Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 35 (2014) 1063 – 1072

Procedia Computer Science

18th International Conference on Knowledge-Based and Intelligent Information & Engineering Systems - KES2014

A personalized recommender system from probabilistic relational model and users' preferences

Rajani Chulyadyo[a,b,*], Philippe Leray[a]

[a]LINA (UMR CNRS 6241), DUKe Research Team, Ecole Polytechnique de l'Université de Nantes, Nantes 44300, France
[b]DataForPeople, Nantes 44300, France

Abstract

Recommender systems are applications to retrieve useful information from large amount of online data to assist users in discovering interesting items/products in the system. Collaborative filtering, content-based filtering, demographics-based filtering and hybrid approach are main approaches to realize recommendation systems. Most of the existing algorithms use a single approach to deal with recommendation problems. Besides, traditional recommendation approaches mainly deal with single dyadic relationships between users and items whereas data in real world are generally conceptualized in terms of objects and relations between them. Recommender systems based on Probabilistic Relational Model (PRM)[1,2], a framework for learning probabilistic models from relational data, have tried to address this issue. However, existing PRM-based recommendation algorithms do not fit into our context where we are struggling with the contradictory situation of a real-world application that requires building a personalized recommender when no user profile exists. Therefore, we propose a novel approach to build a personalized PRM-based recommendation model with the help of users' preferences on decision making criteria. Using our approach, content-based, collaborative filtering as well as hybrid models can be achieved from the same PRM. Applying the model on a real-world data from a cold system, we show that our model is actually capable of personalizing recommendations in coldstart situation.

Keywords: Personalized Recommender Systems; Probabilistic Relational Model (PRM); Users' Preferences; Cold start problem; Hybrid recommender system

1. Introduction

With the advancement of information technologies, huge amount of online information are being available day by day. This abundance of data has increased interest in data analysis and knowledge discovery. Need for systems to extract knowledge or filter useful and interesting information from the overwhelming collection of information has led to the development of different information retrieval (IR) technologies. Moreover, users' increased expectation has made it difficult to please them by searching based on few keywords or some criteria only as it may not capture their

* Corresponding author.
E-mail address: rajani.chulyadyo@univ-nantes.fr

actual interests. To fulfill the demand of intelligent data analysis, many tools and applications have been invented and recommender system is one of them. Recommender system is an intelligent IR tool that helps users discover items or products of their interest from a large collection of items. We can encounter recommender systems not only in e-commerce applications but also in many everyday websites such as news websites, social networking sites, entertainment websites, and so on.

Several recommendation algorithms have been successfully implemented in different domains. These algorithms mainly follow collaborative filtering, content-based filtering, demographics-based filtering and hybrid approaches. Collaborative filtering approach[3] relies on users' past behaviour to make recommendations. This is clearly not applicable in new systems where the users' interaction with the system is not enough to capture their behaviour. Content-based[4] and demographics-based filtering approaches[5] have tried to address this situation, which is often known as *cold start problem*. These methods use information about items and users in order to suggest the items to the users. Hybrid approaches combine these techniques to benefit from their capacity in an attempt to avoid the limitations of either approach.

Most of the existing algorithms for recommendation are based on similarity between users (or items)[6]. Algorithms designed to address cold start problem generally use users' demographics to compute similarity between users. However, the fact that similar users can have different preferences over attributes of items does not seem to be addressed in these approaches. For example, jobseekers with similar degrees and experience may have different preferences for different attributes of jobs. Some may have strong preference about category of job they want whereas other may focus primarily on salary. As another example, let's consider users looking for an accommodation. Two users with similar demographic information and similar search criteria might have completely different preferences about budget, surface area, and location etc. of the accommodation. These preferences might vary even for the same person depending on his current income, family status or other hidden or visible factors. Besides, traditional algorithms rely on flat data representation such as user-item matrix to predict relation between target users and items[3,7]. However, relational database is still prevalent in many real-world applications where the systems are conceptualized in terms of objects and relations between those objects. To apply traditional recommender algorithms on such applications, the relational data need to be converted to flat representation. But, during this conversion, statistical skew is often introduced and useful information that might help the understanding of data is lost. Moreover, in the domains where recommender systems are often used (for example e-commerce), users and items grow continuously. Recommender systems need to be scalable to keep up with the growing size of the dataset without degrading performance.

