Meven

“An Enterprise Trust Recommender System”

By:
Usman Afzal
Md. Mustakimul Islam

Master’s Thesis at Royal Institute of Technology (KTH)
Supervisor: Nima Dokooohaki
Examiner: Prof. MihhailMatskin
Abstract

Growing an online community takes time and effort. Relationships in an online community must be initiated based on trust followed by privacy, and then carefully cultivated. People are using web based social networks more than recent past, but they always want to protect their private data from unknown access; meanwhile also eager to know more people whom they are interested. Among all other system, trust based recommenders have been one of the most used and demanding system which takes the advantage of social trust to generate more accurate predictions.

In this work we have proposed for Meven (An Enterprise trust-based profile recommendation with privacy), which uses Social Network Content (User Profiles and trends) with Trust and privacy control policy. The idea of system is to provide Social Networks with the ability to quickly find related information about the users having similar behaviors as the current user. The users will also be able to set the privacy metrics on their profiles so they will not get recommendation of those they feel less important and this is achieved by Privacy metrics. To generate accurate predictions, we defined trust between two users as a strong bond which is computed using different metrics based on user's activities with respect to different content such as blogging, writing articles, commenting, and liking along with profile information such as organization, region, interests or skills.

We have also introduced privacy metric in such a way so that users have full freedom to hide themselves from the recommendation system or they can also have the opportunity to customize their profiles to be visible to certain level of trustworthy users. We have exposed our application as a web service(api) so that any social network web portal can access the recommendations and publish them as a widget in social network.

Keywords: Recommender System, Trust Network, Social Network, Recommendation, Privacy protocol, Enterprise application.
To Our Loving Parents and all those who directly/indirectly contributed in our work.
Acknowledgments

We are honored to have worked with our examiner, Professor Mihhail Matskin, and we wish to thank him for his invaluable help and support during our work. We are deeply grateful to our supervisor, Nima Dokoohaki, who worked with us side by side and helped us with every bit of this research. He also taught us how to write a scientific text by patiently walking us through this thesis document.

Finally, we would like to thank our caring parents. Words alone cannot express the gratitude we owe them for their unlimited encouragement, love and support.
Table of Contents

Abstract .................................................................................................................................................. 2
Acknowledgments ................................................................................................................................... 4
Table of Figures ...................................................................................................................................... 8
1. Introduction ......................................................................................................................................... 9
   1.1 Meven ............................................................................................................................................ 9
   1.2 Motivation ..................................................................................................................................... 9
   1.3 Contribution ................................................................................................................................. 10
   1.4 Outline ......................................................................................................................................... 11
2. Related Work ....................................................................................................................................... 12
   2.1 Social Network Systems ............................................................................................................... 12
   2.2 Recommendation System ........................................................................................................... 12
   2.3 Content based recommendation systems .................................................................................... 12
   2.4 Collaborative Filtering (CF) Recommendation Systems ............................................................. 13
      2.4.1 Memory-based CF ................................................................................................................ 13
      2.4.2 Model-based CF .................................................................................................................... 14
      2.4.3 User-based CF ....................................................................................................................... 14
      2.4.4 Item-based CF ....................................................................................................................... 14
      2.4.5 Hybrid CF ............................................................................................................................... 15
      2.4.6 Limitation of CF ..................................................................................................................... 15
   2.5 Social networks and enterprise recommendation recommender systems .................................. 16
      2.5.1 Enterprise recommendation solution .................................................................................... 16
      2.5.2 Comparison between existing enterprise solutions ............................................................... 17
   2.6 Trust and privacy in enterprise recommendation systems ......................................................... 18
3. Meven: An Enterprise Credibility Aware Recommender .................................................................... 20
   3.1 System Specification .................................................................................................................... 20
   3.2 Functional flow of Meven ............................................................................................................. 20
   3.3 Implicit Trust ............................................................................................................................... 21
   3.4 Explicit Trust ............................................................................................................................... 21
   3.5 Privacy Protocol ......................................................................................................................... 22
   3.6 Adapters ....................................................................................................................................... 22
3.6.1 Organization Adapter..........................................................................................22
3.6.2 Regional Adapter ...............................................................................................22
3.6.3 Friends-Of-Friend Adapter ................................................................................22
3.7 Correlation functions ............................................................................................23
  3.7.1 Interest Correlation Function (Pattern Matching) .................................................23
  3.7.2 Skills Correlation Function (Tag comparison) ...................................................24
  3.7.3 Membership Correlation Function (Relational Pattern Matching) ...................24
  3.7.4 Friends Correlation Function (Shared friends; Pattern Matching) ....................25
  3.7.5 Blog Correlation Function ..............................................................................25
3.8 Explanation Module ...............................................................................................28
4 Design and Implementation .......................................................................................30
  4.1 System Architecture ............................................................................................30
  4.2 Software Design ..................................................................................................32
    4.2.1 Implementation Schema (Class diagrams) ........................................................32
  4.3 Execution (Flow diagrams) ..................................................................................38
  4.4 Data Base Design and Schema ...........................................................................40
  4.5 System Implementation (tools and applications) ..................................................41
  4.6 Screenshot of Meven ..........................................................................................42
5 Evaluation ................................................................................................................46
  5.1 Setup (Dataset) ....................................................................................................46
  5.2 Accuracy Tests .....................................................................................................47
    5.2.1 Cold -Start Accuracy .....................................................................................47
    5.2.2 Normal Flow Accuracy (When Users has friends in the system) .....................49
  5.3 Caching and Response Time Evaluation ..............................................................51
  5.4 Influence of Privacy metrics ................................................................................52
    5.4.1 Test 1 Influence of Hidden Users on recommendations ....................................52
    5.4.2 Test 2 Influence of Explicit Trust on Recommendations .................................53
  5.5 Trust Related Evaluation .....................................................................................57
    5.5.1 Test # 1 Average Trust based on Correlation Functions ..................................57
    5.5.2 Test # 2 Average Trust (cold-start) Vs Average Trust (normal flow) ...............59
  5.6 Frequency Distribution of Implicit Trust (Dynamic) ..............................................60
  5.7 Frequency Distribution of Recommendation with Explicit Trust (Static) .............61
6 Future Works ............................................................................................................62
# Table of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 1</td>
<td>Outline of the thesis paper</td>
<td>11</td>
</tr>
<tr>
<td>Figure 2</td>
<td>The content based filtering Process presented by Sony's Technology (5)</td>
<td>13</td>
</tr>
<tr>
<td>Figure 3</td>
<td>Sample of enterprise people recommender system</td>
<td>17</td>
</tr>
<tr>
<td>Figure 4</td>
<td>Explanation of recommending users.</td>
<td>29</td>
</tr>
<tr>
<td>Figure 5</td>
<td>Architectural view of Meven</td>
<td>31</td>
</tr>
<tr>
<td>Figure 6</td>
<td>Implementation schema (Data model classes)</td>
<td>32</td>
</tr>
<tr>
<td>Figure 7</td>
<td>Implementation schema (Utility Classes)</td>
<td>33</td>
</tr>
<tr>
<td>Figure 8</td>
<td>Implementation schema (Caching model)</td>
<td>34</td>
</tr>
<tr>
<td>Figure 9</td>
<td>Implementation schema (Adaptor classes)</td>
<td>35</td>
</tr>
<tr>
<td>Figure 10</td>
<td>Implementation schema (Correlation abstracts)</td>
<td>36</td>
</tr>
<tr>
<td>Figure 11</td>
<td>Overview of overall software schema</td>
<td>37</td>
</tr>
<tr>
<td>Figure 12</td>
<td>Execution of data and control flow for log on user</td>
<td>39</td>
</tr>
<tr>
<td>Figure 13</td>
<td>Database design and schema</td>
<td>40</td>
</tr>
<tr>
<td>Figure 14</td>
<td>Widget of recommendation after log on web portal</td>
<td>42</td>
</tr>
<tr>
<td>Figure 15</td>
<td>Reason of recommendation of people</td>
<td>43</td>
</tr>
<tr>
<td>Figure 16</td>
<td>Adding user to connection list with explicit trust</td>
<td>44</td>
</tr>
<tr>
<td>Figure 17</td>
<td>User setting page with privacy settings</td>
<td>45</td>
</tr>
<tr>
<td>Figure 18</td>
<td>Accuracy test based on percentage of Error</td>
<td>50</td>
</tr>
<tr>
<td>Figure 19</td>
<td>Response time evaluation in respect to number of users added</td>
<td>51</td>
</tr>
<tr>
<td>Figure 20</td>
<td>Influence of privacy metrics depending on number of hide users</td>
<td>52</td>
</tr>
<tr>
<td>Figure 21</td>
<td>Ratio of number of recommendation in respect to High explicit trust (4)</td>
<td>54</td>
</tr>
<tr>
<td>Figure 22</td>
<td>Ratio of number of recommendation in respect to Mid-level explicit trust (3,4)</td>
<td>55</td>
</tr>
<tr>
<td>Figure 23</td>
<td>Ratio of number of recommendation in respect to Low level explicit trust (1,2)</td>
<td>56</td>
</tr>
<tr>
<td>Figure 24</td>
<td>Average trust Vs number of Recommendations using individual correlation functions</td>
<td>57</td>
</tr>
<tr>
<td>Figure 25</td>
<td>Average Trust Vs number of recommendations using combination of correlation functions</td>
<td>58</td>
</tr>
<tr>
<td>Figure 26</td>
<td>Average trust and recommendations of different users in cold start</td>
<td>59</td>
</tr>
<tr>
<td>Figure 27</td>
<td>Frequency distribution of dynamic trust</td>
<td>60</td>
</tr>
<tr>
<td>Figure 28</td>
<td>Frequency distribution of recommendations with static (Explicit) trust</td>
<td>61</td>
</tr>
</tbody>
</table>
1. Introduction

1.1 Meven

Meven is a trusted expert in a particular field, who seeks to pass knowledge on to others. The word maven comes from Hebrew, via Yiddish, and means one who understands, based on an accumulation of knowledge\(^1\).

