전자상거래를 위한 상품 추천 에이전트에서의 사용자 질의 처리 모델

이승수⁰ 이광형
한국과학기술원 전자전산학과 전산학전공
(sslee⁰, khlee)@if.kaist.ac.kr

User Query Processing Model in the Item Recommendation Agent for E-commerce

Seungsoo Lee⁰ Kwang H. Lee
Dept. of EECS, KAIST

Abstract

The rapid increase of E-commerce market requires a solution to assist the buyer to find his or her interested items. The intelligent agent model is one of the approaches to help the buyers in purchasing items in online market. In this paper, the user query processing model in the item recommendation agent is proposed. In the proposed model, the retrieval result is affected by the automatically generated queries from user preference information as well as the queries explicitly given by user. Therefore, the proposed model can provide the customized search results to each user.

1. Introduction

Currently the E-commerce market is growing with steep increasing rate. This makes it possible for us to purchase an item that we want, but also makes it difficult to find an item that we are interested in. Searching engines are the most general solution to handle these problems, but a user agent can be an alternative solution. The major advantage of user agent is that it can be specialized from user to user.

User agent can be used in E-commerce to assist user in many different aspects. It can search items on behalf of user, or negotiate with other agents in purchasing items, or automatically collect information that the user may be interested in.

This paper is mainly focused on the processing of user query to retrieve an item list that matches the given query. This paper is based on the user agent model proposed by Lee et al. in [4]. This model is briefly reviewed in section 2, and the processing model of user query is proposed in section 3. The conclusion and further works will follow in section 4.

2. Recommendation agent

In this section, we will briefly review the agent model that is used in user query processing. It is called the recommendation agent. More detailed explanations can be found in [4].

The recommendation agent is designed to support the following functionalities as shown in Fig. 1. It can process user query to find interested items, and collect information and provide it to user automatically. Most importantly, it must be able to keep up with the user's requirements and preference.

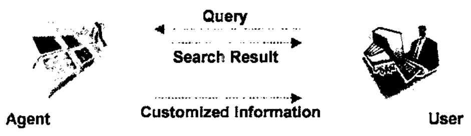

Fig. 1 Recommendation agent

The structure of recommendation agent is shown in Fig. 2. Two kinds of key information are stored in the structure: item type information and user preference information. And the agent categorized each item with two different measures: item type and evaluation criteria. User preference values are also categorized using the same measures. The user history database is used to store the interactions between user and the agent.

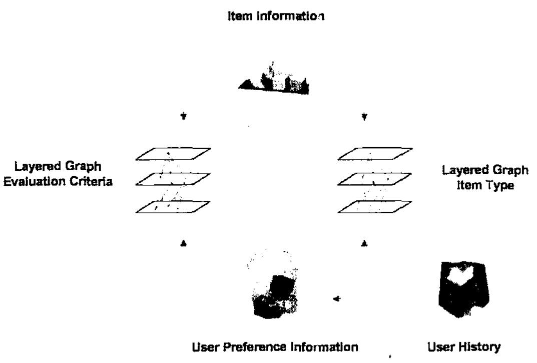
Item Information
Layered Graph Evaluation Criteria
Layered Graph Item Type
User Preference Information
User History

Fig. 2 Structure of recommendation agent

Layered graph is the key component in the recommendation agent. It is used for expressing and managing user preference value. When there is any change in user preference, it can be easily updated using propagation in the layered graph. The structure and operations on layered graph can be found in [3].

3. Processing user query

The processing model of user query is proposed in this section. The overall procedure is shown in Fig. 3.

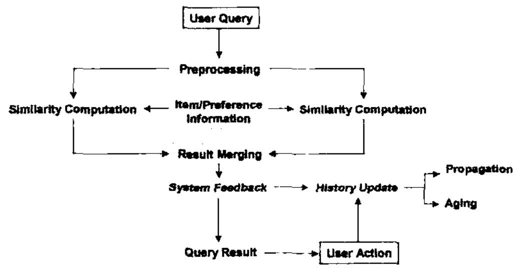

Fig. 3 Overall procedure of user query processing

3.1 Preprocessing

Preprocessing is composed of three steps: conversion, decomposition and insertion.

First, it converts any implicit queries into explicit queries. Searching option such as "Find similar items like this" is a good example of implicit queries. If these kinds of queries are given to the agent, they need to be converted to explicit ones before the calculation of matching degree of each item. This conversion is quite dependent on the current user context. This conversion step can be skipped when user gives explicit queries to the agent.

After conversion, given queries are decomposed into multiple sub-queries. The number of sub-queries is equal to the number of the layered graph in the agent. We use two layered graph in the proposed model as shown in Fig. 2, therefore the number of sub-queries will be two. Because each sub-query is related to one of the layered graph, one of the two sub-queries will be the query related to the item type and the other will be the query related to the evaluation criteria. For example, if user gives a query "I need a black shoes that is not so expensive", this will be decomposed into two sub-queries such as "an item that is a black shoes" and "an item that is not so expensive".