Basically, recommendation systems predict relations between entities (users and items). So, Probabilistic Relational Model (PRM)[8], a framework for describing statistical models over relational domains, can be used to model them. Using PRM in recommender systems has been a field of research from the beginning of its formalism. Getoor and Sahami[9] proposed a recommender using regular PRMs whereas Huang et al.[10] used the concept of *PRMs with existential uncertainty* and built Naive Bayesian classifier from relational attributes to make recommendations. Newton and Greiner[11] extended PRM and proposed Hierarchical Probabilistic Relational Model (hPRM) which could produce recommendation based on hierarchy of items. However, none of them addresses cold start problem. Ben Ishak et al.[12] and Gao et al.[13] proposed PRM-based hybrid recommenders. Ben Ishak et al.[12] follow a pure PRM approach whereas Gao et al.[13] combine traditional collaborative filtering with regular PRM. In their UGCF-PRM approach, Gao et al.[13] use PRMs to handle cold start problem only and, therefore, do not actually exploit full capability of PRM in recommendation[14]. The motivation of our work is the need of an intelligent recommender in an industrial context where the system is in cold start situation and also lacks user profile. In this system, users specify their search criteria to filter relevant items. We aim at aiding the users by presenting their search results sorted by relevance. We are struggling with the contradictory situation of a real-world application that requires building a personalized recommender when no user profile exists. Existing methods are not very useful in this scenario. Thus, we propose a novel approach of constructing a personalized recommendation model from PRMs with the help of users' preferences over their search criteria. With this approach, recommendation can be performed from the same PRM in both cold and hot systems.

2. Underlying frameworks

Probabilistic relational model (PRM)[1,2] is a formal approach to relational learning and specifies probability model for classes of objects. PRM is a relational extension of Bayesian Networks[15] and models the uncertainty over the

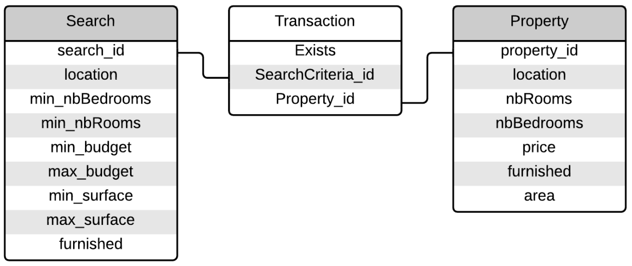

Fig. 1. Relational schema

attributes of objects in the domain and the uncertainty over the relations between the objects. It basically defines a template for probabilistic dependencies in typed relational domains which can be later instantiated with a particular set of objects and relations between them. A PRM is comprised of two components: a *relational schema* of the domain, and a *probabilistic graphical model* describing the probabilistic dependencies between the attributes of the classes in the domain. In this paper, we follow the notation used by Getoor [2].

Relational schema. Relational schema describes a set of classes $\mathcal{X}$ and the relation between them. Each class $X \in \mathcal{X}$ is described by a set of descriptive attributes $\mathcal{A}(X)$ and a set of *reference slots* $\mathcal{R}(X)$, i.e., attribute(s) that allows objects to refer to another objects. In the context of a relational database, a class refers to a single database table, descriptive attributes refer to the standard attribute of tables, for example age, gender, education of users, type/price of products etc. and reference slots are equivalent to foreign keys. An attribute $X.A$ of a class $X \in \mathcal{X}$ takes values on a range $\mathcal{V}(X.A)$. A reference slot of a class X that relates an object of class X to an object of class Y is denoted as $X.\rho$ where Domain$[\rho] = X$ and Range$[\rho] = Y$. For each reference slot ρ, we can define an inverse function of ρ, which we call an *inverse slot* ρ^{-1}. A reference slot gives a direct reference of an object with another whereas a series of reference slots and inverse slots gives an indirect reference of objects with another. Such references are expressed by means of *slot chain*s, a set of reference slots or inverse slots with the range of a slot being the domain of the next slot. Formally, a slot chain is a set of slots $\rho_1, \rho_2, \ldots \rho_n$ such that for all i, Range$[\rho_i]$ = Domain$[\rho_{i+1}]$. Length of a slot chain is the number of slots in the slot chain.

Figure 1 shows an example of a relational schema where Search and Property are entity classes and Transaction is a relationship class that represents the relation between these two classes. The attributes of Search are Search.*location*, Search.*min_nbBedrooms* etc. Transaction is related to Search through the reference slot Transaction.Search and to Property through the reference slot Transaction.Property. Thus, the slot Transaction.Search gives the Search object associated with the current Transaction object. A series of slots Transaction.Search, Transaction.Search^{-1}, and Transaction.Property can relate objects of these three classes in a meaningful way. The slot chain formed by this series, written as Transaction.Search.Search^{-1}.Property, gives all the Property objects associated with the same Search object as the current Transaction object. Length of this slot chain is 3.

Probabilistic model. A relational schema is instantiated with a set of objects for each class, the values of the attributes and the relations between the objects. The probability distribution of such an instance $\mathcal{I}$ of a relational schema is specified by a probabilistic model in a PRM. The model consists of a dependency structure and the parameters associated with it. Dependency structure is a result of parent-child relations of attributes just like in Bayesian networks. The dependencies can be between the attributes of a class or between the attributes of different classes. The associated parameters are conditional probability distributions of each node given its parents.