We named our system as “Meven” that understands, collects and evaluates user information (profile) based on its regional and organizational information along with its trends toward social systems (blogging, commenting etc) in order to recommend other users in the system. This recommendation takes the user privacy metrics into account for trusted recommendation generation.

1.2 Motivation

Recent work in recommendation systems includes intelligent aides for filtering and selecting information accurately. Users in any social networks, often have different personal preferences. So, recommendation system must have personalized systems that process, filter, and display available information in such a manner that suits each individual using them. The need for personalization has led to the development of systems that adapt themselves by changing their behavior based on the inferred characteristics of the user interacting with them.

In this paper, we are presenting you an “Enterprise user recommendation with trust and privacy metrics” based on sharing common friends, items, blogs etc. It helps users to find and select user profiles by generating implicit trust among different users.

“Enterprise user recommendation with trust and privacy metrics” is mainly implemented for Social network system (SNS) which consist of basic features such as user profiles, connections or contacts, blogs, articles, events, tags, categories and comments. Users who are looking for similar interest and taste in their professional life are major target users, but it can also be used in non-professional social network such as social media sites (Facebook\(^2\)).

Among many systems which have realized the impact of "Social Trust", recommender systems have been the most influential ones. Social aware (also known as Collaborative Filtering) recommenders can be configured to take into account the trust relation in-between users, in order to give better suggestions to users. As a matter of fact, users would prefer to receive recommendations from whom they trust more. Hence, recommendations are done based on the interaction given by users who are either directly trusted by the current user or indirectly trusted by another trusting user through implicit trust generation mechanism.

\(^1\)Maven.apache.org
\(^2\)http://www.facebook.com
1.3 Contribution

In our work, we proposed a mechanism named "Meven" which calculates trust between two people with both implicit and explicit ways and finally recommends set of users that system believes that are share similar tastes. The System then involves privacy matrices to provide individual with the possibility of managing their profiles being or not being used during the recommendation process. System also tries to explain the reason of recommendation. This is done through an explanation interface.

We further contributed with different adaptors, which help to generate the implicit trust value depending on different criteria such as people from same region, sharing same organization, their similar interest or skills, action on blog posts etc. Administrator of the system has the privilege to enable/ disable different adaptors according to system needs. System takes different computation time depending on/off different adaptors, which will help to evaluate the system performance. Our proposed system is capable of running during the cold start\(^3\). One of the powerful features of our system is the explicit trust value assigned between two friends. This trust value defines the most trustworthy friend of each user.

We implemented our recommendation system on a real dataset based on about 550 user profiles and more than 3000 contents of blogs posts, articles, tags, categories, groups etc. Currently, the network is running in Europe mostly in Sweden, but due to privacy policy, we cannot disclose the identity of the network members. We exposed our recommendation system as a web service so that it can be easily accessed through any other system.

Moreover, we introduced the privacy metrics in our system to make the system more secure and comfortable for the users. It also helps them to maintain their privacy in social media which is now one of the prime requirements for the internet users. For this, we have provided two privacy setting options in user profile form. One is for blocking themselves from whole recommendation process and the other one is for restricting them from different level of users.

\(^3\) [http://en.wikipedia.org/wiki/Cold_start](http://en.wikipedia.org/wiki/Cold_start)
1.4 Outline

The outline of the thesis is as follows: Chapter 2 provides the background and related work. Chapter 3 presents our approach "Meven" and our solution to maintain networks of trust for generating recommendations. We described how the presented recommendation system is designed and implemented in Chapter 4. Chapter 5 shows our experimental results and discussions. We have presented an overview of the future work in Chapter 6 and finally the conclusion in Chapter 7.

A road map is shown to present an overall view to the readers.

Figure 1: Outline of the thesis paper
2 Related Work

In this Chapter, we reviewed the related work regarding recommendation systems in context of Social networks. We will provide a brief description of related topics about our thesis. We will further present the different types of existing recommendation systems along with their benefits and limitations. We will also present some existing mechanisms on trust based recommendation and give a new picture of our ideas in trust based recommendation system.

2.1 Social Network Systems

Social Network System (1) is a platform where people connects with each other by sharing common friends, interests, likes/dislikes and exchanging knowledge. Now a days, Facebook, linkedIn⁴, twitter⁵ are some of the famous social network systems in the world. With the help of such network systems people are now able to communicate more rapidly. For example, Linkedin is one of the top professional social networks which helps many people to find job according to their expertise. Similarly, facebook and twitter are mostly used for sharing knowledge and content among friends.

2.2 Recommendation System

With such an immense amounts of information available, it is, therefore, a big challenge for the users to find out the desired contents. For this, Recommendation system is a mechanism to filter out unwanted contents for the users. Some of the famous recommendation systems are movieLens, Last.fm, StumbleUpon and so on (2). Whereas, some well-known e-commerce websites such as Amazon⁶, Ebay⁷ also use different kinds of recommendation systems to help users to find out their desired products.

There are two main types of recommendation systems, content based and collaborative filtering. We will go through different types of techniques used in Recommendation Systems in the following sections.

2.3 Content based recommendation systems

Content-based recommendation systems (3) are the systems that recommend an item to a user based upon a description of the item and a profile of the user's interests. Content-based

---

⁴ www.linkedin.com
⁵ www.twitter.com
⁶ http://www.amazon.com
⁷ http://www.ebay.com/
recommendation systems may be used in a variety of domains ranging from recommending web pages, news articles, restaurants, television programs, and items for sale. Main disadvantage of content-based recommendation system is that, user might miss the opportunity to find out new and interesting things because of “Lack of diversity” (4).

Figure 2: The content based filtering Process presented by Sony’s Technology (5)

2.4 Collaborative Filtering (CF) Recommendation Systems

Collaborative filtering (CF) is the most well-recognized and successful technique for recommendation which also overcomes the problems of content-based filtering. Collaborative filtering (CF) (6) is the process of filtering or evaluating items through the opinions of other people. CF technology brings together the opinions of large interconnected communities on the web, supporting filtering of substantial quantities of data.

2.4.1 Memory-based CF
This mechanism uses user rating data to compute similarity between users or items. This was an earlier mechanism and used in many commercial systems (7). It is easy to implement and is effective as well.
The advantages of this approach include: ability to explain the results, which is an important aspect of recommendation systems; it is easy to create and use; new data can be added easily and incrementally; it does not need to consider the content of the items being recommended; and the mechanism scales well with co-rated items.

There are several disadvantages of this approach. Firstly, it depends on human ratings. Secondly, its performance decreases when data gets sparse, which is frequent with web related items. It prevents the scalability of this approach and has problems with large datasets. Thirdly, it cannot handle new users or new items.

### 2.4.2 Model-based CF

Models are developed using data mining, machine learning algorithms to find patterns based on training data. These are used to make predictions for real data. There are many model based CF algorithms. These include Bayesian Networks (8), clustering models (8), latent semantic models such as singular value decomposition, probabilistic latent semantic analysis, Latent Dirichlet allocation (9) and Markov decision process based models (9).

There are several advantages with this paradigm. It handles the sparseness better than memory based ones. It helps with scalability with large data sets. It improves the prediction performance. Finally, it gives an intuitive rationale for the recommendations.

Disadvantage of this approach is in the expensive model building. One needs to find a tradeoff between prediction performance and scalability. A number of models have difficulty explaining the predictions.

### 2.4.3 User-based CF

The main aim of user-based CF is to identify the similar-minded users based on their similarities. If user rates an item, it finds other users who have shown interests in the same item to build user's neighborhood. Then, this user can be recommended with the items highly rated by its respective neighbors. User-based CF usually makes a prediction based on a user-item matrix.

### 2.4.4 Item-based CF

A user is likely to have the same opinion for similar items in item-based collaborative filtering. For example, if someone likes Canon digital (or may be still) cameras, he might also like Canon video cameras. Item features are used to find similarity between items i.e. how other users have rated these items.

Main advantages of item-based CF compared to user-based CF are:

- Reduces cold-start problem on new users.
• Improves scalability (similarity between items is more stable than between users, because a user might change his/her interests over time).

2.4.5 Hybrid CF

A number of applications combine the memory-based and the model-based CF algorithms. These overcome the limitations of native CF approaches. Hybrid based CF improves the prediction performance. Most importantly, it overcomes the CF problems such as sparsely and loss of information. However, they have an increased complexity and are expensive to implement.

