Recommender System based on Higher-order Logic Data Representation

Linna Li Bingru Yang Zhuo Chen

School of Information Engineering, University Of Science and Technology Beijing

Abstract

Collaborative Filtering help users to deal with information overload and guide them in a personalized way to interesting or useful objects in a large space of possible options. In this paper, we present a novel and elegant hybrid recommender system called *HOLCF*, which use higher-order logic as data representation and integrate content and demographic information into a collaborative filtering framework by using higher-order logic distance computation approaches without the effort of feature construction. Our experiments suggest that the effective combination of various kinds of information based on higher-order logic distance computation approaches provides improved accurate recommendations.

Keywords: Collaborative Filtering; Higher-order logic data representation; higher-order logic distance

1. Introduction

Recommender systems have been widely used in e-commerce. Some companies such as Amazon.com can provide interesting and powerful recommendation services. One of the most difficult challenges for these systems is predicting a rating, which indicates how a particular user liked a particular object. Recommender systems are usually classified into the following three categories according to the sources of data on which recommendation is based. In collaborative filtering systems, the typical sources of data consist of a vector of items and their ratings, continuously augmented as the user interacts with the system over time. In content-based recommender systems, the objects of interest are defined by their associated features and the user will be recommended objects similar to the ones the user preferred in the past. In demographic recommender systems, the systems aim to categorize the user based on personal attributes and make recommendations based on demographic classes. In fact, every approach has its own advantages and disadvantages to making recommendation and the strengths of them are complementary [1].

To avoid certain shortcomings of each pure category recommender systems, many researchers have explored hybrid recommender systems [2, 3]. These hybrid recommender systems can be classified as propositional systems and first-order logic systems according to data representation formalism adopted by them. Propositional hybrid recommender systems employ attribute-value language to describe data. As attribute-value language use a fixed number of features to represent data, the additional effort of data form transformation and feature construction is needed in most of the category hy-

brid recommender systems. At the same time, it makes it difficult to apply these systems to domains where data is rich in structure because the simplicity of attribute-value language. First-order logic hybrid recommender systems are proposed with the emerging of research to multi-relational data mining in recent years. The category systems mainly employ relational database to store data and relational distance to compute similarity between objects. However, the scalability and efficiency of these systems are questionable because of large number of database scans when computing similarity between objects. In this paper, we present a novel and elegant hybrid recommender system called *HOLCF*, which use higher-order logic as data representation and integrate content and demographic information into a collaborative filtering framework by using higher-order logic distance computation approaches. Our experiments suggest that *HOLCF* has improved accurate recommendations than propositional hybrid recommender systems and better scalability and efficiency than first-order logic hybrid recommender systems.

The remainder of the paper is organized as follows. Section 2 introduces the general recommender approaches. The higher-order logic recommender system *HOLCF* are discussed in section 3. In section 4, we conduct experiments to compare *HOLCF* to other hybrid recommender systems. Section 5 concludes our work.

2. Recommender approaches

In this section, the fundamental ideas behind three categories recommender systems are discussed. In a recommender system scenario, there is a list of m users $U=\{u_1, u_2, \ldots, u_m\}$ and a list of n items $I=\{i_1, i_2, \ldots, i_n\}$. Each user has a rating vector V, where $v_{i,k}$ represents that user i's rating for item i_k. (The user's opinion and preference are given by the rating score.) The task of recommender system is for a given active user (or called target user) u_a, (1) to predict the rating score for an unrated item i_k (called the target item) or (2) to recommend some items that may be interesting to user u_a. In this paper, we will concentrate on task 1, since if we can achieve task1, task 2 can be easily achieved with some subjective recommendation strategy.

2.1. Collaborative filtering

There are two types of collaborative filtering (CF) algorithms, user-based and item-based. User-based CF algorithms use some statistical techniques to find a set of users, called user neighbors for the target user. Then different methods can be adopted to combine the user neighbors' ratings to produce a prediction rating for the target user. We discuss in detail item-based CF algorithms in the following.

Item-based collaborative filtering algorithms consist of two processes, finding similar items and generating rating prediction based on similar items' ratings. Many measures can be used to compute the similarity between items, for example, cosine similarity, correlation-based similarity and adjusted cosine similarity. The adjusted cosine similarity between item i_p and i_q can be computed as follows.

$$\mathrm{sim}(\mathbf{i}_p,\mathbf{i}_q)=\frac{\sum_{u=1}^{m}(V_{u,p}-\overline{V}_u)\bullet(V_{u,q}-\overline{V}_u)}{\sqrt{\sum_{u=1}^{m}(V_{u,p}-V_u)^2(V_{u,q}-\overline{V}_u)^2}}$$

Here, $\overline{V}_k$ is the average rating of user u.

