

Recommender Systems

Bamshad Mobasher

Recommender systems have become an important part users' everyday interactions with Web based applications, particularly those driving e-commerce. Businesses have come to realize the potential of these personalized and adaptive systems in order to increase sales and to retain customers. Likewise, Web users have come to rely on such systems to help them in more efficiently finding items of interest in large information spaces. We provide a brief characterization of the recommendation problem, in general, and summarize various approaches used for recommendation generation.

1 Introduction

The ability of a recommender system to tailor its output to a particular user implies that it must be able to infer what that user requires based on previous or current interactions with that user or other similar users. The recommendation task can therefore be viewed as a prediction problem: the system must attempt to predict the utility of specific content categories, pages, or items, for a given user and then rank these according to the predicted utilities. The utility of an item is typically expressed as a numerical score reflecting the user's level of interest in that item. The output of a recommender system is generally a set of items with the highest predicted interest values or utilities for an active user. Therefore, a recommender system can be viewed as a mapping of users and items to a set of utility values (or interest scores). The view of recommendation as a prediction task comes from the fact that this mapping is not, in general, defined on the whole domain of user-item pairs, and thus requires the system to estimate the interest values for some elements of the domain.

More formally, let us assume the existence of a set of users, $U = \{u_1, u_2, \dots, u_m\}$, and a set of items, $I = \{i_1, i_2, \dots, i_n\}$. The profile for a user $u \in U$ can be viewed as an n -dimensional vector of ordered pairs,

$$u^{(n)} = \langle (i_1, s_u(i_1)), (i_2, s_u(i_2)), \dots, (i_n, s_u(i_n)) \rangle,$$

where i_j 's $\in I$ and s_u is a partial function for user u , assigning utility values to items in I . Thus, the function $s_u : I \rightarrow R$ represents the profile of a user u , mapping items to an ordered set of utility values, R . In systems involving explicit ratings for items, the function s_u is called a *rating function*, mapping the items to a discrete set of ratings. Note that because the mapping s_u for a given user u is not in general defined on the whole domain of items, the system must estimate or predict the interest scores of a given user for some elements of the domain.

In a typical recommender system, user profiles are collected over time and stored for all users. Conceptually, the database of all user profiles can be represented as the $m \times n$ matrix, $UP = [s_{u_k}(i_j)]_{m \times n}$, where utility $s_{u_k}(i_j)$ represents the degree of interest in item i_j by a user u_k . Alternatively, we may view UP as a set of n -dimensional user profile vectors.

A recommender system can therefore be viewed as a mapping $REC : \mathcal{P}(UP) \times U \rightarrow \mathcal{P}(I)$, with each user mapped

to a set of recommended items based on that user's profiles and a (possibly empty) set of other user profiles. Typically, the range of this mapping is a subset $I' \subseteq I$ of items not previously rated by the user. If we assume, without loss of generality, that the item with the highest predicted value for a given *target user* $u_k \in U$ is to be returned, then the mapping REC is defined by:

$$REC(up, u_k) = \{\operatorname{argmax}_{i_j \in I'} s_{u_k}(i_j)\},$$

where, up is a (possibly empty) subset of the set of user profiles, UP , and $s_{u_k}(i_j)$ is the predicted interest score of user u_k on item i_j . Typically, recommender systems return a set of top N recommended items sorted by their predicted interest scores. Depending on the prediction algorithm used, the system may not be able to arrive at an interest score for a particular item, in which case the REC mapping may be assumed to produce a `null` or an undefined value.

Traditional approaches to personalized recommendation have included content-based, collaborative, and knowledge-based systems. Each of these approaches is distinguished by the specific type of data collected to construct user profiles, and by the algorithmic approach used to generate recommendations. We discuss the common approaches to personalized recommendation in more detail below.

2 Types of Recommender Systems

From an architectural point of view, recommendation generation approaches fall into two main categories: *memory-based* and *model-based*. Standard user-based collaborative filtering and most content based filtering systems that use lazy learning algorithms are examples of the memory-based approach, while item-based and other collaborative filtering approaches that learn models prior to deployment are examples of model-based systems. Memory based systems simply memorize all the data and generalize from it at the time of generating recommendations. They are therefore more susceptible to scalability issues. Model-based approaches, that perform the computationally expensive learning phase offline, generally tend to scale better than memory based systems during the online recommendation generation. On the other hand, as more data is collected, memory based systems are generally better at adapting to changes in user interests compared to model based techniques whose models must