2.4.6 Limitation of CF

Amazon uses a CF technique for recommending products, people might interest in. A recent study (10) has shown that the quality of recommendation is impacted by online retailer. Amazon usually provides good recommendations, but the quality of recommendations is impacted by several known problems of collaborative filtering systems, as well as by limitations of the underlying algorithms used by Amazon.com to enhance the system's understanding of each user. In combination with these factors, various weaknesses in the Amazon.com user interface contribute to users providing incorrect information about their preferences. This impacts the quality of recommendations, decreasing users' perception of the system's usefulness as well as their trust in the recommendations - a critical risk when competitors are only a click away.

From the case study, some of the limitations and risks are listed below:

• One of the big problems is popularity bias for example if a user likes a book that a lot of users also liked, this rating does not help the system learn much about that user (10).
• Noise can be introduced into the data by careless users (natural noise) or by users trying to promote or demote products via ratings and reviews (malicious noise).
• Data sparsity is unlikely that any two users have rated many of the same items, making it difficult to calculate the degree of similarity between users and limiting the range of recommender partner users that can be evaluated (10).
• Two users might share a similar interest in web design, but might not share the same interest in the impact of culture on web design. System has matched these users based on inadequate data.
• New users are likely to get unsatisfactory recommendations (10) because they have not provided any personal information or ratings, thus the system has no data on which to base recommendations and cannot accurately evaluate the user’s closest neighbors.
• Gift-giving: Collaborative filtering systems do not have a computational model that is capable of recognizing two distinct interests in a user's profile (10)

---

8www.amazon.com
2.5 Social networks and enterprise recommendation recommender systems

2.5.1 Enterprise recommendation solution

The Internet enables individuals to maintain existing social ties and develop new ones with the people who share similar interests (11). The emergence of the social web introduces new opportunities for people to interact and discover those with similar interests. As the users of the social web join online communities and contribute content (as in wikis and blogs) and metadata (such as tags, comments, and ratings), new ways of forming and maintaining relationships are becoming possible.

Social network systems have found that people primarily connect to individuals they already know, and are less likely to approach strangers to initiate and maintain a connection (12). SNSs have also emerged within enterprises. Research indicates that in order to stay in touch with close colleagues, employees use enterprise SNSs to reach out to employees they do not know and build stronger bonds with their weak ties. Their motivations include connecting on a personal level with more coworkers, advancing their career within the company, and campaigning for their ideas (13). The same study also recommends that “enterprise social software specifically supports users in discovering new colleagues through exploration and searching around common interests.” Most of the recommendations are based on two of the core elements of social media–people and tags. Relationship information among people, tags, and items, is collected and aggregated across different sources within the enterprise. Based on these aggregated relationships, the system recommends items related to people and tags that are related to the user. Each recommended item is accompanied by an explanation that includes the people and tags that further leads to its recommendation, as well as their relationships with the user and the item.
2.5.2 Comparison between existing enterprise solutions

Ido Guy, Sigalit Ur, Inbal Ronen, Adam Perer, Michal Jacovi (14) proposes an approach for recommending strangers in enterprise system with whom user shares similar interests. They aim at bringing new people to the user, in contrast to the exploration and search approach among their neighbors. They feel that connecting to strangers within the organization can be more valuable for employees in many ways, such as getting help or advice (15), reaching opportunities beyond those available through existing ties (16), discovering new routes for potential career development, learning about new projects and assets and so on. Compare to our works, our approach also helps end users to get recommendation from strangers based on similar attributes and taste. But people often are not interested to get friend request from the people whom they do not know at all. They mostly like to connect with friends of friends who sharing the same interest. Our enterprise solution solves this issue by finding out similarity in their organizations. Moreover, our approach

---

9 www.explow.com
also finds out some similarity where people did some action on common content such as commenting or liking on the same blog post.

Social aggregation system SAND (17) suggested a tag based recommendations, highlighting the value of tags as concise and accurate content descriptors that takes into account human perceptions of the content (18). In their approach they did not use any explicit input to the system such as rating, liking so on. But our system highly depends on explicit trust value among the neighbors which make it more reliable to the enterprise users. We also successfully deal with cold start problem with the new users (19). Tagging is normally used as a free text in most of the systems. It does not always reflect what users want. Our system also uses user-tag relationship but we provide low value for our tagging adaptor. We have high priority to users input to the system which might reduce the performance of the system but provide more accurate and reliable output to the users.

2.6 Trust and privacy in enterprise recommendation systems

Users in social network system need to express their relationships with other users which stores as an information in the system. This information leads us to the social notion of trust which helps users to find their trustworthy friends and share their preferences for an item like a movie or music. This is also due to the fact that users tend to have recommendations from their trusted partners (20).

Trust plays crucial role in many research areas such as psychology, philosophy, and sociology and computer science. It is difficult to clearly define the word “trust” as it is perceived differently by every other person. Normally we believe in something or someone based on our knowledge about them and if that belief reaches to a certain stage then it becomes Trust.

Trust has two main components: belief and commitment. The first part reflects the feeling of one towards something or someone while second part shows the bond (connectivity) towards that.

Collaborative filtering (CF) generally gives the recommendations based on similarity between users. But, similarity measurement is not sufficient enough when user profiles are sparse. The connection between how similar two users are also depends on how much they trust each other. An analysis of data in *Film Trust* (21) shows that there is a correlation between similarity and trust (22) what they read about movies, rate them and write reviews. Hence, trust can be considered as a measure for expressing the relationship between two users in recommendation systems.

Massa and Avesani presented architecture for a trust-aware recommender in which the “web of trust” is explicitly expressed by the users (23). They depicted that trust can be aggregated for all of the users in a social network, and the importance of a certain user is predicated by using a graph walking algorithm.

In our system, we defined trust between two users as a strong bond between them and this bond is computed using different metrics (as explained later) between them based on either user trends (blog writing, liking or commenting) or its profile (organization, region, interests or skills). We computed this trust as a weighted average of correlation between two users based on different
metrics that are defined below (Chapter 3). A very basic definition of our system that will explain the trust computation at a very high level is as follows:

\[ f(u, v) = \text{Avg}(\sum_{n=1}^{N} (\mu \text{Cor}(u, v))) \]

Where \( u \) and \( v \) are users between whom we find trust, “Cor\((u,v)\)” represents the correlation between users based on a specific correlation function and \( N \) represents the no of Correlation functions. In the above formula “\( \mu \)” represents the weight we give to different correlation functions. The correlations functions are explained in chapter 3.

Due to the huge exposure of personal information, now a challenge is to design effective privacy mechanisms that protect user’s information against unauthorized access to their data. Now-a-days, different social network system uses different trust models that exploit the underlying inter-entity trust information (24). The objective of designing such privacy scheme ensures a user’s online information is disclosed only to sufficiently trustworthy parties.

In our paper, we have defined our privacy protocol in such a way so that users have full freedom to block themselves from the recommendation system. It enables them to private their presence in the network. Along with that, they also have the option to customize their profiles to be visible according to the different levels of trust they define in their profile settings.
3 Meven: An Enterprise Credibility Aware Recommender

Meven, an enterprise recommendation system that is based on Content (user profiles) with Trust and privacy control policy. The idea is to provide Social Networks with the ability to quickly find related information about the users having similar attitude as the current user. The users will also be able to set the privacy matrices on their profiles so they will not get recommendation of those they feel irrelevant and this is achieved by Privacy metrics (defined below). The following section explains our system in detail along with the component description.

3.1 System Specification

This section explains overall system in detail from the conceptual perspective. As we have mentioned earlier in introduction that our system recommends users based on different correlation functions and then compute the trust as a weighted average of these correlation functions so this function also explains these correlation functions.

We will first see how “Meven” works and later we will explain each individual component.

3.2 Functional flow of Meven

Meven engine performs the recommendation when a user visits the site and log in with its account. As engine is exposed as a web service so a call is made from the portal to the engine and engine performs the following tasks.

1. The System first gets the current user profile that contains his/her regional and organizational information along with interests, skills, and blog related information.
2. The system checks either the user has any friends or not (cold-start).
3. When a User has no Friends (Cold Start) the system continues as
   a. Find users (neighborhood cluster) based on the followings
      i. Region
      ii. Occupation
4. When a User has Friends the system continues as
   a. Find the Friends of most trusted friends till a threshold of K (K=15).
      i. Repeat it until we have get the threshold value or there is no friend left
   b. If the limit of selected users does not meet the threshold find the Find users (neighborhood cluster) based on the followings
      i. Region
      ii. Occupation
   c. Combine the result of ‘a’ and ‘b’ to prepare neighborhood cluster
5. Filter profiles based on the Privacy Metrics (remove all those profiles where user has indicated not to use his profiles in recommendation)

6. Find User Correlation between current user and selected Neighborhood Clusters based on followings
   i.Memberships in similar Groups/Communities
   ii. Similar Interests
   iii. Skills
   iv. Shared Friends
   v. Similar Blogs (Blog Posts, Likes, Comments, Category, Tags)

7. Calculate Trust (Implicit) between Current user and Neighborhood Cluster Users by taking the weighted average of above computed correlations.

8. Filter every user ‘x’ from step’d’ to choose only if computed trust with the current user is equal or more than the allowed value in ‘x’.

9. Return the chosen users in a descending order with respect to computed trust.

The above description explains how meven works while recommending a list of users to the current user. In following sections we will now explain the different adapters (helpers to fetch, normalized profile information from database) and correlation functions (helper to find closeness or similarity between users based on some features).

