Dealing with the Cold Start Problem when Providing Personalized Enterprise Content Recommendations

Tolga Oral¹, N. Sadat Shami²

¹IBM Corporate Headquarters, ²IBM TJ Watson Research Center
1 Rogers St., Cambridge, MA 02142, USA
+1 720 342 1459, +1 617 693 1409
{tolga_oral, sadat} @ us.ibm.com

ABSTRACT
We demonstrate how we can take advantage of employee information and digital traces of interaction in order to provide personalized recommendations of content to enterprise users. In particular, we focus on the cold start problem encountered when a service lacks relevant data to base recommendations. Our recommendation service – Steer – first seeks out user preference data. In the absence of such data it utilizes a wide range of sources to create a user’s enterprise interest profile, which is used to provide recommendations. Users that Steer cannot generate an interest profile for are given generic recommendations based on a novel algorithm. We describe the algorithm that drives Steer, how the service scales, and takes into consideration content that could be outdated. We conclude with a discussion of an evaluation plan for Steer and future directions we wish to explore.

Categories and Subject Descriptors
H.5.3 [Information Interfaces and Presentation]: Group and Organizational Interfaces – Collaborative computing; H.3.3 [Information Storage and Retrieval]: Information Search and Retrieval – Information filtering

General Terms

Keywords
Recommender systems, collaborative filtering, social web.

1. INTRODUCTION
The popularity of Web 2.0 and social software applications has led to a proliferation of digital traces of user activity. Enterprises are increasingly realizing the value of these applications, leading to increased adoption of social software for work purposes [e.g. 7]. These traces can be mined to infer such characteristics as social connections, interests and expertise [4, 10]. Such aggregated data from the collective may point us to new ways to discover and recommend content.

There has been recent interest in utilizing social software data for providing recommendations [6]. Using social software data for content recommendations in an enterprise setting presents both opportunities and challenges. Unlike the internet, employees in an organization can be tied to a corporate directory. That data can be leveraged for recommendations. Employees also participate in social software in the organization, which leaves digital traces that can be mined [4]. However, not everyone in an organization uses social software or are active users. A recent study of an enterprise social file sharing system found that 72% of users were lurkers, leaving no trace of their participation [8]. Such usage is not atypical for social software use in the enterprise. Furthermore, unlike internet users, enterprise users are much smaller in number and are limited to their firewall-protected intranet where typically very little cross-application linking occurs. The lack of digital traces to base recommendations creates a challenge for enterprise recommender systems. This leads to the ‘cold start’ problem – the difficulty in providing recommendations of content when there is no data available to base it on [9]. In the absence of data to base recommendations on, how can a service provide personalized recommendations?

In this paper we describe a solution to the cold start problem for enterprise users that leverages the collective activity of others. We focus on a particular application, providing recommendations for ‘innovations’ in an enterprise that are relevant to a user, although our approach can be applied to other applications as well. We start by reviewing prior work in this area and proceed by introducing our recommendation service. We describe its various features, including the algorithm that drives the service. We conclude with a rough evaluation plan and directions for future work.

2. RELATED WORK
Prior research on providing personalized recommendations of content for enterprise users has been limited. Guy et al. discuss a personalized recommendation service for bookmarked webpages, blog entries, and online communities [6]. Their recommendation service uses the SONAR social network aggregation service [4] and is based on calculating a user’s social network [6]. However, providing recommendations when one’s social network cannot be inferred remains a challenge. In this research, we add to the work on providing personalized recommendations for enterprise users by considering other sources in addition to social networks. There has also been research on ways to recommend people for ‘innovations’ of interest to a user, although our approach can be applied to other applications as well. We start by reviewing prior work in this area and proceed by introducing our recommendation service. We describe its various features, including the algorithm that drives the service. We conclude with a rough evaluation plan and directions for future work.

The cold start problem can be a significant impediment to providing personalized recommendations in an enterprise. There has been limited treatment of this problem for enterprise settings. Work by Schein et al. provides some guidance in this regard [9]. They use a folding in algorithm to combine collaborative data with content data to make predictions for unrated movies. Somewhat similarly, Givon and Lavrenko combine relevance models with collaborative filtering to predict social tags for new books [3]. In this paper, we attempt to contribute to solutions to the cold start problem by describing an approach that addresses it for an enterprise setting.
3. THE ‘STEER’ RECOMMENDATION SERVICE

The Steer recommendation service was designed to provide personalized recommendations of a variety of content to enterprise users. In this paper, we focus more narrowly on the problem of providing recommendations on ‘innovations’. Innovations refer to software applications that IBM developers have advertised through a catalog called the Technology Adoption Program (TAP). Through this catalog, they can recruit early adopters to try their innovation and receive feedback. There are around 3,000 innovations listed on the TAP site, which can get overwhelming for a user to wade through. The problem we address in this paper involves providing personalized recommendations of innovation(s) relevant to a user.