Another important concept in PRM is the notion of *Aggregator*. Aggregators come into play when there is dependency between the objects that have one-to-many or many-to-many relations. An aggregator, denoted γ, is a function which takes a multi-set of values and produces a single value as a summary of the input values. Average, mode, cardinality etc. can be used as an aggregation function.

Formally, a regular Probabilistic Relational Model (PRM) Π for a relational schema $\mathcal{R}$ is defined as follows [2]. For each class $X \in \mathcal{X}$ and each descriptive attribute $A \in \mathcal{A}(X)$, we have:

- a set of parents $\text{Pa}(X.A) = \{U_1, \ldots U_l\}$, where each U_i has the form $X.B$ or $\gamma(X.\tau.B)$, where B is an attribute on which A depends, τ is a slot chain and γ is an aggregator of $X.\tau.B$.
- a conditional probability distribution (CPD), $P(X.A|Pa(X.A))$.

Probabilistic inference is performed on a *Ground Bayesian Network* (GBN) obtained from a PRM for the given instance $\mathcal{I}$. A GBN is generated by a process (also called unrolling) of copying the associated PRM for every object in $\mathcal{I}$. Thus a GBN will have a node for every attribute of every object in $\mathcal{I}$ and probabilistic dependencies and CPDs as in the PRM. Standard inference algorithms for Bayesian network can be used to query the GBN.

A *PRM with existence uncertainty*[8,2] is an extension of a regular PRM and provides probabilistic models for the existence of relations between objects too. This extension models the uncertainty of existence of relationship between objects by adding a binary existence attribute *X.exists* to the relationship class. This probabilistic attribute gives whether the relationship object actually exists or not. Huang et al. [10] have shown that PRM with existence uncertainty can be interesting for recommendation applications.

3. Proposed model

We propose a recommender system, based on PRM with existence uncertainty, which can personalize recommendation from users' preferences. Sections 3.1 and 3.2 present the relational schema, and the probabilistic structure of our model respectively. Section 3.3 deals with the ways to assign parameters to get a complete model. Section 3.4 proposes some methods to rank decision factors. Using PRM to obtain recommender is discussed in 3.5. Types of achievable models are presented in section 3.6.

3.1. Relational schema

The model does not use traditional user-item matrix but relational database. The input relational schema is supposed to have three main classes – items, search (details) and relation between them. Figure 1 shows the relational schema from a real estate search system where the main classes are property, search (session) and the properties viewed by the users in the given search session. The relation between Property and Search, namely 'Transaction', is a many-to-many relation between them. Thus, the relation from Search to Transaction is one-to-many and that from Transaction and Property is many-to-one. A binary attribute 'exists' is added to the Transaction class. It takes value '1' if there exists a Transaction between the given pair of Property and Search instances, i.e., if the property is visited by the user in the given search session, otherwise '0'. Our approach is slightly different than traditional one as we are using instances of Search instead of User. The schema can be extended by adding more entities related to these main entities. Given the relational schema and users' preferences for the search criteria (explained in the following sections), the task for the model is to predict the probability of existence of relation between item and search session, i.e., to predict the value of the relational attribute Transaction.*exists*. In other words, it predicts how likely the user would visit the property for the given search criteria.

3.2. Probabilistic structure

The target attribute Transaction.*exists* actually depends on many attributes from Property and Search. This makes it difficult to know the distribution of the target attribute given very specific parent configurations because the distribution table can be very big due to large number of parents. Thus, to simplify this as well as to capture users' preferences for every search criteria, we *divorce*[16] the parents or introduce intermediate variables so that the target attribute will depend on small number of attributes. These intermediate variables, referred to as *Decision factors* (or simply *DF*s), will belong to the Transaction class and are chosen in such a way that each decision factor represents a search criterion. The PRMs before and after parent divorcing are shown in figure 2. It should be noted that decision factors can have more than two parents in complex cases and these parents can be further divorced to simplify the conditional distribution table.