### 3.3 Implicit Trust

"Implicit Trust" is a computed trust between user X and user Y by measuring the correlation between users. The value (between 1-4) is generated by the system taking into account different correlations between users. See section 2.6.1 for more details on how trust is computed in our system.

### 3.4 Explicit Trust

In our System we have defined Explicit Trust as a measure by which a user can explicitly define criteria of being used by recommendation system. The idea is that a user can specify to whom it should be recommended by choosing a level of trust (we call it Explicit Trust). We have following levels of Explicit Trust in our System

Level 4 = Users who want to be recommended to everyone
Level 3 = Users who want to be recommended to Majority of Users
Level 2 = Users who want to be recommended to Limited Users
Level 1 = Users who want to be recommended to more Limited Users

Explicit Trust is used by the system while recommending a user X to current user if the defined explicit trust of X is equal or greater than the trust value computed by the system (Implicit Trust).

An example of this is like "System is generating recommendations for user named James, and it computes the trust between James and Alina 2.156 (Implicit Trust). Alina has defined her Explicit
Trust value as 1 (she wants to be recommended to very limited users). The system will then exclude Alina from the list of recommendations generated for James.

3.5 Privacy Protocol

We have defined our privacy protocol in such a way so that users have full freedom to block themselves from the recommendation system. It enables them to private their presence in the network. Along with that, they also have the option to customize their profiles to be visible according to the different levels of trust they define in their profile settings.

3.6 Adapters

We define helper components for mining information from data-base as Adapter. We have used three adapters for the purpose of fetching information from the data base and these are as follows

3.6.1 Organization Adapter

During the time of Cold start (when user does not have any friends and content) or when users have fewer number of friends, the system needs to use Organization adaptor. It finds out neighborhood cluster (a cluster of users those who the system feels closer to current user) based on workplace (company or employer) of the current user.

3.6.2 Regional Adapter

During the time of Cold start (when user does not have any friends and content) or when users have fewer number of friends, the system also needs to use Regional adaptor to find another neighborhood cluster (a cluster of users those who the system feels closer to current user) based on Region (address, area or country) of the current user.

3.6.3 Friends-Of-Friend Adapter

This adapter is used when current user has friends. The adapter will start from the friends having high trust with the current user and find the friends of that friend. The system will continue building a list of users until

a) Threshold (15 users in our settings) reached.

b) Or System has traversed all friends of current user.
Following Procedure is performed for finding friends of friends of current user

\[
\text{friendsOfFriends} = []
\]

\[
\text{trustedFriends} = \text{SelectFriend(user, Trust=X)}
\]

\[
\text{Foreach (Friend f in trustedFriends)}
\]

\[
\text{If(friendsOfFriends.Length} \geq Y)
\]

\[
\text{break;}
\]

\[
\text{If(!friendsOfFriends.Contains(f))}
\]

\[
\text{friendsOfFriends.add(SelectFriend(f, Trust=X)}
\]

Where \( Y \) is the threshold value for finding friends (15 in our case) and \( X \) is the threshold of trust value to declare a friend as trusted friend (2 in our case). The helper function \( \text{SelectFriend(f, Trust=X)} \) returns the friends of user ‘f’ having trust equals to \( X \).

### 3.7 Correlation functions

We define correlation functions as functions that compute the closeness (similarity) between two users or more precisely these functions define the bond between two users. We have used the following correlation functions in our system and all these correlations functions are used once we have a neighborhood cluster either after a cold start state or normal execution where current user have friends. All correlation functions generate a correlation value between 0 to 4 where high correlation value shows the more similarity between the users.

#### 3.7.1 Interest Correlation Function (Pattern Matching)

This correlation function computes the correlation between two users based on the interest they have mentioned in their profiles. As we have Interest as choices (pre-defined) field in our system so this makes it easy to compare the interests between users. It is a simple pattern matching that we are doing in this correlation functions. The total number of matched interests are computed between two users and finally scaled between 0 and 4. The procedure for performing this is as follows

\[
\text{MatchedInterestCount} = 0
\]

\[
\text{Foreach (interest in I(u))}
\]

\[
\text{If(I(v).Contains(interest)}
\]

\[
\text{MatchedInterestCount}++;
\]

\[
\]

\[
\text{Cor}_I(u, v) = \text{Scale(MatchedInterestCount)}
\]
Where

\( Cor_I(u, v) \) is correlation between user \( u \) and \( v \)

Scale(…) is the function that normalize the integer into the range\([1,4]\)

\( I(u) \) and \( I(v) \) are interests of \( u \) and \( v \)

### 3.7.2 Skills Correlation Function (Tag comparison)

This correlation function computes the similarity between two users based on the skills they have mentioned in their profiles. As in our system the skills are declared “tags” so we have studied and applied different algorithms and finally chosen “Damerau–Levenshtein distance”\(^{10}\) for better and correct string comparison.

Since the algorithm computes the distance between two strings by adding/replacing/removing characters to match them, but in our settings we have used the algorithm in a different way and we computes distance of each tag of user ‘\( u \)’ with all tags of user ‘\( v \)’ to find the minimum distance with any tag of user ‘\( v \)’. The following explains the above

\[
\text{tagMatchedCount} = 0 \\
\text{For}-\text{each} \{ \text{U1Tag: user1.Tags} \} \\
\text{\{} \\
\quad d = \text{Min}( \text{DamerauLevenshtein} \ (\text{U1Tag}, \ \{\text{user2.Tags}\}) ) ; \\
\quad \text{U1Tag.distance} = d ; \\
\quad \text{If}(\text{U1Tag.distance}) \leq 40) \\
\quad \text{tagMatchedCount}++ ; \\
\text{\} }
\]

For each tag of ‘\( u \)’ we compute the minimum distance with all tags of ‘\( v \)’ and then if the distance is less than or equal to 40 points (60% accuracy) we take it similar tag with tags of ‘\( v \)’.

\[
Cor_I(u, v) = \mathcal{E}[u, v] \rightarrow [1, 4]
\]

### 3.7.3 Membership Correlation Function (Relational Pattern Matching)

This correlation function computes the similarity between two users based on their memberships in different groups. In our system the groups have parent child relationship that defines the semantics between groups.

\(^{10}\) en.wikipedia.org/wiki/Damerau–Levenshtein_distance
The correlation function does direct matching where it tries to find if user ‘u’ and user ‘v’ are members of similar group as well as in-direct matching where the functions tries to find the relationship between the groups and then computes the similarity between user memberships.

The relationship between the groups is maintained in a separate table that helps in finding the parent-child relationship between the groups. We use to cache this table for a specific time span for better performance.

### 3.7.4 Friends Correlation Function (Shared friends; Pattern Matching)

This is a simple correlation based on shared friends between two users. The function computes the correlation by comparing (pattern matching) friends of user ‘u’ and ‘v’ and returns the scaled value between 0 and 4. It is a simple matching of users in friend lists of two users.

### 3.7.5 Blog Correlation Function

This function computes blog correlation between two users by applying different filter functions. The idea to find the taste of two users based on their blog activities (liking, commenting, categorizing and tagging). Our goal is to come up with the correlation based on mutual taste of two users that how likely both will like to add each other to their social networks. Following model is used to compute the blog correlation between two users using different filters

\[
BCor(u, v) = \sum_{i=1}^{n} (\text{sharedBlogs} + \text{UserSelfInterestedBlogs} + \text{NeighborInterestedBlogs} + \text{similarBlogs} + \text{similarTags})/n
\]

Where

- \( BCor(u, v) \) is blog correlation between user u, and v
- sharedBlogs, UserSelfInterestedBlogs, NeighborInterestedBlogs, similarBlogs and similarTags are filters for identifying how similar trends both user have.

#### 3.7.5.1 Shared Blogs

This function calculates the number of blogs that are shared between user u and v. The idea is to find the similar taste between the users by looking onto their trends of “Liking” or “Commenting” blogs in social networks. We calculate this as follows

\[
\text{SharedBlogsCount} = \text{Blogs Liked By both Users} + \text{Blogs commented by both users}
\]

\[
\text{Cor}(u, v) = \text{Scale(SharedBlogsCount)} [1,4]
\]
3.7.5.2 UserSelfInterestedBlogs

This function computes that how likely user 'v' would be interested in current user and we do this by finding the interest of user 'v' into user 'u' by looking into its trends toward the blogs of user 'u'. The function calculates the number of blogs of user 'u' that are liked by user 'v' and then scaled the number into range of 1 to 4.

3.7.5.3 NeighborInterestedBlogs

This is a special function from the perspective of recommending someone to current user based on current user's explicit interest towards the recommended user blogs. The overall idea of the recommendation system is to figure out how likely a user will be interested into another user and the result will be highly affected if the user has explicitly shown interest into other users.

This function calculates the number of blogs of user 'v' that are liked by current user 'u' and then scale the number into range of 1 to 4. We, later, give this function more weight in our final formulation of trust between user 'u' and 'v'.