3.1 Architecture

Steer is implemented with scalability and performance in mind. Its architecture and components resemble high throughput websites and services on the internet today. Steer mainly consists of two subsystems: a service subsystem, and an asynchronous computational subsystem. The Service subsystem provides RESTful end points for data consumption insertion by other services in the enterprise. The asynchronous computational subsystem provides infrastructure to do pre-computing such as computing inferred scores and recommendations and scheduled data retrieval. Separating the two subsystems allow the whole system to scale easily.

3.2 Sources of data

Steer utilizes multiple sources of data to provide recommendations. First class data is received from different application providers related to usage of their application by their users. Within an enterprise, users across multiple applications can be identified uniquely through an enterprise directory ID or email. Thus usage can be identified across multiple applications.

Steer receives user data from 3 different services: Bookmarks; a social bookmarking application that is part of the Lotus Connections’ suite, the TAP site; and ThinkPlace site; a place for IBM employees to share and collaborate on new ideas.

Second class data (auxiliary data) is received from a couple of different sources, but used to provide recommendations to all applications when those applications do not provide enough first class data to compute proper recommendations. This data consists of information collected from the enterprise directory about users’ enterprise attributes and a set of top 30 key-terms the user is associated with. Attributes include a manager flag, normalized job title, and division/department. Key-terms associated with the user is created by crawling external sources like AlphaWorks² articles, developerWorks³ articles, IBM Redbooks⁴ and papers, patents and patent applications authored by IBM employees, and 25 IBM internal content sources (e.g. internal social software and services). Some of these sources provide direct association of key-terms to employees like tags on their internal social profile. Other sources provide implicit key-terms like tags employees have used or tags on items that they have shown interest in, and in some cases keywords used on content they have authored. Steer keeps a count of the occurrences of key-terms and these key-terms can be ordered by the number of their occurrence. For the purpose of this paper, auxiliary data will be referred to as the enterprise interest profile, and is discussed in section 3.4.2.

3.3 User data

User data, referred to as preferences, show the relationship of a user to an item and the strength of that relationship. Steer defines a user preference on an item as a vector consisting of: user-id, item-id, score, source, action, and status.

\[
p = \langle \text{user}, \text{item}, \text{score}, \text{source}, \text{action}, \text{status} \rangle
\]

where \( p \) is any given preference.

The user-id and the item-id defines the internal representations of the IBM employee and the item URI. The score defines a value given to this relation. The source attribute defines which service this preference originates from. The same item URI might belong to multiple applications. For example a TAP innovation URL might also be bookmarked in Lotus Connections Bookmarks. The action attribute further distinguishes the action done by the user to create this preference (e.g. “viewing” a webpage versus “bookmarking” it). Status defines an attribute of the item in discussion. In some cases an item should no longer be recommended to users, however preferences created with that item could still be used for similarity calculations.

Currently Steer has a total of 1,954,645 raw preferences for 175,925 users. Specifically for the TAP site, Steer has 524,463 raw preferences created by 129,926 IBM employees. However only 5994 of these preferences are ready to be used in recommendation computations as they include explicit score attribute for the innovation (item).

Explicit preferences, which contain score information, is even less common in the enterprise than on the internet. This is due to the fact that users receive minimal to no benefit from providing such ratings. However implicit data is relatively abundant and easier to associate with a user. Preferences for TAP is differentiated by the action attribute. TAP has only one explicit preference “rating” action which has a non zero score and four implicit preference actions: try, tag, comment, and bug. 97% of the TAP raw preferences are that a user has tried (“try” action) the innovation listed. Rest of the data is composed of users commenting (“comment” action), tagging (“tag” action), and reporting defects (“bug” action) on a given innovation.

