Journal of Universal Computer Science, vol. 22, no. 3 (2016), 319-339
submitted: 13/10/15, accepted: 23/1/16, appeared: 1/3/16 © J.UCS

A Domain Ontology in Social Networks for Identifying User Interest for Personalized Recommendations

Rung-Ching Chen
(Chaoyang University of Technology, Taichung, Taiwan
crching@cyut.edu.tw)

Hendry
(Chaoyang University of Technology, Taichung, Taiwan
Hendry.honk@gmail.com)

Chung-Yi Huang
(Chaoyang University of Technology, Taichung, Taiwan
chungyi410@gmail.com)

Abstract: Social media and the development of web 2.0 encourage the user to participate more interactively in social networks. In social network relationships may be identified by the user posts and interactions. Using this data, the system can make recommendations tailored to specific users. However, when the user is on social network for the first time, the recommendation system cannot make recommendations, since the user has no history. In this paper, we design an ontology combined with social networks. We develop the ontology based on data from users and their friends. Using the user interest and community influences, we propose a system to solve the cold start problem in recommendation systems. The system calculates the similarity between users, using user preferences and uses a rule generating algorithm to create the dynamic inference rule. The ontology is updated each time the content of the personal ontology is updated. The newest ontology will be retained to increase the accuracy the next time the recommendation system is executed.

Keywords: Personalized Recommendation, ontology, social network analysis
Categories: H.3.3

1 Introduction

Social media stores vast quantities of data on its users. The development of web 2.0 encourages user participation in social networks such as Facebook, Twitter, and Google+. Users communicate with each other, post their activities, upload their images, and give their personal data to the social media [Kaplan, 10]. Social media is also a place to perform business, offering advertisements, e-commerce, and game markets [Merrill, 11].

The size of data available in social media offers opportunities for researchers to discover knowledge in it. One clue to knowledge about users is user interest [Qian, 13], the preferences of the users in posting, liking, and sharing data in social media. This could be used to offer the users products such as advertisements [Liu, 11], applications, or games.

In social networks, relationships are defined by posts and likes made by users. Each post or like on other user's walls can affect user interest. Users who make many influences on other users we call *central users*, while users who make the fewest influences on other users we called *followers*.

Currently, the recommendation system is defined by the way users give scores or rating on the website. Personal recommendation systems have been developed to enhance the user's experience on the website and are same on both computers and mobile devices. They are designed to enable the user obtain a greater number of exploration choices when seeking movies, music, or books online.

For companies seeking to use this information, a major problem is the cold start user, a user using the services for the first time, with no record of history of likes or posts. Since personal recommendation systems cannot process the user without information, the social network can address this problem via the activity of the user's friends [Juang, 05]. Recommendation systems are also challenged by the ubiquity of mobile devices, since recommendations must be made in an environment where calculation is more limited.

In recent years, recommendation system is gradually escaping the traditional single recommendation. Many studies begin to investigate the recommendation system which combines semantic web reasoning mechanism. The related technologies such as ontology are used to achieve intelligent recommendation. Through combine mobile devices, so digital content can be transmitted more easily for the users. In the ontology inference developing process, some recent studies have focused on the establishment of personal ontology. By way of the user's properties and using records, user's personal ontology can be produced. The ontology can be adjusted at any time by using the record contents to provide a better service for users.

In this paper, we propose a novel method to improve personalized recommendation systems for mobile device users, applying a combination of social network data, real time communication software, and ontology. We collected the data from the social network Facebook. We crawl the user's friend list data and construct a personalized ontology using real-time communication software. Finally, we use an inference engine to generate the results for the user. We use social networks to solve the cold start problem. Thus, whenever users login for the first time, the system can make recommendations for the user based on the hobbies of their community.

2 Related Work

In the following, we briefly review methodology relevant to this paper, including previous work on ontology, inference engines, and personal recommendation systems.

2.1 Ontology

Ontology is a formal, explicit specification and supports the concept of sharing [Studer, 98]. Its advantage is its ability to process information in open, distributed, heterogeneous, and weakly structured environments [Quan, 06]. Ontology also could use to automatic-semi automatic generation of the service agreement, which recommend some user about the services they want to use [Mariam, 14]. Ontology uses variables needed for a set of computations and establishes the relationship between them. Ontology consists of vocabulary and terminology. Vocabularies define the concepts and relationships used to describe and represent an area of concern.

There are several programs for editing ontology such as Protégé and OntoEdit, but the process is tedious. Much research focuses on developing ontologies automatically by integrating them with knowledge acquisitions and machine learning technology. Technologies used to improve the capability of ontologies include Natural Language Processing (NLP), association rule mining, and hierarchical clustering.