3.7.5.4 Similar Tags

Tags are extensively used in social systems for tagging object (documents, profiles etc) and are normally reused by the users. In our system users used to tag its blogs or other blogs with appropriate keyword and these are plain text as skills (discussed earlier Section 3.5.2). We used the same algorithm "Damerau–Levenshtein distance"\(^ {11}\) as for comparing the skills. The following explains the tag comparison using "Damerau–Levenshtein distance"

\[
tagMatchedCount = 0 \\
\text{For-each } \{ \text{U1Tag: user1.Tags}\} \\
\{ \\
    d = \text{Min( DamerauLevenshtein (U1Tag, \{user2.Tags\}) );} \\
    \text{U1Tag.distance} = d; \\
    \text{If} (\text{U1Tag.distance} <= 40) \\
    \text{tagMatchedCount}++; \\
\}
\]

For each tag of 'u' we compute the minimum distance with all tags of 'v' and then if the distance is less than or equal to 40 points (60% accuracy) we take it similar tag with tags of 'v'. 

The return value of this correlation is again between the range of 1 and 4 like other correlation functions.

\(^{11}\)en.wikipedia.org/wiki/Damerau–Levenshtein_distance
3.7.5.5  Similar Blogs

This is another more important function that computes the overall similarity between the blogs of two users and it mainly looks for the meta-data of the blog. In our System Blogs have categories as a meta-data field for identifying the field of interest for the Blog. The category is a choice field with parent child relationship with other categories.

The function computes the similarity between the blogs of user ‘u’ and ‘v’ in a following way

1. Two Blogs are similar if they have same title or same category (String Comparison).
2. Two Blogs b1 and b2 are similar if
   a. Category of b1 is parent of category of b2, or
   b. Category of b2 is parent of category of b1, or
   c. Category of b1 and b2 have same Parent, or
   d. Category of b1 is a child of any child of Category of b2, or
   e. Category of b2 is a child of any child of Category of b1.

The function computes the number of similar blogs as mentioned above and finally scales that into 0 to 4.

3.7.5.6  Overall Blog Correlation

We have explained each sub function in Blog correlation now we compute the overall blog correlation as a weighted average of the correlation from each individual correlation function as discussed above. The following formulation is used for the finding Blog correlation between user u and v.

\[
\text{BlogCor}(u,v) = \frac{\sum_{i=1}^{n} (\alpha \text{sharedBlogs} + \beta \text{UserSelfInterestedBlogs} + \mu \text{NeighborInterestedBlogs} + \gamma \text{similarBlogs} + \delta \text{similarTags})}{n}
\]

Where \( \mu, \beta, \) and \( \alpha \) are weights with the associated correlation.

We kept the weight of “NeighborInterestedBlogs” as the highest in our setting compared to other correlation functions and the reason is that this correlation indicates the explicit interest of current user to the recommended user.
3.8 Explanation Module

As we have mentioned that our system tracks every step during the execution of recommendation algorithm and builds the explanation of the recommendation so we have used a Recommendation module that captures the following information

1. List of Reason(s) to recommend a particular user
2. List of ReasonItem(s) associated with each Reason.

For example!

User “Albert, Johan” is recommended to the current user with the following information

Reason1: You and Albert Share Similar Region
Reason Item(s):
   Region: Stockholm (Link to Item)

Reason2: You and Albert Share Similar Tags
Reason Item(s):
   Tags: Stockholm, Social media, Information Technology, Management

Reason3: Albert is friend of your friends
Reason Item(s):
   Friends: Peter, Hina, Sam

Reason4: You and Albert have Written Similar Blogs
Reason Item(s):
   Blogs: How information flows, Ask me if you can
For achieving above results we have used our explanation module in each correlation function as well as in adapters. So we keep track of the selected users from the very start of selection of users till the recommendation is made to the current user.

Each "Reason Item" actually keeps the link to actual object (like user profile link, or blog link) for later use. Finally, the list of Reasons is attached with each recommended user for the explanation of its selection as a trusted user.
4 Design and Implementation

This section explains the design and architecture of “Meven” as well as the implementation constructs tools and applications. As we have designed our proposed solutions for mid-level organizations from 500 to 1000 users so we have kept the design of the system from scalability perspective by exposing our engine as a web service (WSDL is available in Appendix A).

4.1 System Architecture

“Meven” architecture is service oriented where the engine is exposed with its functionality as a web service and web client consumes the service for getting recommendations. The recommender engine uses the shared database of social portal (Drupal12).

The following diagram shows high level architecture of our system. The main building blocks of our system are as follows

1. Front end server containing
   a. The Portal, web client for providing interactive user interface for users (Drupal site)
2. MySQL13 Data base server having
   a. Profiling Database (Drupal standard database with some custom fields)
3. Application Server exploiting
   a. Published web service for exposing recommender engine by using the profiling database
   b. Cached recommendations and user sessions

The following architectural view of the system shows the basic building blocks of our solution as well as the network traffic between them.

12http://www.drupal.org
13http://www.mysql.com
Figure 5: Architectural view of Meven
4.2 Software Design

In this section we will explain the system implementation (Recommender Engine) from the conceptual point of view and the relationship between different artifacts of the engine. We have divided the complete design into different conceptual sections that will make it easy to understand the entity model of the system.

4.2.1 Implementation Schema (Class diagrams)

4.2.1.1 Data Model Classes

In “Meven” almost every information is encapsulated using objects implementing Standard or custom Interfaces. The following classes represent the Objects that we have used during the development of Meven. The classes also represent Interfaces that these classes have implemented. Each class is annotated with the properties and method/constructors.

![Implementation schema (Data model classes)](image)

*Figure 6: Implementation schema (Data model classes)*
4.2.1.2 Utility Classes

In “Meven” we have used some utility class also called helper classes for encapsulating standard/static or reusable information. The following diagram shows abstract level information about these classes.

*Figure 7: Implementation schema (Utility Classes)*
4.2.1.3 Caching Model

Our system’s performance heavily depends on Caching mechanism we have used at the application sever level where “Meven” is hosted as a web service. We have cached two kind of information

SQL Objects, we have cached the frequently required information from database and we kept the standard time (30 min) as the expiration time. The expiration policy is configurable.

Application (Conceptual) Objects, most frequently information is stored in objects and then cached for a certain time (30 min, configurable). For example clustered users (users from same region or organization) for a specific user are cached and reused for further recommendation.

The following model represents the schema for our caching mechanism. Each object is stored as a “Cache Item” and “MavinCache” is the class for maintaining (inserting, removing, returning) objects in cache.

![Figure 8: Implementation schema (Caching model)](image-url)
4.2.1.4 Adapter Classes

As explained earlier that we define helper classes for fetching information from database as “Adapters” so we have the following conceptual relational object model for mining information from data base. Each adapter Implements the “IGenericAdapter” that gives the opportunity of adding more adapters in future.

![Diagram of Adapter Classes](image)

*Figure 9: Implementation schema (Adaptor classes)*
4.2.1.5 Correlation Abstracts

Once the information is mined from database and stored in appropriate objects then we can apply our correlation functions to process the information to get results. We have following correlation functions in our system and each of them implements “ICorrelation” interface allowing us to introduce more correlation functions in future. Each class in the following diagram represents individual correlation function.

![Figure 10: Implementation schema (Correlation abstracts)](image-url)
4.2.1.6 Overall Software Schema

In this section, we have represented the main building objects of "Meven". The following class diagram represents overall schema of the recommender engine where "Mavin" (we use different spelling to differentiate our application constructs from the conceptual constructs) is the main class containing exposed methods to portals (for example! "Recommend ()").

"Mavin" interacts with "RecommenderManager" that is responsible of communicating with "UserClusterManager" and building list of users to process. Once user list (clustered users) is ready then it is used by "Mavin" to apply all correlation functions. The result of all correlation functions is finally processed by "Mavin" to filter users from cluster user list. The result is then sent to the calling client.

Figure 11: Overview of overall software schema
4.3 Execution (Flow diagrams)

This section explains the flow of our solution while recommending a list of users to the current user. The following flow diagram shows the data and control flow when a request t is made for recommendation for a log on user. The description is also explained in chapter 3.

1. The System first gets the current user profile that contains his/her regional and organizational information along with interests, skills, and blog related information.
2. The system checks either the user has any friends or not (cold-start).
3. When a User has no Friends (Cold Start) the system continues as
   a. Find users (neighborhood cluster) based on the followings
      i. Region
      ii. Occupation
4. When a User has Friends the system continues as
   a. Find the Friends of most trusted friends till a threshold of 15.
      i. Repeat it until we have get the threshold value or there is no friend left
   b. If the limit of selected users does not meet the threshold find the Find users (neighborhood cluster) based on the followings
      i. Region
      ii. Occupation
   c. Combine the result of ‘a’ and ‘b’ to prepare neighborhood cluster
5. Filter profiles based on the Privacy Metrics (remove all those profiles where user has indicated not to use his profiles in recommendation)
6. Find User Correlation between current user and selected Neighborhood Clusters based on followings
   i. Memberships in similar Groups/Communities
   ii. Similar Interests
   iii. Skills
   iv. Shared Friends
   v. Similar Blogs (Blog Posts, Likes, Comments, Category, Tags)
7. Calculate Trust (Implicit) between Current user and Neighborhood Cluster Users by taking the weighted average of above computed correlations.
8. Filter every user ‘x’ from step ‘d’ to choose only if computed trust with the current user is equal or more than the allowed value in ‘x’.
9. Return the chosen users in a descending order with respect to computed trust.
Cold Start
Normalized Start (when current user has friends)
Regional Adapter
Friends of Friend Adapter
Prepare User Cluster for processing
Prepare User Cluster for processing
Users Found > 15
Is Cold Start?
Organizational Adapter
Regional Adapter
Start (Call is made for recommendation)
Start (Call is made for recommendation)
End (List of Recommended Users)
End (List of Recommended Users)
Filter Profiles with Privacy Settings (Remove users that have blocked them from recommendation)
Filter Profiles with Privacy Settings (Remove users that have blocked them from recommendation)
Correlation Finder Step
Correlation Finder Step
Trust Estimation Step
Trust Estimation Step
Figure 12: Execution of data and control flow for log on user
4.4 Data Base Design and Schema

In this section we represent the database schema that is used by recommender engine. We are using the standard database schema of Drupal Portal but we are presenting only those tables that are processed during the recommendation of users. We have left the remaining standard portal tables for simplicity. The following diagram represents the relational tables that we have used in our recommender engine.