Our idea is to pick one or more Search attributes that represent a search criterion, and their corresponding attributes from Property in order to form a decision factor. The selection of the attributes for decision factors is, in fact, a

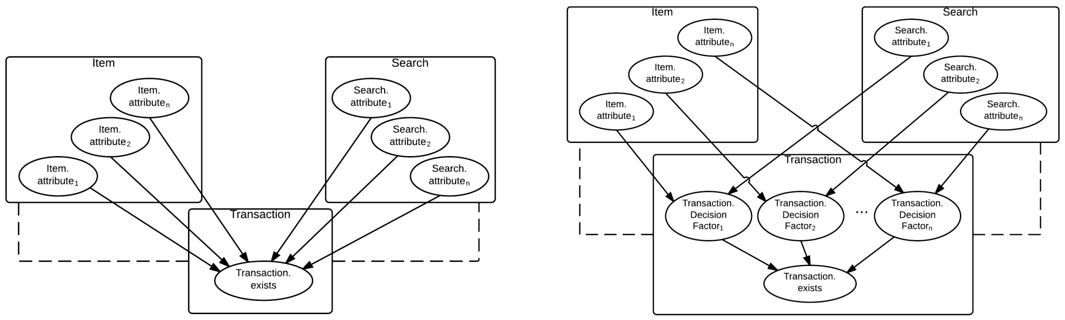

Fig. 2. PRM of our system (a) before introducing decision factors; (b) after introducing decision factors.

domain-specific problem. Thus, we need experts' advice to define parents of decision factors. In our model, we have a search criterion called 'Minimum number of rooms', represented by Search.*min_nbRooms*, which gives the minimum number of rooms the user is looking for. The decision factor that corresponds to this search criterion has Search.*min_nbRooms* and Property.*nbRooms* as its parents. Similarly, the search criterion 'Budget', which gives the range of amount he can afford, is represented by a decision factor whose parents are Search.*minimum_budget*, Search.*maximum_budget* and Property.*price*. Decision factors, in fact, give a measure of how close or far the item attribute is from the criterion expressed by the users. They serve a way to impact the final decision by tuning the level of impact according to the users' importance on the search criteria. This will be further discussed in the following section.

3.3. Parameters

Once the structure of the model is defined, we need to assign parameters to the model. Parameters that are related to items only are learned from data whereas those related to search are assigned a uniform distribution because search attributes are always observed in the final recommendation model (explained in section 3.5. Parameters related to decision factors $P(\text{Tx}.DF \mid Parents(\text{Tx}.DF))$ are estimated by experts. Here, DF stands for decision factor. Table 1 shows an example of the conditional probability distribution of an intermediate variable whose parents are Property.*furnished* and Search.*furnished*. Finally, the last table corresponding to the recommendation process with the help of decision factors, i.e., $P(\text{Tx}.exists \mid \text{Tx}.DF_1, \dots \text{Tx}.DF_n)$, takes into account users' preferences expressed during search.

Table 1. CPD of $P(\text{Tx}.DF\_furnished \mid \text{Search}.furnished, \text{Property}.furnished)$

Tx.*furnished*	Yes		No	
Search.*furnished*	Yes	No	Yes	No
Positive	0.892	0.402	0.598	0.99
Negative	0.108	0.598	0.402	0.01

Conditional distribution table of the target variable, i.e., $P(\text{Tx}.exists \mid \text{Tx}.DF_1, \dots \text{Tx}.DF_n)$, can have a large number of parameters even after parent divorcing. Thus, we propose to use some approximation methods to build this table. Noisy-OR[15], and Leaky Noisy-OR[17] are well-known approximation algorithms which require separate influence of each DF. Experts can help identify the importance of each DF and specify their influence on the target variable. However, this approach would not consider users' preferences over decision factors making the system insensitive to users' interests. For this reason, we view the problem of constructing conditional distribution table of the target variable as a Multi-criteria Decision Making (MCDM)[18] problem. From the users' preferences over decision factors, we find weights W_i for each decision factor DF_i and apply some heuristics, e.g., Weighted Sum Model (WSM)[19], Noisy-OR[15], or Leaky Noisy-OR[17], to generate conditional probability table for the target variable. Because this table is user-specific, it is computed during instantiation of the PRM. In the next section, we propose some ways to rank DFs to obtain their weights.

3.4. Methods for ranking decision factors

As we do not have any information about users in our database (cf. hypotheses explaining our model), we aim at personalizing the recommendation using their preferences over decision factors. We propose here three methods to compute weights W_i for each decision factor DF_i by ranking the decision factors according to users' preferences. Users are asked to provide their preferences over a subset of decision factors only to reduce the number of questions to be asked to the users. Low weights are assigned to the decision factors that are not presented to the users. Here, we use the terms 'decision factors' and 'search criteria' interchangeably because each decision factor is associated with a search criterion.

Method I. The users are asked to sort the decision factors according to their importance. Then predefined weights are assigned to decision factors such that the most important decision factor gets the largest weight and the least important one gets the lowest weight. For example, assign 1 to the least important decision factor and increase the weight by 1 for the next decision factor in the sorted list so that the weight of the most important decision factor will be equal to Number of decision factors + 1. The decision factors that are not presented to the users get the lowest weight (1 or less in the previous example).