The last step is insertion. If a layered graph has no sub-queries that are related to it after the decomposition step, the agent itself will generate a customized query for that layered graph. In the above example, if user gives a query "I need a black shoes", then there will be no sub-queries related to the layered graph of evaluation criteria after the decomposition. Then the agent will decide which kind of item is more suitable for user request. The generated queries are based on user history and user preference information. For example, in the case of price, if the agent thinks that this user is generally more interested in an item that is not so expensive, it will automatically add a new sub-query "item that is not so expensive". Using this insertion step, each user can get a difference query result based on their preference even if all of them give the same queries to the agent.

3.2 Similarity computation

After preprocessing, matching degrees between sub-queries and item information in the layered graph are computed. If we denote the set of leaf vertices in layered graphs as V and the set of items as T such that

$$V = \{v_1, \cdots, v_m\}$$
$$T = \{t_1, \cdots, t_n\}$$

Then each item can be described with its position in the layered graph defined as follows that is called an item descriptor.

$$t_i = \{(v_j, \mu_{t_i}(v_j)) \mid v_j \in V\}$$

where μ is the membership function of t_i to a set v_j.

To compare the user query and item information, each sub-query is converted to a virtual vertex t_q.

$$t_q = \{(v_j, \mu_{t_q}(v_j)) \mid v_j \in V\}$$

The calculation of similarity degree is adopted from the vector model in information retrieval [1]. The similarity between t_q and $t_i \in T$ is defined as the correlation between two vector $\vec{\tau}_q$ and $\vec{\tau}_i$

$$sim(t_q, t_i) = \frac{\vec{\tau}_q \cdot \vec{\tau}_i}{|\vec{\tau}_q||\vec{\tau}_i|}$$

where $\vec{\tau}_q$ and $\vec{\tau}_i$ are defined as

$$\vec{\tau}_q = (\mu_{t_q}(v_1), \cdots, \mu_{t_q}(v_m))$$
$$\vec{\tau}_i = (\mu_{t_i}(v_1), \cdots, \mu_{t_i}(v_m))$$

3.3 Result merging

After the similarity computation, we will get multiple numbers of computational results generated from each layered graph. This multiple results are merged into one final result that will be shown to user in the merging step.

As we mentioned in 3.1, recommendation agent may insert some customized queries in the preprocessing step and this will also generate a result list. Intuitively, the queries given directly by user must be considered more important than the queries automatically generated by the agent. We will use the weight factor α to balance between these two kinds of queries. If the result list from given queries is denoted as R_{given} and the list from generated queries is denoted as $R_{generated}$, then the merged list R can be obtained using the following equation.

$$R_{given} = \{(t_i, sim_{given}(t_q, t_i)) \mid t_i \in T\}$$
$$R_{generated} = \{(t_i, sim_{generated}(t_q, t_i)) \mid t_i \in T\}$$
$$R = \{(t_i, r(t_i)) \mid t_i \in T\}$$
$$\text{where } r(t_i) = \max\{\alpha \cdot sim_{given}(t_q, t_i), (1-\alpha) \cdot sim_{generated}(t_q, t_i)\}$$

If we use $\alpha = 1$, then the result list R is completely dependent on the given queries by user, whereas it is completely dependent on the generated queries by agent when $\alpha = 0$. In general situations, the meaningful range of α value will be $0.5 \le \alpha \le 1$.

3.4 System feedback

Feedback is essential point to make the system keep up with current user preference.

System feedback can be initiated in three different cases. The first case is when a user query is given. User query has the information of item groups that the user has interested in, and the preference values of highly evaluated items will be increased in the system feedback phase. The second case is direct user actions on some specific items. The examples of the second case are the events that user selects an item to view its detailed information, or the events that user purchases an item. Each of these events will increase or decrease the user preference of targeted item. The last case is some special operations that will be initiated by the agent itself. Aging is a typical example of the last case that will decrease the preference values of the items that are out of user interest for the given period of time.

4. Conclusion

The processing model of user query in recommendation agent is proposed in this paper. In the case of search engine or information retrieval system, same query always gives the same result. Using the proposed model, however, each user will be provided the customized search results using automatically generated queries that is obtained from user preference. The query result will be different from user to user, and even will be different for the same user with time because user preference is always subject to change.

Currently the similarity degree is calculated independently for each layered graph in the proposed model. But if the interrelation between two different layered graphs is considered in the similarity computation phase, it may improve the search result. It can be considered as a further work. And the estimation model of user preference for new information from current user preference and item relations is also considered as a further work.

Reference

[1] Richardo Baeza-Yates and Berthier Ribeiro-Neto, "Modern Information Retrieval," *Addison-Wesley*, 1999.

[2] Seungsoo Lee and Hyung Lee-Kwang, "A Layered Network Model for User Characteristics," *Proc. of KFIS Fall Conference 1999*, vol. 9 no. 2, pp. 108-111, 1999.

[3] Seungsoo Lee and Hyung Lee-Kwang, "Fuzzy Layered Graph for User Modeling," *Proc. of 2001 HCI Conference*, pp. 409-414, 2001.

[4] Seungsoo Lee and Hyung Lee-Kwang, "Item Recommendation Agent using Fuzzy Layered Graph," *Proc. of the 28th Conference of Korea Information Science Society (B)*, pp. 343-345, 2001.