After computing the similarity between items, a set of most similar items to the target item can be obtained. Suppose the set is S_k. the prediction $v_{a,k}$, on target item i_k for user u_a can be computed by adopting these similarity and ratings of

each item in S_K given by user u_a. We use a weighted sum as follows.

$$\mathbf{V}_{a,k} = \frac{\sum_{i_p \in S_k} (\mathbf{V}_{a,p} \bullet \mathrm{sim}(\mathbf{i}_k, \mathbf{i}_p))}{\sum_{i_p \in S_k} \mathrm{sim}(\mathbf{i}_k, \mathbf{i}_p)}$$

2.2. Content-based Recommendations

Content-based methods make recommendations by analyzing the description of the items that have been rated by the user and the description of items to be recommended. A variety of algorithms have been proposed. Many approaches are specialized versions of classification learners, in which the goal is to learn a function that predicts which class an item belongs to (i.e. either liked or not-liked). Other algorithms would treat this as a regression problem in which the goal is to learn a function that predicts a numeric value. There are two important sub problems in designing a content-based filtering system. The first is finding a representation of items. The second is to create a profile that allows for target items to be recommended.

2.3. Demographic-based Recommendations

Demographic information can be used to identify the types of users that like a certain item, for example, the information on the age, gender, education, etc. of users that rated an item together with their rating of the item. One might expect to learn the types of user that like a certain item. A demographic-based recommender system attempts to identify one of pre-existing clusters to which a user belongs to tailor recommendations to users based upon information about others in this cluster.

3. Recommender system based on Higher-order Logic Data Representation

The recommender system *HOLCF* actually is item-based collaborative filtering. One critical step in the item-based collaborative filtering algorithm is to compute the similarity between items. Here, we want to calculate similarity between items by adopting not only ratings on these items but also some semantic information about items. We employ *Escher* [4, 5], a higher-order logic declarative programming language, to represent item individuals which include all above mentioned information. Furthermore, we use distance measure over the representation with *Escher* to address exactly the problem of calculate similarity using all these information.

3.1. Escher

Escher is a strongly typed declarative programming language which integrated the best features of both functional and logic programming language. It has types and modules, higher-order and meta-programming facilities, concurrency, and declarative input/ouput. It combines the best ideas of existing functional and logic languages, such as Haskell and Gödel in a practical and comprehensive way [4]. As the knowledge representation formalism, its basic principle is that an individual should be represented by a term. In order to describe knowledge, it needs the following syntax:

- Integers, natural numbers, floats, characters, strings, and Booleans.
- Data constructors.
- Tuples.
- Sets.
- Multisets.
- Lists.
- Trees.
- Graphs.

The first group, called the base types, is the basic building blocks for knowledge representation. The type of integers is denoted by *Int*, the type of natural

numbers by *Nat*, the type of floats by *Float*, the type of characters by *Char*, the type of strings by *String*, and the type of Booleans by Ω.

Also needed are data constructors for user-defined types. For example, see the data constructors *Sunny*, *Overcast*, *Rain* for the type *Outlook* in example 1. These data constructors have arity 0 and are usually called *constants*. Data constructors for more complicated types are also needed and will be discussed in the following.

Example 1. Consider the classical dataset used in machine learning: playing tennis according to the weather. Type weather can be defined as follows.

Outlook = *Sunny*|*Overcast*|*Rain*;
Temperature=*Hot*|*Mild*|*Cool*;
Humidity=*High*|*Normal*|*Low*;
Wind=*Strong*|*Medium*|*Weak*;
Weather=*Outlook*×*Temperature*×*Humidity*×*Wind*

Here, *Outlook*, *Temperature*, *Humidity*, *Wind* and *Weather* all are types, each of them is composed with its own data constructors. *Weather* is also a user-defined type which is composed by types *Outlook*, *Temperature*, *Humidity* and *Wind*.

Tuples are essentially the basis of the attribute-value representation of individuals. It is constructed by data constructor ×. If $\mu_1, \ldots, \mu_n$ are types, then $\mu_1 \times \ldots \times \mu_n$ is a tuple.

Set is constructed by data constructor {}. If μ is a type, then $\{\mu\}$ is a set whose elements have type μ.