Method II. For every pair of decision factors, the users are asked for relative importance of the DFs. Then, we apply Analytical Hierarchical Process (AHP)[20] to compute final ranks and weights of the DFs. Again, the decision factors, not presented to the users, are assigned weights lower than or equal to the lowest weight.

Method III. The first method does not capture users' view on relative importance of the search criteria, which can be a crucial input for better result. The second method captures this information but is not very practical because we need to ask for relative importance for every pair of search criteria and the number of such pairs grows combinatorially. Thus, we propose another approach in which we ask users to rank them in a scale such that the gap between two criteria can represent relative importance of the search criteria. From this information, we prepare the matrix required to perform AHP and follow the method II afterwards.

3.5. Recommendation model from PRM

In most of the existing PRM-based recommenders, PRM is simply unrolled and recommendation is performed on the obtained GBN. Unrolling PRM in figure 2 would result in a simple content-based model. The main problem of this model is that it will be largely affected by the distribution of unobserved attributes of Search and Property when they should not. Not observing a Search attribute means that this attribute should not have any effect on the final decision. Therefore, to overcome this, we propose another method to build a recommendation model. We first create relational attributes from PRM by traversing through reference slots. We keep in the model the observed Search attributes and their spouses (from Property) for their corresponding decision factors. Thus, the final model consists of the target attribute Transaction.*exists*, its parent decision factors whose parent Search attribute(s) is (are) observed, and the parent attributes of these decision factors from Search and Property classes. The attributes from Property may or may not be observed though because in real world, not all the attributes of the items can be expected to be observed. Inference will be performed on this model after assigning a conditional distribution table of Transaction.*exists* (explained in section 3.3).

3.6. Relational attributes and types of model

Depending on the length of slot chain of the relational attributes used in the model, the achievable models are categorized into three types. Figure 3 depicts the three types of models. Here, we will be referring Transaction class as Tx, reference slot from Transaction to Search as S and reference slot from Transaction to Property as P.

Type I. This is the simplest model where the length of the slot chain is 1. It is a typical scenario of cold start problem where the current user does not have search history at all and the system does not have enough data from existing users to predict based on their behaviour. The prediction from this model will be the result from experts' knowledge and users' preference of the search criteria. Hence, this model is purely a content-based model.

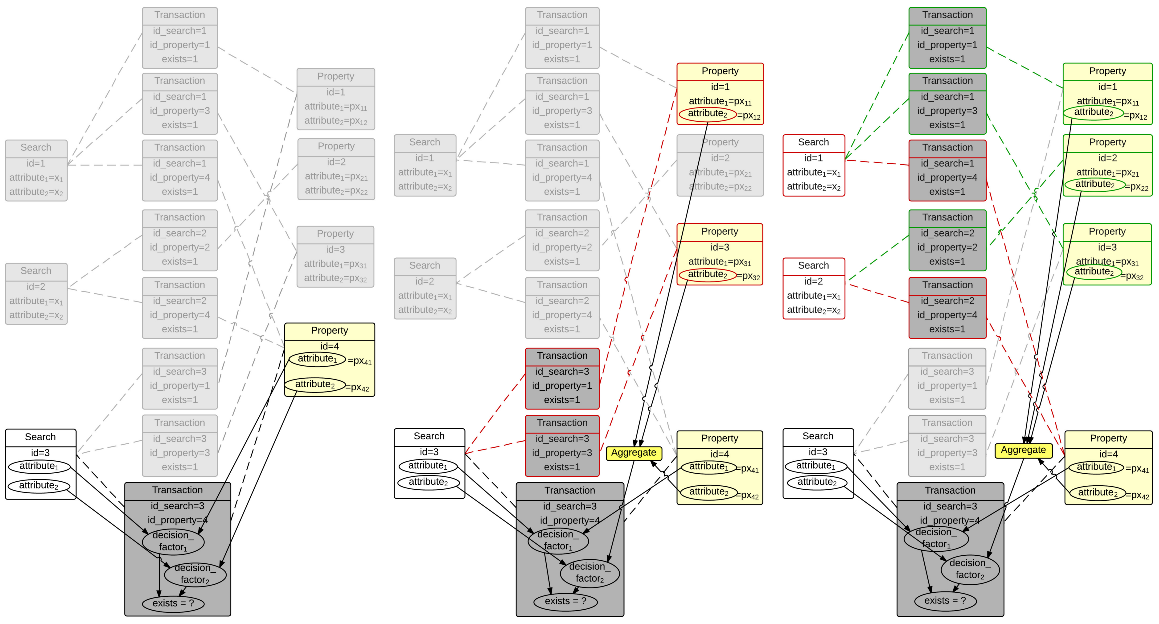

Fig. 3. The three types of models: (a) Type I model – Decision factors depend on attributes of slot chain of length 1; (b) Type II model – Decision factors depend on at least one attribute of slot chain of length 3. Here, the attribute Tx.*decision_factor*$_2$ depends on Aggregate(Tx.S.S^{-1}.P.*attribute*$_2$). The instances of Property in red border (all instances *p* of Property such that *p.id* ∈ {1, 3}) are the properties visited in the current session; (c) Type III model – It has at least one decision factor that depends on relational attributes of slot chain of length 5. Here, Tx.*decision_factor*$_2$ depends on Aggregate(Tx.P.P^{-1}.S.S^{-1}.P.*attribute*$_2$). The instances of Property in green border (all instances *p* of Property such that *p.id* ∈ {1, 2, 3}) are the properties visited in the previous search sessions when the target property (id = 4) is also visited.