2.2 Inference Engine

An inference engine is a tool that is able to infer logical consequences from a set of facts or axioms. Technologies such as RuleML, forward chaining, backward chaining, and *Rete Algorithm* are applied in inference engines.

RuleML defines a rule as a mark-up language, based on XML and tree structure [Boley, 06]. In RuleML the rule must continue to accumulate, and each rule may only be used once. For each new case or facts, the engine must add a new rule. RuleML stores its knowledge in a tree, where the root element is the label and the node element is the data. Researchers have also used object-oriented methodology that transforms RuleML into object oriented RuleML (OO-RuleML). The advantages of OO-RuleML are that it allows object-oriented sets, making the inheritance easier for unordered arguments sequences, along with Uniform Resource Identifier (URI) grounding and order-sortedness [Boley, 03].

JESS is a rule engine developed for Java platform by Ernest Friedman Hill. JESS adopts *Rete Algorithm* to strengthen the efficiency of its inference [Friedman, 03]. JESS is not a procedural algorithm, where only a single program is activated in one loop. Instead, JESS applies the declarative paradigm, continuously applying a collection of rules to a collection of facts. This process is called pattern matching. Rules modify the facts or execute Java code.

Jena is an inference engine developed in Java language that provides an inference subsystem. A range of inference engines or reasoners may be plugged into Jena [Jena, 11]. Jena software resources are open source and its interface is free. It may be used and modified freely. The Jena API enables system development to more easily cooperate with an ontology.

2.3 Personal Recommendation System

Much research focuses on personalized recommendation systems. Xueming et al. [Xueming, 14] proposed a method to calculate a recommendation system based on user interests from a user in a social circle. They used the inferred trust from social circles to recommend items. Ahn et al. [Ahn, 12] proposed that personalized recommendation systems could be based on the familiarity of the user with the same topic in a social network. This approach considers how the user likes and comments on Facebook posts based on the relationship of the user. Fong et al. [Quan, 06] proposed methods to calculate emotional intensity within a certain time period. This requires a camera to capture the human face and calculate its expression. Mobile device have limitations in processing the calculation. Social network data is semi unstructured data, hard to process within the limitations of mobile device processors.

[Juliana, 06] proposed a method of personal knowledge recommendation system, using agents, learning ontologies and web mining. The system achieved the improvement of the agents to learn knowledge faster by motivated it to create a recommendation chain between the users. [Fong, 12] proposed a method to generate personal ontology based on consumer's emotion and behavior analysis. The system improves the recommendation through emotion between users based on fuzzy logic. Fuzzy could handle with uncertainty data. All the proposed methods are needed complex calculation time and are not suitable for mobile devices processing.

In this study, we use Ontology Web Language (OWL) to establish an ontology, and solve the semi unstructured dataset using the ontology. We develop a personal ontology and user interest ontology that consists of a movie, music, and book ontology. These ontologies will record individual information from Facebook pages, and the movie, music, and book pages they comment on or like. We use combinational ontology to build the relationship between their friends in social networks. Mobile devices obtain the results from the ontology reasoners as soon as the web server processes the inference between user interests in the social network. We use AndroidJena's API as an inference engine to process the results and present the results to the mobile device.

3 Research Method

This section describes the operation of our proposed method. We describe the architecture of our system and develop an algorithm to generate the dynamic rule, acceptance rate, and to calculate the similarity rank.

3.1 Architecture of the System

The system is developed in mobile device interface. Since we collect data that requires user permission, the user must log in to their social network – in this case Facebook – and give the system the privilege to access their data. The system will crawl their friend list, basic profile (name, gender, age), and the pages about the movie, music, and book categories that they follow. The data is later sent to our web server to process and construct the relationship. We also perform pre-processing to repair and modify missing data or data not relevant to our research.

In the next step the system constructs the ontology for each person and category from the dataset. The ontology file is sent to Jena to perform the inference and obtain the results. Finally the user will receive movie, music, and book recommendations displayed on their mobile devices. The system workflow is shown in Figure 1 and is as follows: (1) the system is started using an Android app. From this app the system collects the user data from the user's social network friends. Users need to login to the system and grant the system privileges to access their data in Facebook. The system then crawls the data of the user, and collects the movie, music, and book category that is liked, commented on, or followed by the user. The system will also collect the user's basic profile data such as name, gender and age. (2) The system stores the data in a web server database, perform pre-processing to modify missing data and data irrelevant to the research methodology.

