanwhile, we add a boosting tree model as stage 2 to effectively reduce the potential spam when we push notifications to users according to our recommendation results. Our experiment results show that the proposed method can better perform the baseline and properly handle the cold-start scenario as well.

Our future work can be done in the feature engineering part. Some important features are still missing but crucial for the real-estate market. For example the life style information about the property, such as how far the property is away from school zone or shopping mall can be important factors that influence the decision of users on buying a house. Also, temporal information should be further explored. User's preferences keep changing every now and then. The next property that users are interested may be highly related to the very last

property he/she has viewed but not correlated with the one he has viewed more than 2 weeks ago. The temporal dynamics in the data need further investigation.

ACKNOWLEDGMENT

This work was supported by the Australian Research Council (ARC) under Discovery Grant [DP170101632]. It is also supported by UTS Research Project RES18/1417.

REFERENCES

[1] J. Lu, D. Wu, M. Mao, W. Wang, and G. Zhang, "Recommender system application developments: a survey," *Decision Support Systems*, vol. 74, pp. 12–32, 2015.

[2] B. Yang, Y. Lei, J. Liu, and W. Li, "Social collaborative filtering by trust," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 8, pp. 1633–1647, 2017.

[3] M. Mao, J. Lu, G. Zhang, and J. Zhang, "Multirelational social recommendations via multigraph ranking," *IEEE Transactions on Cybernetics*, vol. 47, no. 12, pp. 4049–4061, 2016.

[4] Q. Shambour and J. Lu, "An effective recommender system by unifying user and item trust information for b2b applications," *Journal of Computer and System Sciences*, vol. 81, no. 7, pp. 1110–1126, 2015.

[5] W. Wang, G. Zhang, and J. Lu, "Member contribution-based group recommender system," *Decision Support Systems*, vol. 87, pp. 80–93, 2016.

[6] X. Luo, M. Zhou, S. Li, Z. You, Y. Xia, and Q. Zhu, "A nonnegative latent factor model for large-scale sparse matrices in recommender systems via alternating direction method," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 27, no. 3, pp. 579–592, 2016.

[7] Q. Zhang, J. Lu, D. Wu, and G. Zhang, "A cross-domain recommender system with kernel-induced knowledge transfer for overlapping entities," *IEEE Transactions on Neural Networks and Learning Systems*, 2018.

[8] S. Boutemedjet and D. Ziou, "Predictive approach for user long-term needs in content-based image suggestion," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 23, no. 8, pp. 1242–1253, 2012.

[9] Y. Gu, B. Zhao, D. Hardtke, and Y. Sun, "Learning global term weights for content-based recommender systems," in *Proceedings of the 25th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 2016, pp. 391–400.

[10] B. Shapira, F. Ricci, P. B. Kantor, and L. Rokach, "Recommender systems handbook," 2011.

[11] J. Bobadilla, F. Ortega, A. Hernando, and A. Gutiérrez, "Recommender systems survey," *Knowledge-Based Systems*, vol. 46, pp. 109–132, 2013.

[12] J. B. Schafer, J. Konstan, and J. Riedl, "Recommender systems in e-commerce," in *Proceedings of the 1st ACM Conference on Electronic Commerce*. ACM, 1999, pp. 158–166.

[13] G. Adomavicius and A. Tuzhilin, "Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 6, pp. 734–749, 2005.

[14] R. Burke, "Hybrid recommender systems: survey and experiments," *User Modeling and User-adapted Interaction*, vol. 12, no. 4, pp. 331–370, 2002.

[15] U. Shardanand and P. Maes, "Social information filtering: algorithms for automating "word of mouth"," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM Press/Addison-Wesley Publishing Co., 1995, pp. 210–217.

[16] M. J. Pazzani and D. Billsus, "Content-based recommendation systems," in *The adaptive web*. Springer, 2007, pp. 325–341.

[17] P. Lops, M. De Gemmis, and G. Semeraro, "Content-based recommender systems: state of the art and trends," in *Recommender systems handbook*. Springer, 2011, pp. 73–105.

[18] M. Balabanović and Y. Shoham, "Fab: content-based, collaborative recommendation," *Communications of the ACM*, vol. 40, no. 3, pp. 66–72, 1997.