Type II. It is an extension of Type I model with some relational attributes of slot chain of length 3. An aggregation function (e.g., mode, mean, cardinality) is required when the slot chains are not guaranteed to be single-valued. In figure 3(b), applying mode would result in a relational attribute MODE(Tx.S.S^{-1}.P.*attribute*) in the model. Clearly, Tx.S.S^{-1}.P gives a set of properties visited in the current search session. Thus, this model can capture the history of visiting items in the current search session. Even a new user can get recommendation from this model based on his few interactions in the current session.

Type III. It is an extension of Type II model with relational attributes obtained from longer slot chains. Such attributes can capture information from previous search sessions and, hence, be applied for new users to recommend existing items that have already appeared in other search results. For example, MODE(Tx.P.P^{-1}.S.S^{-1}.P.*attribute*) is a relational attribute of slot chain of length 5 . Tx.P.P^{-1}.S.S^{-1}.P is a multiset of Property objects that are visited in previous Search sessions when the target Property object is also visited. In fact, this model is equivalent to traditional collaborative filtering where users are recommended the existing items which already have some kind of interactions with the existing users. The grayed-out objects, in figure 3, are the ones not used in the model. Thus, as shown in the figure, with the increase in slot chain length, the model deals with more and more instances. Type III models or the models with very long slot chains actually may not be very suitable for cold systems.

4. Experimental results

We performed our preliminary experiment on a system which is in cold start situation with very few search sessions and a small number of transactions in each search session. In the following sections, we describe our dataset and explain how we performed the experiment and evaluated the model. Then we will present the results and discuss on the findings.

4.1. Dataset

The dataset used in this experiment was from Kyzia (www.kyzia.fr), a real estate search system developed by DataForPeople (www.dataforpeople.fr). It is a new system and represents cold-start problem well. Relational schema of this system is shown in figure 1. The system presents the users a list of Property objects matching their search criteria. We assume that the users visit the details page of a property if they find it interesting. This information is modeled by the class Transaction and were collected by logging the users' clickthroughs. The dataset has more than 70,000 real estate properties, around 1400 search sessions but less than 100 transactions. Ranking method II was used to collect users' preferences. Because the experiment was performed on an evolving system rather than a fixed dataset, we haven't provided the exact size of the dataset here. Users were asked to perform search on Kyzia website but since the feature of providing their preferences had not been implemented yet on the website, they were asked to provide their preferences separately to us. Only those search sessions with more than one transaction and whose search criteria preferences are known were kept for evaluating the models. After cleaning, only 7 search sessions were left and the average number of transactions per search session was 3 (2 being the minimum value).

4.2. Experiment methodology

A limited number of attributes from Search and Property classes was used in this experiment. Though there were many more attributes in both classes, only the attributes shown in figure 1 were taken into consideration in order to prove our concept. With the help of experts, all decision factors were identified for our PRM. If there is enough data, parameters for each attribute can be learned from data. However, available data was not enough to learn parameters of all attributes. Only parameters of Property attributes were learned from data. Parameters were assigned to the decision factors with the help of experts. Search attributes were assigned a uniform distribution because they are always observed in the final model. The Transaction objects were not used during model construction but were kept for evaluation of the models.

For every search session initiated by a user, models were built based on his search criteria and his preferences over them. Type I and Type II models were built from the PRM by keeping the target attribute Transaction.*exists*, the observed Search attributes, and the children (decision factors) and the spouses (i.e., $\mathsf{Tx}.K.\mathsf{P}.B$ where B is an attribute of Property) of the observed Search attributes. Type III models were not tested in this experiment because good amount of transaction history is required to create this type of models. While constructing Type I and Type II models, two heuristics, Noisy-OR, and Weighted Sum Method (WSM), were applied to compute the conditional distribution of the Transaction.*exists*. Thus, the experiment involves two parts – comparison of heuristics used in the model, and comparison of Type I and Type II models.