To enable use of implicit preferences Steer utilizes a linear function to compute a score value. For any given innovation (item) if a preference exists with a “rating” action, the score value is used directly as the computed score. All other actions of the user on the same item are ignored. However if no explicit score is available, all preferences for the user on the innovation is used to compute a score.

\[
P(u, i) = \text{set of all } p \text{ where user}(p) = u \text{ and item}(p) = i \
\]

\[
a_n(u, i) = \begin{cases} \frac{1}{n}, & \text{if } \exists p \in P(u, i) : \text{action}(p) = \text{rating} \\ 0, & \text{else} \end{cases}
\]

\[
\text{score}_u(u, i) = \begin{cases} \text{score}(p), & \text{if } p \in P(u, i) : \text{action}(p) = \text{“rating”} \\ \sum_{n \in \text{try, tag, comment, bug}} a_n(u, i), & \text{else} \end{cases}
\]

---

where \( p \) is preferences defined in section 3.3

\[
\text{user}(p) = p[0] \text{ and item}(p) = p[1]
\]

\[
\]

\( score(u, i) \) is the computed score for user \( u \) and item \( i \)

\[
C_{tr} = 2.5 \text{ and } C_{tag} = 0.7
\]

\[
C_{comment} = 0.2 \text{ and } C_{bug} = 0.1
\]

Among preferences that include a score value; 58% have a value of 5 out of 5, 16% have 4 out of 5, and the rest are very low with very little scoring in mid ranges. It seems users rate innovations with a score when they really like it or really dislike it. Our computed scores for implicit preference fit right in the middle of the range leading us to believe that we are projecting user behavior properly.

### 3.4 Recommender algorithm

Recommendations for users are calculated in three categories: (A) Users who have enough preferences data to provide meaningful recommendations. (B) Users that don’t fall in the first category but have enough enterprise interest profile (auxiliary data) to create a user similarity metric. (C) The rest of the users that do not have any data we can use to make recommendations. For these users, we provide generic recommendations as will be discussed in section 3.4.3. In essence, category B and C represent the cold start problem and we address this by providing a graceful fallback depending on which category a user falls under. As more services are supported by Steer, user behavior across different applications can be used to identify usage patterns, an aspect we discuss in future work.

#### 3.4.1 User preferences based recommendation

It was apparent from an initial prototype that a traditional implementation of an item based recommendation algorithm would not perform to our requirement for recommendations being refreshed at a minimum once every day. Our initial approach took over three days to compute 20 recommendations each for 129,926 users on three commodity workstations.

We thus used Hadoop Map Reduce\(^5\) to distribute this recommendation algorithm. Our algorithm defines item similarity by creating an \( n \times n \) matrix \( O(n,n) \) where \( n \) is the total number of possible items. The total number of occurrences of \( \text{item}(i) \) and \( \text{item}(j) \) in user\((k)\) preferences list is added to \( O(i,j) \). This matrix defines items similarities to each other. Items that are similar should appear together in more users preferences and hence have a bigger value in the matrix.

\[
O(i,j) = \sum_{u \in \text{users}} \begin{cases} 1, & P(u,i) \neq \emptyset \land P(u,j) \neq \emptyset \\ 0, & \text{else} \end{cases}
\]

where \( O(i,j) \) is the matrix value for row \( i \) and column \( j \)

\( P(u,i) \) is as defined in section 3.3

A preferences matrix \( P(1,n) \) for a user is created where \( P(1,i) = \text{score for item}(i) \). Additionally, a masking matrix \( M(n,n) \) is created where \( M(i,i) = 0 \) if \( \text{item}(i) \) exists in users preference list, or \( \text{item}(i) \) status denotes that it should not be recommended. For all other cases \( M(i,i)=1 \). Note that the masking matrix \( M(n,n) \) is a modified identity matrix with non zero values only appearing diagonally.

The masking matrix helps remove items out of the recommendation matrix if the user has already provided a preference for it (hence seen it) or if the item can be used for similarity calculations but should not be recommended to users. Later filtering information is provided in the \textit{status} attribute of the item as discussed earlier.

\[
M_{(i,j)} = \begin{cases} 1, & i = j \land P(u,i) = \emptyset \land i \notin F \\ 0, & \text{else} \end{cases}
\]

where \( M_{(i,j)} \) is the matrix value for row \( i \) and column \( j \)

\( P(u,i) \) is as defined in section 3.3

\( F \) is a set of items that are not supposed to be recommended to the user

The recommendation matrix \( R(n,1) \) is computed by multiplying \( O(n,n) \times P(1,n) \times M(n,n) \). Cells in matrix \( R(n,1) \) having a non zero value \( R(z,1) \) are sorted and the top 20 \( \text{item}(z) \) are stored in a datastore for later retrieval through the service subsystem.

#### 3.4.2 Enterprise interest profile based recommendation

Users that do not receive any recommendations by the earlier algorithm are most likely users that are not using the application that Steer is providing recommendations for. Additionally, users’ enterprise interest profiles change slowly over time. This causes the Steer service to calculate these recommendations a lot less frequently.

