A Scalable House Recommendation System

Ming-ruo SHI
Beijing Wuzi University, 101149, China

Keywords: Recommendation, Content based filtering, Ranking, Hadoop.

Abstract. A recommendation system is increasing important in online video application, news recommendation and electronic commerce especially in search engine. Recommendation on search engines are more related to hundreds of millions of users like Google, Bing and Baidu. Our proposed filtering solution is based on content match to handle large number of users and houses for search engine.

Introduction

Recommender systems have become extremely common in recent years, and are utilized in a variety of areas: some popular applications include movies, music, news, books, research articles, search queries, social tags, and products in general. There are also recommender systems for experts [1], collaborators [2], jokes, restaurants, garments, financial services [3], life insurance, online dating, and Twitter pages [4].

Recommender systems typically produce a list of recommendations in one of two ways – through collaborative and content-based filtering or the personality-based approach [5,11,12]. Collaborative filtering approaches building a model from a user's past behavior (items previously purchased or selected and/or numerical ratings given to those items) as well as similar decisions made by other users. This model is then used to predict items (or ratings for items) that the user may have an interest in [6]. Content-based filtering approaches utilize a series of discrete characteristics of an item to recommend additional items with similar properties [7]. These approaches are often combined (called Hybrid Recommender Systems).

A recommendation system usually including two major parts. One is recommendation which will create a recommendation item candidates based on user's visit/click history. The other part is to create top n recommendation results by ranking the recommended candidates. There are some papers discussing some scalable recommendation to users including Google news recommendation [8,10]. To our best knowledge, no literature gives the detailed introduction how a large-scale recommendation system is designed and implemented scaled both to number of users and items on the house recommendation vertical.

In this paper, we proposed a Content-based filtering recommendation algorithm scalable to both users and items which can be applied to house recommendation esp. on search engine.

The paper is organized as follows: in section 2, we describe in detail our algorithm to support large numbers of users and houses. In section 3 introduces our experiments. We summarize our ongoing work in section 4.

Algorithm

Problem Statement

Problem statement is for recommendation as follows:

Input:
- A list of N users \(\{U_1, U_2, ..., U_N\}\) with user profile
- A list of M hours \(\{H_1, H_2, ..., H_m\}\) with house content and status information
- The visit history of all users for the fixed period (e.g. 3-month log)

Output:
- A list of preferred houses for each user for today.
Our solution is to develop a content-based collaborative filtering recommendation system. We firstly extract the side information for user to build user profile and house information for match. Then use the visiting history and side information including user profile and house information in both training and rank phases of the algorithm.

**Basic Idea**

At high level, the idea to recommend to a user, say $A$, is to find the set of users, say $C_A$, that are like $A$, and recommend to $A$ items viewed by users in $C_A$. Pseudocode:

**Algorithm A:**

Choose a threshold $\theta$

For each user $A$ and possible interested house $H$

- $C_A \leftarrow \emptyset$ // the set of houses user $A$ may be interested in
- If $s(P_A, H) \geq Q$ // $S(P_A, H)$ is the score how $A$ is possibly interested in house $H$
  - Add $(H, s(P_A, H))$ to $C_A$
- Recommend to $A$ at most $k$ houses in $C_A$ by rank score $s(P_A, H)$

**Build User Profile**

Nowadays search engine has the capacity to build user profile based on user’s search behavior. The basic user profile information includes age, location, gender etc. which could get for registered users.

Besides the mentioned basic user information, search engine could leverage the information inferred from user’s daily search activities, i.e. searched queries and click activities. For house segment, barely queries themselves are hard to build user profile.

We build user profile by leveraging search graph of house entity detailed information. The most important information will be extracted from house entity are address, zip code, listing price, sold price, city, built year, square feet, number of bed rooms. We can also leverage more house entity information such lot size, estimated price. Then we build (feature, feature value) key value pairs by aggregating a fixed period search history for example 90 days.

House features for user profiles could be built by different ways for a specific recommendation target. Propose one way including features: country, city, zip code, zip code | listing price bucket, built year, zip code | sold price, square feet. The feature value is percentage by aggregating last 90-day search history.