4.3. Evaluation metrics

With the amount of available data, it is not possible to perform extensive offline evaluation[21]. Evaluation was performed using the available Transaction objects in two phases. In each phase, 4 models were created for each search session in the evaluation dataset as explained in section 4.2. Each model was then applied over the properties that were actually shown up to the users during the particular search session, and the properties were ranked based on the probability of existence of transaction given the decision factors, $P(\mathsf{Tx}.exists \mid DF_1, DF_2, \ldots, DF_n)$. In the first phase, only top-3 recommendations were considered during evaluation whereas in the second phase, top-5 recommendations were taken. Standard recommendation quality metrics such as precision, recall and F-measure were calculated by comparing the top-N properties in this ranked list with the Transaction objects for the search session.

For a search session $s \in \mathcal{I}(\mathsf{Search})$, a set of Transaction objects tx' in the recommendation list and a set of Transaction objects $tx = \{t : t \in \mathcal{I}(\mathsf{Transaction}); t.\mathsf{Search} = s\}$ in the evaluation dataset, the evaluation metrics for Type I model are calculated in the following way:

Table 2. Evaluation result

		Top-3			Top-5		
Model Type		Precision	Recall	F-measure	Precision	Recall	F-measure
Type I	Noisy-OR	0.38	0.38	0.18	0.42	0.68	0.24
Type I	WSM	0.38	0.38	0.18	0.39	0.60	0.22
Type II	Noisy-OR	0.46	0.59	0.24	0.40	0.76	0.25
Type II	WSM	0.46	0.59	0.24	0.45	0.76	0.26

$$\text{Precision} = \frac{\text{Cardinality}(tx \cap tx')}{N} \tag{1a}$$

$$\text{Recall} = \frac{\text{Cardinality}(tx \cap tx')}{\text{Cardinality}(tx)} \tag{1b}$$

$$\text{F-measure} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{1c}$$

For type II model where the previously visited properties in the current search session are also considered, temporal information from the clickthrough logs were employed to compute the evaluation metrics. For a given search session and a set of Transaction objects related to it, evaluation is performed n times where $n = \text{Cardinality}(tx)$. In each iteration i, only a subset of tx that are visited before the current is used in the model and metrics are computed using equations 1. Average of the metrics in all iterations gives the overall metrics for the particular search session.

4.4. Results and discussion

Metrics obtained from the experiment are presented in table 2. In the first part of the experiment, two heuristics, Noisy-OR, and WSM, were compared. Both of these methods produced similar results. With limited data, choice of approximation algorithms did not seem to have big impact the result. The second part involved the comparison of types of models. When N was changed from 3 to 5 in top-N evaluation approach, a slight increase in precision but significant increase in recall was observed. However, in both the cases, Type II models performed better than Type I models did. This signifies that the properties are better ranked in the type II models. Top-3 recommendation metrics for type II model were somewhat closer to top-5 recommendation metrics for type I model. Recall of type II model is quite good when top 5 models are taken into account for evaluation.

Offline evaluation in cold systems may not actually give the clear picture of the performance of the model because of insufficiency of test data. However, standard datasets for evaluating recommender systems, such as MovieLens (www.grouplens.org/datasets/movielens/) and Last.fm (www.dtic.upf.edu/~ocelma/MusicRecommendationDataset/) datasets, are not applicable in our context because they lack users' preferences over item attributes or decision factors. Comparing our model with another recommendation algorithms is also not possible because of lack of user model in our system and incapability of integrating users' preferences over decision factors in existing algorithms. Therefore, in this scenario, better approach to assess the model is to perform online evaluation where users interact with the system and take the next step based on the result they get from the model. Evaluating the model in such a way could also reveal how novel or serendipitous the results are. However, it was expensive to evaluate our system that way. Thus, we had to stay with the offline evaluation approach. It is also worth noticing that the quality of a model might have been affected by hidden causes that can affect users' decision. For instance, users might have made decision for some properties based on the quality of the pictures posted in the announcements, but such information have not been included in our model and are not in the scope of this work.

5. Conclusion

There has been significant progress in the field of recommendation systems in past few years. Though many algorithms have been devised to perform recommendations, only few have addressed cold start problem and even less performs relational learning. In this paper, we have presented a personalized model based on Probabilistic relational model (PRM) with existence uncertainty and capable of performing recommendations in cold as well as hot systems. The model was applied on a real-world data which lacked information about users. Due to this fact, we needed users to express their preferences over search criteria explicitly in order to get personalized recommendations. Using our approach, content-based, collaborative-filtering and hybrid models can be achieved from the same PRM by varying the length of slot chains. Our preliminary experiment on real-world dataset has shown that we were able to get good result even with small dataset using our approach. We are planning to implement this model in Kyzia with user-friendly interface to collect users' preferences over search criteria and to perform online evaluation of the model through user interactions. In future, we also expect to extend our model making it capable of handling bigger domains with lots of entities and to evaluate the model in terms of time required for model construction.