Multisets are a useful generalization of sets. A straightforward approach to multisets is to regard a multiset as a function of type $\mu \rightarrow Nat$, where μ is the type of elements in the multisets and the value of the multiset on some item is its multiplicity, that is, the number of times it occurs in the multisets.

Lists are represented using the following data constructors.

[]: *List* μ
(:):$\mu \rightarrow List\ \mu \rightarrow List\ \mu$

A list has a type *List* μ, where μ is the type of the items in the list. Lists are constructed as usual from the empty list [] and the cons function (:).

The standard representation of a tree (where the subtrees have an order) uses the data constructor *Node*, where

Node: $\mu \rightarrow List(Tree\ \mu) Tree\ \mu$

Here *Tree* μ is the type of a tree and *Node* is a data constructor whose first argument is the root node and second argument is the list of subtrees.

Graphs are divided into undirected Graphs and directed Graphs. The type of an undirected graph is *Graph v* ε, where *v* is the type of information in the vertices and ε is the type of information in the edges.

Label=*Nat*
Graph v ε=$\{Label \times v\} \times \{(Label \rightarrow Nat) \times \varepsilon\}$

The type of an undirected graph is *Digraph v* ε, where *v* is the type of information in the vertices and ε is the type of information in the edges.

Digraph v ε=$\{Label \times v\} \times \{(Label \rightarrow label) \times \varepsilon\}$

3.2. Higher-order logic distance

As each individual is expressed by a term, the distance on two terms can be defined as follows.

Distance function *d* is: *d*: ß×ß→*R*, *R* denotes the set of real numbers. Here, ß represents a basic term which is represented with Escher [6].

Definition 1. The function *d:* ß×ß→*R*, is defined inductively on the structure in ß as follows, let *s*, *t*∈ß,

(1) If *s*, *t*∈ßα,where $\alpha=T_{\alpha 1 \ldots \alpha n}$, for some *T*, $\alpha_1, \ldots, \alpha_n$, then If $C \neq D$, $d(s, t)=1$, otherwise, $d(s,t) = \sum_{i=1}^{n} \frac{1}{2^i} d(s_i, t_i)$ where *s* is $C\ s_1 \ldots s_n$ and *t* is $Dt_1 \ldots t_m (n \geq m)$, $s_1, s_2, \ldots, s_n, t_1, t_2, \ldots, t_m$ all are type, *C* and *D* are type constructor. $C\ s_1 \ldots s_n$,

represents a type which is constructed by $s_1 \ldots s_n$ under type constructor C. $Dt_1 \ldots t_m$ have the similar meaning. $ß\alpha = \{t \in ß \mid t$ has type more general than $\alpha\}$. The intuitive meaning of $ß\alpha$ is that it is the set of terms representing individuals of type α.

(2) If $s, t \in ß\alpha$, where $\alpha = \beta \to \gamma$, for some β, γ, then

$$d(s,t) = \frac{\Sigma_{B\beta} d(V(sr), V(tr))}{1 + \Sigma_{B\beta} d(V(sr), V(tr))}$$ where, if β, γ typed, then $\beta \to \gamma$ represents a multiset, each element of which is type β, γ usually is type *Nat*. $V(t, r)$ is the value returned when t is applied to r.

(3) If $s, t \in ß\alpha$, where $\alpha = \alpha_1 \times \ldots \times \alpha_n$, for some $\alpha_1, \ldots, \alpha_n$, then $d(s,t) = \frac{1}{n} \sum_{i=1}^{n} d(s_i, t_i)$ where s is $(s_1, \ldots, s_n)$ and t is $(t_1, \ldots, t_n)$.

(4) If there does not exist $\alpha \in \upsilon^*$ such that $s, t \in ß_\alpha$, then $d(s, t)=1$.

In Part 1 of the definition, if $n=0$, then $\sum_{i=1}^{n} \frac{1}{2^i} d(s_i, t_i) = 0$. The factor of 1/2 means that the greater the "depth" to which s and t agree, the smaller will be their distance apart. So for lists, for example, the longer the prefixes on which two lists agree, the smaller will be their distance apart. This part is the case for complex type is composed by several basic types or user-defined types. It also can be the condition that several complex types compose more complex type. The definition means that distance between complex types is determined by distance between their subtypes.

Also in Part 2 of the definition, the sum $\sum_{r \in B_\beta} d(V(sr), V(tr))$ is finite since s and t differ on at most finitely many r in $ß_\beta$. In the case of sets and multiset, $\sum_{r \in B_\beta} d(V(sr), V(tr))$ is the cardinality of the symmetric difference of the sets s and t. The part mainly is for the case of set and multiset.