**Extract House Features**

Our target is to recommend houses to users. We therefore only consider active listing houses. And extract the same features as user profile. For example, we could extract features such as country, city, zip code, zip code | listing price, built year, and square feet.

**Rank Recommendation Result**

We could calculate match score between users and houses with equal country and city. For matched user and house pair, we could get match score by the following way:

$$s_{u,h} = \sum_{i=1}^{n} c_i f_i .$$

$s_{u,h}$ is the match score. $n$ is the number of matched features for user $u$ and house $h$. $f_i$ is the feature value in user’s profile for a give user $u$, we get top $k$ houses ranked by match score satisfy $s_{u,h_1} \geq s_{u,h_2} \geq \cdots \geq s_{u,h_k}$ where $k$ is a predefined number.

**Decide Coefficients**

There are a few ways to decide the coefficients $c_i$ during user and house matching process. We use logistic regression machine learning method on user search history to figure out the optimized coefficients $c_i$. 

93
Finally search engine could make a recommendation to user in some ways. For example, a user could be recommended by search engine’s home page or recommendation email.

Experiments

Evaluation Methodology

We take three-month house entity search log in our test search engines and split the visit log into the training set and the test set. The test set is one-day log, i.e., the visit log of the latest day in the log. We use the training set to get recommendation for every user and compare the recommendation list with the list of houses that users visited in the test set. When we measure, we remove users/entities in the test file that is NOT in the training file. Here note that the offline evaluation does not truly reflect the performance of the online system (the performance should be much better when we launch the system online) since we haven’t posed the recommendation to users yet. But it helps us determine what ideas can potentially improve the performance of the online system.

Running time: In our log, the number of distinct users is up to 15364267 (~15 million) and houses is 5121937 (about 5 million). We filtered out the traffic visited by user with age less than 18 years old. The training can be done in 4 hours, using about 70 computers in a Hadoop cluster. So, in the following, we mainly discuss about the prediction accuracy.

Two main metrics below are considered for the prediction accuracy:

1. TrueRate: We assign score 1 to a user if he/she visited a recommended item in the test set and 0 otherwise.

   \[
   \text{TrueRate} = \frac{\text{Total Score}}{\text{Total # Users}}.
   \]

   Essentially, TrueRate computes the fraction of users searched for at least one recommendation in the test set

2. NDCG (Normalized Discounted Cumulative Gain): we define NDCG @ k as:

   \[
   \text{NDCG}_k = \frac{\text{DCG}_k}{\text{DCG}^*_k}.
   \]

   where \( \text{DCG}_k = \sum_{i=1}^{k} 2^r_i \frac{r_i}{\log_2(i+1)} \) and \( \text{DCG}^*_k \) is the ideal \( \text{DCG}_k \). We assign \( r_i \) to equal 1 if there was a click on item ranked in position \( i \). Here, \( k \) is the recommendation length, i.e., select top \( k \) entities/news to present to users. Basically, NDCG assign higher score to visited items that in the top of the recommendation list than the visited items in the bottom of the list.

![Figure 1. True rate.](image)
Evaluation Result Besides our algorithm, we also developed a Collaborative Filtering algorithm which doesn’t use any house entity information at all. However, our algorithm takes important signals from users’ search history and search graph. Note that we only take top (at most) 3 houses to recommend to users. The evaluation results are shown in Figure 1. and Figure 2.

As Figure 1. and Figure 2. suggest, our algorithm clearly outperforms collaborative filtering algorithm.

Future Work

Firstly, personalized ranking methods from Information Retrieval could be leveraged. Researchers came up with various ranking strategies to present relevant information to users. There is a line of research that incorporate user’s interests into the ranking, called personalized ranking [11]. The same idea can be applied in ranking the recommendation result. Finally, probabilistic recommendation algorithm might be helpful. pLSI (probabilistic latent semantic indexing)-based algorithm may outperform our hashing-based algorithm [12].

Acknowledgement

This research was financially sponsored by the key lab of intelligent logistic system BZ0211 PXM2015_014214_00000.

References