![Database design and schema diagram](image)

**Figure 13: Database design and schema**

We have used conceptual classes in recommender engine which are responsible of communicating and mining information from above tables. The information is fetched using standard SQL queries and then stored in objects for information processing.
4.5 System Implementation (tools and applications)

As we have a proposed solution to enterprise social systems so we have used standard service oriented architecture for our solution and we have a “Meven” application exposed as a web-service. The following are the tools that we used for the development of Meven.

“Meven” application is developed in C#, an object-oriented programming language from Microsoft\textsuperscript{14} that aims to combine the computing power of C++ with the programming ease of Visual Basic. C# is based on C++ and contains features similar to those of Java.

We exposed the application as a web service so that our portal can access the recommendations and publish them as a widget in social network portal.

Our Social network portal is developed in Drupal\textsuperscript{15}, written in PHP\textsuperscript{16} and distributed under the GNU General Public License source.

MySQL\textsuperscript{17} is an open source relational database management system based on the SQL is used for user management and content storage.

\textsuperscript{14}www.microsoft.com
\textsuperscript{15}www.drupal.org
\textsuperscript{16}www.php.net
\textsuperscript{17}www.mysql.com
After successfully logged in to the network, user will redirect to their profile page. 

**Meven** will find some users for him depending on their settings. User will able to see the recommended user list on the right side of their profile page with recommended user picture and name.

*Figure 14: Widget of recommendation after log on web portal*
When user click on the button "**Why**" he will get a small popup window which explaining the reason the meven find out for recommending individual user.
If user interests adding user to his connection list, he need to click on “Add” button. It will open new popup window with an option to define the trust level between the users. Meven generated trust value from Low to High (1 - 4) between recommended and logged in user will be selected by default. User always has chance to modify the trust level between them.
In user setting page, users have the option to hide their profiles from recommendation system. This is similar to “Opt-out” privacy mechanism on the web. They can define the trust level **Everyone to None (4 -1)** where everyone means they wanted to be recommend to all people in the network and More limited means they want to recommend very limited amount of people. They have also an option to hide themselves completely from the recommendation procedure by checking that single checkbox.
5 Evaluation

5.1 Setup (Dataset)

The data we have used for our system represents a social network of highly skilled business related professionals who work together to bring innovative ideas. The system contains profiles of professionals from different departments and companies and they are very much interested in knowing about others in the system. The users are mainly from sales background, management department or even the CEO’s of the companies. The profiles in the system contain information about region, organization, education, interests, skills, social activity feeds (blogs, comments, linking, and articles), and friends of a user. The system also has the ability to fetch information from other social network system like LinkedIn18.

In Meven we have used Standard data base schema of Drupal (6.0) Content Management System. The relational schema is represented in Chapter 4. There are 415 active user profiles in our system and these are the actual profiles that are being used in the system. The following represents configuration of our system that we use throughout the development and evaluation of the system.

- Total User Profiles = 415
- Users whoBlockedthemselvesfromRecommendations= 50 users
- Users who want to be recommended to Everyone (Explicit Trust19=4)= 270 users
- Users who want to be recommended to Majority of Users (Explicit Trust=3) = 100 users
- Users who want to be recommended to Limited Users (Explicit Trust=2) = 30 users
- Users who want to be recommended to Very Limited Users (Explicit Trust=1) = 15 users
- Threshold Trust for Finding friends of Friend = 2

We have used the above configuration for evaluating the performance, reliability and predictability of our System. We have divided our evaluation into following different Categories.

1. Recommendations by the System (Accuracy of the Recommendations)
2. Caching and Response time Evaluation of the System.
3. Influence of Privacy Control features on Number of Recommendation and Average Trust
4. Influence of different correlation functions on Number of Recommendation and Average Trust

18 www.linkedin.com
We have created some test user profiles for our experiments and these are as follows

1. UserId:506
2. UserId:508
3. UserId:509
4. UserId:510

The following Section explains our experiments and results for each of the above mentioned categories of evaluation.

5.2 Accuracy Tests

The Idea of the accuracy tests is to show how the quality of our recommendation results are is. As our system handle recommendations during the cold start as well as normal flow (when user has friends) so we need to perform experiments where we can show recommendation in these two stages.

5.2.1 Cold –Start Accuracy

We started with our newly created users (Cold Start State) as mentioned above (Section 5.1) and we have predicted some recommendations for them. The following table represents our manual recommendations (based on users region and education) along with the results of “Meven” during cold start state.

<table>
<thead>
<tr>
<th>UserId</th>
<th>Manual Recommendations</th>
<th>System Generated Recommendations</th>
<th>% Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>UID-506</td>
<td>439 Göran Ölander</td>
<td>439 Göran Ölander</td>
<td>0%</td>
</tr>
<tr>
<td></td>
<td>440 Elisabeth Sylve</td>
<td>441 Pernilla Arnell</td>
<td></td>
</tr>
<tr>
<td></td>
<td>441 Pernilla Arnell</td>
<td>440 Elisabeth Sylve</td>
<td></td>
</tr>
<tr>
<td>UID-508</td>
<td>1 Innovation Pioneers</td>
<td>1 Innovation Pioneers</td>
<td>13.33%</td>
</tr>
<tr>
<td></td>
<td>33 Anna Romboli</td>
<td>33 Anna Romboli</td>
<td></td>
</tr>
<tr>
<td></td>
<td>38 Annika Viktorsson</td>
<td>38 Annika Viktorsson</td>
<td></td>
</tr>
<tr>
<td></td>
<td>45 Birgitta Alm</td>
<td>45 Birgitta Alm</td>
<td></td>
</tr>
<tr>
<td></td>
<td>49 Carolina Sachs</td>
<td>51 Catharina Ottestam</td>
<td></td>
</tr>
<tr>
<td></td>
<td>51 Catharina Ottestam</td>
<td>52 Cecilia Beer</td>
<td></td>
</tr>
<tr>
<td></td>
<td>52 Cecilia Beer</td>
<td>57 Claes Tjäder</td>
<td></td>
</tr>
<tr>
<td></td>
<td>57 Claes Tjäder</td>
<td>110 Madeleine Linins-Mörner</td>
<td></td>
</tr>
<tr>
<td></td>
<td>111 Margareta</td>
<td>112 Magnus Bergmark</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>120 Maria Åstrand</td>
<td></td>
</tr>
</tbody>
</table>
We have found that our system recommends users with an accuracy of 90.89% (based on above results) in cold start state.
5.2.2 Normal Flow Accuracy (When Users has friends in the system)

After having the above results we started adding users as friends of the current user. Now the newly created users have friends in their profiles and they should get more recommendations (Non-Cold Start execution flow). We have crafted some manual recommendations to see the accuracy of our system and below tables shows the %error between our manual recommendations and recommendations generated by the system.

<table>
<thead>
<tr>
<th>User ID</th>
<th>Manual Recommendations</th>
<th>Meven Recommendations</th>
<th>% Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>UID-506</td>
<td>Lisa Lindqvist 290 Anna Schreil 34 Mikael Lindholm 132 Albert Bengtson 309 Anna Bertilsson 31 Pernilla Arnell 441</td>
<td>Lisa Lindqvist 290 Anna Schreil 34 Mikael Lindholm 132 Albert Bengtson 309 Anna Bertilsson 31</td>
<td>16.66%</td>
</tr>
<tr>
<td>UID-508</td>
<td>Bengt Järrehult 42 Tobias Vahlne 220 Joachim Cronquist 88 Emma Berglund 203 Innovation Pioneers 1 Maria Åstrand 120 Jan Sandqvist 86 Birgitta Alm 45</td>
<td>Bengt Järrehult 42 Tobias Vahlne 220 Joachim Cronquist 88 Emma Berglund 203 Innovation Pioneers 1 Maria Åstrand 120 Jan Sandqvist 86 Birgitta Alm 45</td>
<td>22.22%</td>
</tr>
<tr>
<td>UID-509</td>
<td>Mikael Ydholm 216 Maria Åstrand 120 Eva Johnson 209 Bengt Johansson 43 Jonas Klevhag 93 Peter Cullin 145 Peter Öhman 148 Sven Andrén 168</td>
<td>Mikael Ydholm 216 Maria Åstrand 120 Eva Johnson 209 Bengt Johansson 43 Jonas Klevhag 93 Peter Cullin 145 Peter Öhman 148 Sven Andrén 168</td>
<td>12.5%</td>
</tr>
</tbody>
</table>
The above results show that the % error of recommendations during the normal flow of execution and we found that the recommendation results of Meven are almost 83% correct.
5.3 Caching and Response Time Evaluation

The idea of such tests is to show the response time of the system while recommending users with and without cache. The performance is greatly affected by the configuration of the servers hosting recommendations system. We have used the following configurations for our server hosting Meven.