Recommendations are calculated using user based CF algorithms. For the purposes of this section, key-terms associated with the users enterprise interest profile is assumed to be items, and occurrence count of key-terms is assumed to be score in classic CF terms.

Computations start with calculating user similarity. Even though the system has score values for key-terms (items), they are ignored for user similarity calculations. User similarity is calculated using following formula.

\[
T(\text{user}_i, \text{user}_j) = \frac{|P_i \cap P_j|}{\sqrt{|P_i||P_j|}}
\]

where \( T(\text{user}_i, \text{user}_j) \) is the user similarity index

\( P_i \) is the preferences set for user \( \text{user}_i \)

A set of similar users \( N_i \) is calculated using the above user similarity with \( k \)-nearest neighbor algorithm (k-NN). The recommendation vector for \text{user}_i \) is computed by iterating over every \text{item}_k \) that at least one \text{user}_j \) in \( N_i \) has preferences for but \text{user}_i \) doesn’t have preference for yet (so the item is a candidate for recommendation). \( S_n \), a running average score, for \text{item}_k \) is calculated where \( S_n \) is a running average of all \text{user}_j \’s scores in \( N_i \) who has preferences for \text{item}_k \) multiplied by \( T(\text{user}_j, \text{user}_i) \).

These recommendations are also stored in the datastore for immediate consumption by services (i.e. TAP).

#### 3.4.3 Generic recommendation

For those users that Steer couldn’t compute any personalized recommendations, the system provides a generic list of recommendations. These generic recommendations coupled with a feedback feature allows IBM employees who are new to the

service to immediately engage and start generating their own preferences. The feedback feature is discussed in section 3.6.

Generic recommendations are calculated using a time decay popularity calculation. A popularity score for each innovation (item) is calculated, and the top 20 popular items are returned as generic recommendations. The popularity score is a function of the number of votes (preferences) an innovation (item) receives, age of these votes and age of the innovation itself.

\[
P(i) = \frac{\sum (V_j \times c)}{(d + t_j)^{\alpha}}
\]

where \(P(i)\) is the popularity score for item \(i\), \(V_j\) is a raw preference for item, and \(t_j\) is time passed since that vote. \(c, d, \alpha\) are constants.

\(t_j\) is time passed since item creation

\(V_j\) is 1 for preferences for item, when action = “try”, “comment”, “bug”, “tag”

\(V_j\) is equal to score for preferences for item, when action = “rating” and score > 2.5

The computation frequency of generic recommendations depends on activity in the service Steer is providing recommendations for. In the case of TAP, generic recommendations are calculated once every hour.

3.5 Time slicing

One aspect of enterprise interest profile based recommendations is the fact that users’ interest profiles change over time. For an employee, it is not uncommon to completely switch job roles every other year, hence adversely affecting the cumulative calculation of enterprise interest profile key-terms. In return these interest profiles would potentially result in recommendations that are relevant to an employee’s earlier interest profiles.

To counteract this effect, we have utilized a time slicing algorithm while returning the top x number of key-terms where the score of items in the given time slice is significantly boosted over key-terms retrieved from content created outside of that time slice. Based on informal experiments, the time slice value for our calculations is set to two years.

3.6 Feedback on recommendations

Steer provides a way for users to give feedback on their recommendations. Services that choose to expose this functionality will have a user interface feature that allows a user to provide positive or negative feedback about the recommendations. The user interface could be as simple as a ‘more like this’ or a ‘less like this’. Feedback is especially useful for users that receive generic recommendations. This allows Steer to generate preference data for them and provide personalized recommendations.

Feedback on recommendations is incorporated back into the system in a very simple way. Positive feedback creates a preference for the user to the item with a high score while negative feedback creates a preference with a low score.

4. EVALUATION PLAN

Evaluating the effectiveness of a recommender system can be a challenge. An approach to evaluating Steer could be to randomly segment the user population into three categories and give each category of users recommendations based on only a) user preferences, b) enterprise profile data, and c) Steer recommendations. Based on user ratings of recommendations and interviews with users, we could determine how well Steer is performing against other recommendation algorithms.

5. CONCLUSION AND FUTURE WORK

In this paper we have identified an approach of utilizing user data in the enterprise in order to provide personalized recommendations for ‘innovations’ that may be of interest to a user. Our work contributes to the growing area of research on tackling cold start content recommendations for enterprise users. Our future work includes extending this service to provide recommendations of content across a wide variety of systems within IBM. Extending Steer across applications will enable us to extrapolate user behavior from one comparable system to another, as the user will remain constant across applications. We intend to conduct experiments to determine the effectiveness of Steer recommendations in the near future.

6. REFERENCES