Part 3 is the case for tuple type.

Part 4 means that if the types of two individuals do not match, the distance is 1.

It should be clear that the definition of d does not depend on the choice of α such that $s, t \in ß\alpha$. There may be more than one such α. What is important is only whether α has the form $T_{\alpha 1 \ldots \alpha n}$, $\beta \to \gamma$, or $\alpha_1 \times \ldots \times \alpha_n$.

The definition given above for d is a general framework. As for specific application, there are some possibilities. For example, in part 1, a weight w_i can be set for $d(s_i, t_i)$, $d(s, t)$ can be $\sum_{i=1}^{n} w_i d(s_i, t_i)$ ($\sum_{i=1}^{n} w_i = 1$). $d(s, t)$ can be a function φ about $\sum B_\beta d(V(sr), V(tr))$ and satisfy $\varphi(x) \in [0.$ 1] and $\varphi(x+y) \le \varphi(x) + \varphi(y)$ in part 2.

It can be proofed that the distance measure takes values in the range [0, 1] and triangle inequality holds for it. The similarity between two individuals is 1-d.

4. Experiments

4.1. Datasets and Evaluation Measure

Two well-known datasets of movie rating are used in our experiments: MovieLens [7] and Book-Crossing [8]. For MovieLens dataset, we extracted a subset of 500 users with more than 40 ratings. For Book-Crossing dataset, we extracted a subset of 10,000 users with more than 40 ratings.

Recommender systems research has used several types of measures for evaluating the quality of a recommender system. We use MAE as our choice because it is most commonly used and easiest to interpret directly. MAE is a measure of the deviation of recommendations from their true user-specified values. For each ratings-prediction pair $<p_i, q_i>$ this metric treats the absolute error between them,

i.e.,$| p_i - q_i |$ equals. The MAE is computed by first summing these absolute errors of the N corresponding rating-prediction pairs and then computing the average. Formally,

$$MAE = \frac{\sum_{i=1}^{N} | p_i - q_i |}{N}$$

The lower the MAE, the more accuracy the recommendation system predicts user ratings.

4.2. Experiments Results

We compared *HOLCF* to standard collaborative filtering approaches, including user-based Pearson Correlation Coefficient (User-Based), item-based approach (Item-Based) which only use collaborative information [9] and inductive learning approach (IL) which use collaborative and content information [2].The results of our experiments are shown in Fig. 1. Our experiments suggest that the effective combination of various kinds of information based on relational distance approaches provides improved accurate recommendations than other approaches.

	MovieLens	Book-Crossing
User-Based	0.741	0.731
Item-Based	0.733	0.725
IL	0.727	0.717
HOLCF	0.719	0.707

Fig. 1: Experimental result.

5. References

[1] Pazzani, M. J. A Framework for Collaborative, Content-Based and Demographic Filtering. Artif. Intell. Rev. 13, 393-408. 1999.

[2] C.Basu, H.Hirsh and W.Cohen. Recommendation as classification: using social and content-based information in recommendation. In Proceedings of the Fifteenth National/Tenth Conference on Artificial intelligence/innovative Applications of Artificial intelligence, Menlo Park, CA, 714-720. 1998.

[3] Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions. IEEE Trans. on Knowl. and Data Eng. 17, 734-749. 2005.

[4] J.W.Lloyd. Declarative Programming in Escher. Technical Report CSTR-95-013, Department of Computer Science. University of Bristol, 1995.

[5] P.A.Flach, C.Giraud-Carrier and J.W.Lloyd. Strongly Typed Inductive Concept Learning. In Proceedings of the Eighth International Conference on Inductive Logic Programming. LNAI 1446, Springer-Verlag, pp. 185-194. 1998.

[6] J.W.Lloyd. Logic for Learning: Knowledge Representation, Computation and Learning in Higher-order Logic. Springer-Verlag Berlin and Heidelberg GmbH & Co. KG, 2002.

[7] MovieLens dataset : http://www.cs.umn.edu/Research/ GroupLens/

[8] Book-Crossing dataset: http://www.informatik.uni-freiburg.de/~cziegler/BX/

[9] Sarwar B., Karypis G., Konstan J., and Reidl J. 2001. Item-based collaborative filtering recommendation algorithms. In Proceedings of the 10th international Conference on World Wide Web (Hong Kong, Hong Kong, May 01 - 05, 2001). WWW '01. ACM, New York, NY, 285-295. DOI= http://doi.acm.org/10.1145/371920.372071

[10] http://racr.shmtu.edu.cn/en_index.asp.