- Intel Core 2 due CPU
- 2.4 GHz
- 2 GB Ram, 64 bit OS

As mentioned earlier we have used caching mechanism to improve the performance of our recommendation engine. The following results show response time for user UID-506 when different no of users added to the system.

We have found that our system recommends users at an average response time of 28.2 sec while the average response time goes much lower when cache is introduced (0.6 Sec). Furthermore, we have diagnosed that most of the time taken by the recommendations is used during the mining of data from data base. We have already introduced this improvement in the future work of our project.
5.4 Influence of Privacy metrics

The idea of these tests is to show how privacy metrics effect recommendations and the average trust value. We have used the same data as above and perform the following tests for cold-start and normal flow of execution.

As we mentioned throughout that we have two kinds of privacy control features.

1. Users may hide themselves for not being used during the recommendation process.
2. Users have the possibility to set the explicit trust that defines to whom they want to get recommended.

We have performed two kind of experiments to evaluate the above two constraints.

5.4.1 Test 1 Influence of Hidden Users on recommendations

The following results represent the trend while increasing the number of Hidden users (those who blocked themselves from recommendations). The following results show the number of recommendations in both cold-start and normal flow for a particular setting of blocked users.
The above results show recommendations size (Y-axis) for different users (as above) while increasing the number of users (X-axis) who hide themselves from recommendation.

We have noticed that as we keep increasing the number of blocked users our recommendations are greatly affected. As we can see the number of recommendations for user ("UID-509") was 55 while 50 users have blocked themselves from recommendations process. This number goes to 25 when 250 users were blocked.

The results indicate that the reliability of our recommendation system is immensely dependent on the involvement of the users in the system. As more users allow their profiles to be used by the recommender the better will be the throughput.

5.4.2 Test 2 Influence of Explicit Trust on Recommendations

Now we will take the other measure of privacy control by which users can control to whom they want to get recommended to. As we mentioned earlier users can define Explicit Trust to control to which user they want to get recommended. To perform these tests we had to tweak our configurations to achieve results.

In the following tests we keep on changing the explicit trust values of the users in the system and noted the number of recommendations.
The above graph shows number of recommendations plotted for individual users where most of the users in the system specify their Explicitly Trust as (4 = Users who want to be recommended to everyone). We have found more recommendations in this case when more users in the system want to get involved in the recommendation process then the one below where number of users with mid-level trust value (3 to 4) is higher.

---

More Users with High Explicit Trust (4)

Figure 21: Ratio of number of recommendation in respect to High explicit trust (4)

Level 4 = Users who want to be recommended to everyone
Level 3 = Users who want to be recommended to Majority of Users
Level 2 = Users who want to be recommended to Limited Users
Level 1 = Users who want to be recommended to more Limited Users
Figure 22: Ratio of number of recommendation in respect to Mid-level explicit trust (3,4)

As we can notice that number of recommendations are reduced compared to when most users had explicit trust value as 4\(^2\) (Level 4 = Users who want to be recommended to everyone).

To show further dependencies we have increased the number of users who have lower Explicit Trust value and found the following results.

---

\(^2\) Level 4 = Users who want to be recommended to everyone  
Level 3 = Users who want to be recommended to Majority of Users  
Level 2 = Users who want to be recommended to Limited Users  
Level 1 = Users who want to be recommended to more Limited Users
From the above results we can see that the more users having lower explicit trust values (1, and 2\textsuperscript{22}) the lower number of recommendations are generated by the system.

This shows the direct impact of users “Explicit trust” values on the system. If more users contribute in the system by allowing to get recommended to bigger cluster of users the better the recommendations will be with improved average trust values.

\textsuperscript{22} Level 4 = Users who want to be recommended to everyone  
Level 3 = Users who want to be recommended to Majority of Users  
Level 2 = Users who want to be recommended to Limited Users  
Level 1 = Users who want to be recommended to more Limited Users
5.5 Trust Related Evaluation

5.5.1 Test # 1 Average Trust based on Correlation Functions

The idea of these tests is to show the dependencies and influence of the of different correlation functions to the generated recommendations. As we have mentioned throughout this document that we have used different correlation functions to compute Trust between two users so we have performed our tests in two different ways as follows

5.5.1.1 Average Trust using correlation functions individually

In the results below, average trust values of recommendations for user (UID=1) with individual correlation functions have shown.

![Average Trust Vs # Recommendations](image)

*Figure 24: Average trust Vs number of Recommendations using individual correlation functions*

This is noted that individual correlations functions produce high trust values when used alone. According to our analysis these recommendations have high trust values but not accurate as these include only one type of correlation between users.
5.5.1.2 Average Trust using combination of Correlation functions

As our major focus is on user’s multiple activities in social system like blogging, commenting, and liking along with the friends, interests, and skills so we believe that the recommendations are more accurate when different correlations are combined together.

In the following graph we have shown the results of average trust values produced when different correlation functions were used together.

![Average Trust Vs # Recommendations](image)

**Figure 25: Average Trust Vs number of recommendations using combination of correlation functions**

We can see that average trust values have reduced than before but these recommendations are most likely to be interesting for the current user. The green line (when all correlation functions are used) show that the average trust value remains almost the same even if we increase the number of users while the others tend to decrease when increasing number of recommended users.
5.5.2 Test # 2 Average Trust (cold-start) Vs Average Trust (normal flow)

The idea of the following tests is to differentiate the recommendations produced in two different flows of execution cold-start and normal flow. We have kept the same settings as other experiments and found the following results.

**Average Trust & Recommendations in Cold Start**

<table>
<thead>
<tr>
<th>Uid</th>
<th>Average Trust</th>
<th>Number of Recommendations</th>
</tr>
</thead>
<tbody>
<tr>
<td>506</td>
<td>1.125</td>
<td>3</td>
</tr>
<tr>
<td>508</td>
<td>1.125</td>
<td>50</td>
</tr>
<tr>
<td>509</td>
<td>1.125</td>
<td>8</td>
</tr>
<tr>
<td>510</td>
<td>1.125</td>
<td>4</td>
</tr>
</tbody>
</table>

*Figure 26: Average trust and recommendations of different users in cold start*

**Average Trust & Recommendations in Normal Flow**

<table>
<thead>
<tr>
<th>Uid</th>
<th>Average Trust</th>
<th>Number of Recommendations</th>
</tr>
</thead>
<tbody>
<tr>
<td>506</td>
<td>1.7</td>
<td>11</td>
</tr>
<tr>
<td>508</td>
<td>1.25</td>
<td>60</td>
</tr>
<tr>
<td>509</td>
<td>1.25</td>
<td>15</td>
</tr>
<tr>
<td>510</td>
<td>1.32</td>
<td>8</td>
</tr>
</tbody>
</table>

*Figure 27: Average trust and recommendations of different users in Normal flow*
The results indicate that as user starts using the system and adds more friends the recommendations become better than cold start. As we can see for user (User ID=506) the difference of the recommendations is clear for cold-start and normal flow.

The above results also indicates that the Average Trust values affected by the adding more users as friends. As user starts adding more friends into its profile, more users become available for the system to process resulting into better recommendations with improved average trust.

5.6 Frequency Distribution of Implicit Trust (Dynamic)

The above graph shows the frequency distribution of “Average Implicit Trust” generated while adding range of users (adding 10 users at a time).

We have noted the Average Trust while adding 10 users each time for UID-1 till we have added 100 new users. The newly users were having the similar Region, country for getting visible impact on Trust values.

We have found dynamic trust value generated between 1.5 and 2.5. The reason of this behavior is that we have different adaptors and correlation functions which generate this value.
5.7 Frequency Distribution of Recommendation with Explicit Trust (Static)

The above graph shows the frequency distribution of Number of Recommendations generated while adding range of users (adding 10 users at a time) and Keeping different Explicit Trust Values. (Blue bars represent “Explicit Trust 4”, Red bars represent “Explicit Trust 3”, Green bars represent “Explicit Trust 2”, and Purple bars represent “Explicit Trust 1”).

We apply this test when **UID-1** was logged on to the system. After adding 100 news users with the interval of 10 and setting the Explicit Trust value to 4, 3, 2 and 1. We have found and plotted recommendations against the Explicit Trust Values.

We have found when adding more users with lower Explicit Trust Values (keeping less involved in Recommendation) the number of recommendation were lower compared to when adding new users with high Explicit Trust Values.
6 Future Works

Currently, we have used three adapters for the purpose of fetching information from the database (organization, regional and friend of friend). We also plan to add some more adaptors such as educational adaptor. Users sometimes want to add friends from their school, high school whom they lost with the passage of life. Using educational adaptor, we could find out a pattern of similar schools and high schools or universities and programs they study together.

Meven currently focus on very basic content type in social network such as blog post or article, which has general field like keywords, categories. In future, we can plan to provide some complex content types such as events, forum to be involved to compute the trust values for our recommendation system.

We can also plan to use description field of blog post or article to find out some common pattern among different users by finding keywords in description field.