Acknowledgements

The authors would like to acknowledge Mr. Cédric Houssin, CEO of DataForPeople and Mr. Romain Perruchon, CTO of DataForPeople for their domain expertise required for the construction of the models.

References

1. Friedman, N., Getoor, L., Koller, D., Pfeffer, A.. Learning probabilistic relational models. In: *International Joint Conference on Artificial Intelligence*; vol. 16. Lawrence Erlbaum Associates Ltd; 1999, p. 1300–1309.
2. Getoor, L.. *Learning statistical models from relational data*. Ph.D. thesis; Stanford University; 2001.
3. Su, X., Khoshgoftaar, T.M.. A survey of collaborative filtering techniques. *Advances in Artificial Intelligence* 2009;**2009**:1–20.
4. Pazzani, M.J., Billsus, D.. Content-based recommendation systems. In: Brusilovsky, P., Kobsa, A., Nejdl, W., editors. *The Adaptive Web*; vol. 4321 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg. ISBN 978-3-540-72078-2; 2007, p. 325–341.
5. Ricci, F., Rokach, L., Shapira, B., Kantor, P.B.. *Recommender Systems Handbook*. Springer-Verlag New York, Inc.; 1st ed.; 2010. ISBN 0387858199, 9780387858197.
6. Adomavicius, G., Tuzhilin, A.. 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* 2005;**17**(6):734–749.
7. Koren, Y., Bell, R., Volinsky, C.. Matrix Factorization Techniques for Recommender Systems. *IEEE Computer* 2009;**42**(8):30–37.
8. Getoor, L., Friedman, N., Koller, D., Taskar, B.. Learning probabilistic models of relational structure. In: *Proceedings of the Eighteenth International Conference on Machine Learning*; ICML '01. ISBN 1-55860-778-1; 2001, p. 170–177.
9. Getoor, L., Sahami, M.. Using probabilistic relational models for collaborative filtering. In: *Workshop on Web Usage Analysis and User Profiling (WEBKDD'99)*. 1999, p. 1–6.
10. Huang, Z., Zeng, D., Chen, H.. A unified recommendation framework based on probabilistic relational models. In: *Fourteenth Annual Workshop on Information Technologies and Systems (WITS)*. 2004, p. 8–13.
11. Newton, J., Greiner, R.. Hierarchical probabilistic relational models for collaborative filtering. In: *Workshop on Statistical Relational Learning, 21st International Conference on Machine Learning*. 2004, .
12. Ben Ishak, M., Ben Amor, N., Leray, P.. A relational bayesian network-based recommender system architecture. In: *Proceedings of the 5th International Conference on Modeling, Simulation and Applied Optimization (ICMSAO 2013)*. 2013, .
13. Gao, Y., Qi, H., Liu, J., Liu, D.. A recommendation algorithm combining user grade-based collaborative filtering and probabilistic relational models. In: *Proceedings of the Fourth International Conference on Fuzzy Systems and Knowledge Discovery - Volume 01*; FSKD '07. Washington, DC, USA: IEEE Computer Society. ISBN 0-7695-2874-0; 2007, p. 67–71.
14. Chulyadyo, R., Leray, P.. Probabilistic Relational Models for Customer Preference Modelling and Recommendation. Tech. Rep.; Laboratoire d'Informatique de Nantes Atlantique (LINA); 2013. URL: `http://hal.archives-ouvertes.fr/hal-00967044`.
15. Pearl, J.. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.; 1988. ISBN 1558604790.
16. Kjaerulff, U.B., Madsen, A.L.. *Bayesian Networks and Influence Diagrams: A Guide to Construction and Analysis*. Springer Publishing Company, Incorporated; 1st ed.; 2007. ISBN 0387741003, 9780387741000.
17. Henrion, M.. Some practical issues in constructing belief networks. In: Kanal, L.N., Levitt, T.S., Lemmer, J.F., editors. *Proceedings of the Third Annual Conference on Uncertainty in Artificial Intelligence (UAI '87)*; vol. 8 of *Machine Intelligence and Pattern Recognition*. Amsterdam: North-Holland; 1989, p. 161–174.
18. Triantaphyllou, E., Mann, S.H.. An examination of the effectiveness of multi-dimensional decision-making methods: A decision-making paradox. *Decision Support Systems* 1989;**5**(3):303–312.
19. Fishburn, P.C.. Additive utilities with incomplete product sets: Application to priorities and assignments. *Operations Research* 1967;**15**(3):537–542.
20. Saaty, T.L.. *The Analytic Hierarchy Process*. McGraw-Hill; 1980.
21. Shani, G., Gunawardana, A.. Evaluating recommendation systems. In: *Recommender systems handbook*. Springer; 2011, p. 257–297.