We have seen the performance of Meven is greatly affected by the mining of data from data base and we plan to focus on data mining process in the system in future for improved overall response time.
7 Conclusion

In this work, we have implemented an enterprise trust based user profile recommendation system that calculates trust value depending on user basic information, interests, skills and activities on different contents in social network. We also include privacy metrics to reduce the level of exploration of user existence from other people depending on user settings.

We introduce different caching mechanism for both frontend and backend to improve the performance of the system.

We examine our system based on a live web portal of around 500 users (due to NDA we are not allowed to write the real name) developed on Drupal (Content Management System). So, the recommendation results, we are generating from our system based on real user profiles and already workings an enterprise solution.

In our evaluation part, we have represented some histogram to graphically show our prediction and accuracy of recommendations. To justify our result, we presented our histogram based on different computed trust values in respect to different number of users. We also present the histogram on privacy metrics in respect to different size of users.
8 Bibliography


17. Social media recommendation based on people and tag. **Ido Guy, Naama Zwerdling, Inbal Ronen, David Carmel, Erel Uziel.**


24. Trust-aware Privacy Control for Social media. **Na Li, Maryam Najafian Razavi, Denis Gillet.**
9 Appendix A

9.1 WSDL

Following is the WSDL file of the Service that is hosted to expose our recommender to the web portal.

```xml
                 xmlns:org="http://schemas.xmlsoap.org/wsdl/soap12/"
                 xmlns:wsa10="http://www.w3.org/2005/08/addressing"
                 xmlns:tns="http://tempuri.org/"
                 xmlns:msc="http://schemas.microsoft.com/2003/10/Serialization/">
  <wsdl:import schemaLocation="http://localhost:4786/Service1.svc?xsd=xsd0"/>
  <wsdl:message name="IService1_AddFriend_InputMessage">
    <wsdl:part name="AddFriend" type="tns:AddFriend"/>
  </wsdl:message>
  <wsdl:message name="IService1_AddFriend_OutputMessage">
    <wsdl:part name="AddFriendResponse" type="tns:AddFriendResponse"/>
  </wsdl:message>
  <wsdl:message name="IService1_ClearMavinCache_InputMessage">
    <wsdl:part name="ClearMavinCache" type="tns:ClearMavinCache"/>
  </wsdl:message>
  <wsdl:message name="IService1_ClearMavinCache_OutputMessage">
    <wsdl:part name="ClearMavinCacheResponse" type="tns:ClearMavinCacheResponse"/>
  </wsdl:message>
  <wsdl:message name="IService1_GetRecommendations_InputMessage">
    <wsdl:part name="GetRecommendations" type="tns:GetRecommendations"/>
  </wsdl:message>
  <wsdl:message name="IService1_GetRecommendations_OutputMessage">
    <wsdl:part name="GetRecommendationsResponse" type="tns:GetRecommendationsResponse"/>
  </wsdl:message>
  <wsdl:message name="IService1_GetMessage_InputMessage">
    <wsdl:part name="GetMessage" type="tns:GetMessage"/>
  </wsdl:message>
  <wsdl:message name="IService1_GetMessage_OutputMessage">
    <wsdl:part name="GetMessageResponse" type="tns:GetMessageResponse"/>
  </wsdl:message>
  <wsdl:message name="IService1_GetMavinCache_InputMessage">
    <wsdl:part name="GetMavinCache" type="tns:GetMavinCache"/>
  </wsdl:message>
  <wsdl:message name="IService1_GetMavinCache_OutputMessage">
    <wsdl:part name="GetMavinCacheResponse" type="tns:GetMavinCacheResponse"/>
  </wsdl:message>
  <wsdl:message name="IService1_ReturnMessage_InputMessage">
    <wsdl:part name="ReturnMessage" type="tns:ReturnMessage"/>
  </wsdl:message>
  <wsdl:message name="IService1_ReturnMessage_OutputMessage">
    <wsdl:part name="ReturnMessageResponse" type="tns:ReturnMessageResponse"/>
  </wsdl:message>
  <wsdl:message name="IService1_Service1Add_MavinCache_InputMessage">
    <wsdl:part name="Service1Add_MavinCache" type="tns:Service1Add_MavinCache"/>
  </wsdl:message>
  <wsdl:message name="IService1_Service1Add_MavinCache_OutputMessage">
    <wsdl:part name="Service1Add_MavinCacheResponse" type="tns:Service1Add_MavinCacheResponse"/>
  </wsdl:message>
  <wsdl:service name="Service1">
    <wsdl:port name="Service1Soap12" binding="tns:Service1Soap12Binding" address="http://tempuri.org/">
      <wsdl:operation name="AddFriend" message="tns:IService1_AddFriend_InputMessage" output="tns:IService1_AddFriend_OutputMessage"/>
      <wsdl:operation name="ClearMavinCache" message="tns:IService1_ClearMavinCache_InputMessage" output="tns:IService1_ClearMavinCache_OutputMessage"/>
      <wsdl:operation name="GetRecommendations" message="tns:IService1_GetRecommendations_InputMessage" output="tns:IService1_GetRecommendations_OutputMessage"/>
      <wsdl:operation name="ReturnMessage" message="tns:IService1_ReturnMessage_InputMessage" output="tns:IService1_ReturnMessage_OutputMessage"/>
      <wsdl:operation name="Service1Add_MavinCache" message="tns:IService1_Service1Add_MavinCache_InputMessage" output="tns:IService1_Service1Add_MavinCache_OutputMessage"/>
    </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```
<wsdl:portType name="IService1">
  <wsdl:operation name="GetMessage">
    <wsdl:input>
      <wsdl:part name="tns:IService1_GetMessage_InputMessage" use="literal"/>
    </wsdl:input>
    <wsdl:output>
      <wsdl:part name="tns:IService1_GetMessage_OutputMessage" use="literal"/>
    </wsdl:output>
  </wsdl:operation>

  <wsdl:operation name="GetRecommendations">
    <wsdl:input>
      <wsdl:part name="tns:IService1_GetRecommendations_InputMessage" use="literal"/>
    </wsdl:input>
    <wsdl:output>
      <wsdl:part name="tns:IService1_GetRecommendations_OutputMessage" use="literal"/>
    </wsdl:output>
  </wsdl:operation>

  <wsdl:operation name="ClearMavinCache">
    <wsdl:input>
      <wsdl:part name="tns:IService1_ClearMavinCache_InputMessage" use="literal"/>
    </wsdl:input>
    <wsdl:output>
      <wsdl:part name="tns:IService1_ClearMavinCache_OutputMessage" use="literal"/>
    </wsdl:output>
  </wsdl:operation>

  <wsdl:operation name="AddFriend">
    <wsdl:input>
      <wsdl:part name="tns:IService1_AddFriend_InputMessage" use="literal"/>
    </wsdl:input>
    <wsdl:output>
      <wsdl:part name="tns:IService1_AddFriend_OutputMessage" use="literal"/>
    </wsdl:output>
  </wsdl:operation>
</wsdl:portType>
9.2 Messages

The message envelopes are as follows

9.3 Get Recommendation Service Request

```xml
<soapenv:Envelope
  <soapenv:Header/>
  <soapenv:Body>
    <tem:GetRecommendations/>
    <!--Optional:-->
    <tem:currentUserId>506</tem:currentUserId>
  </soapenv:Body>
</soapenv:Envelope>
```

9.4 Get Recommendation Service Response

```xml
<s:Envelope
xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
  <s:Body>
    <GetRecommendationsResponse
xmlns="http://tempuri.org/">
      <GetRecommendationsResult
xmlns:i="http://www.w3.org/2001/XMLSchema-instance">
        <a:GenericUser>
          <!--Optional:-->4
          <a:ExplicitTrust>1.125</a:ExplicitTrust>
          <a:ImplicitTrust>4</a:ImplicitTrust>
          <a:Organization>Microsoft AB</a:Organization>
          <a:RecommendationReason>
            <a:Reason>
              <a:ReasonItem>
                <a:Descriton>Same Friend</a:Descriton>
                <a:Link>201</a:Link>
                <a:Title>takim Islam</a:Title>
              </a:ReasonItem>
            </a:Reason>
          </a:RecommendationReason>
          <a:Items>
            <a:ReasonItem>
              <a:Descriton>You are friends of friend.</a:Descriton>
              <a:Link>201</a:Link>
              <a:Title>Kista</a:Title>
            </a:ReasonItem>
          </a:Items>
        </a:GenericUser>
      </GetRecommendationsResult>
    </GetRecommendationsResponse>
</s:Body>
</s:Envelope>
```
<a:Title>You share the same Organization.</a:Title>
</a:Reason>
</a:RecommendationReason>
<a:Region>Sweden</a:Region>
<a:State/>
<a:UserId>441</a:UserId>
<a:UserName>PernillaArnell</a:UserName>
<a:GenericUser/>
</GetRecommendationsResult>
</GetRecommendationsResponse>
</s:Body>
</s:Envelope>

9.5 Clear Meven Cache Request

  <soapenv:Header/>
  <soapenv:Body>
    <tem:ClearMavinCache/>
  </soapenv:Body>
</soapenv:Envelope>

9.6 Clear Meven Cache Response

<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
  <s:Body>
    <ClearMavinCacheResponse xmlns="http://tempuri.org/"/>
  </s:Body>
</s:Envelope>