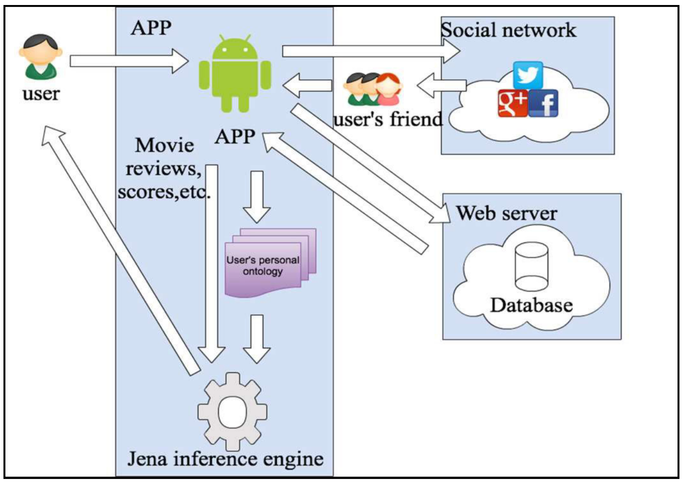
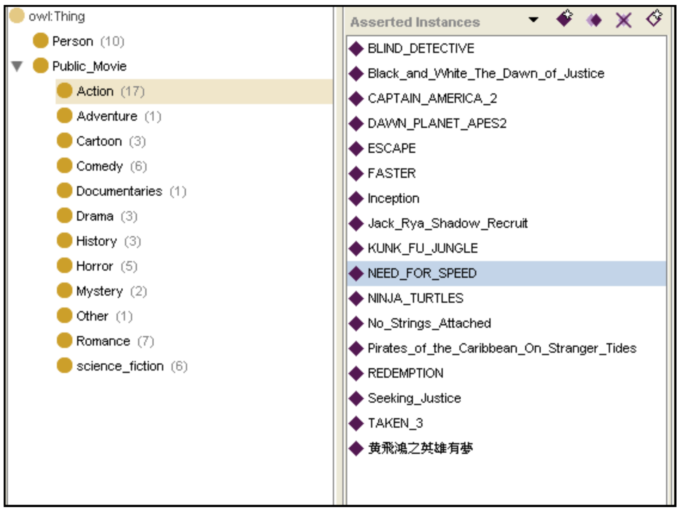


Figure 1: System Architecture

The system crawls for missing data such as the genre of movies, music, or books. (3) We define an OWL to establish the ontologies and use Protégé tools to develop the ontology. Figure 2 shows the ontologies we use in this research. (4) The system will load the ontology on to the mobile device and perform inferences to generate the results of the recommendation list. A candidate recommendation list will be generated by a decision making algorithm. It generates rules from inference rules.


owl:Thing
Person (10)
Public_Movie
Action (17)
Adventure (1)
Cartoon (3)
Comedy (6)
Documentaries (1)
Drama (3)
History (3)
Horror (5)
Mystery (2)
Other (1)
Romance (7)
science_fiction (6)

Asserted Instances
BLIND_DETECTIVE
Black_and_White_The_Dawn_of_Justice
CAPTAIN_AMERICA_2
DAWN_PLANET_APES2
ESCAPE
FASTER
Inception
Jack_Rya_Shadow_Recruit
KUNK_FU_JUNGLE
NEED_FOR_SPEED
NINJA_TURTLES
No_Strings_Attached
Pirates_of_the_Caribbean_On_Stranger_Tides
REDEMPTION
Seeking_Justice
TAKEN_3
黄飛鴻之英雄有夢

Figure 2: The Ontology Structure

3.2 Ontology Generation

The ontology is unique for every object of a domain to describe or to be conceptualized. We build 4 ontologies for our research: personal ontology, movie ontology, music ontology, and book ontology. The movie, music, and book ontology have the same structure and differ only on attributes. Figure 3 shows our ontology for the movie category. In the Figure 3, the movie ontology includes Romance, Action, Drama, Comedy, Horror, and Science Fiction. Each movie category has individual movie's title. Class Person listed the users that have watched or like movie page from Facebook. In the next section, based on this ontology we want to recommend movies for a cold start user from the relation between the user preferences that already collected from the inference reasoning.
We used Protégé tools to develop the ontology, then the Jena API framework to build the program to process the ontology. We describe movies as a class that has sub-classes. We take genre as our concern, and classify the genre according to our dataset. Each genre has an *individual* of movies. We focus only on genre in classifying the movie. Code program 1 shows part of the ontology classified by genre.

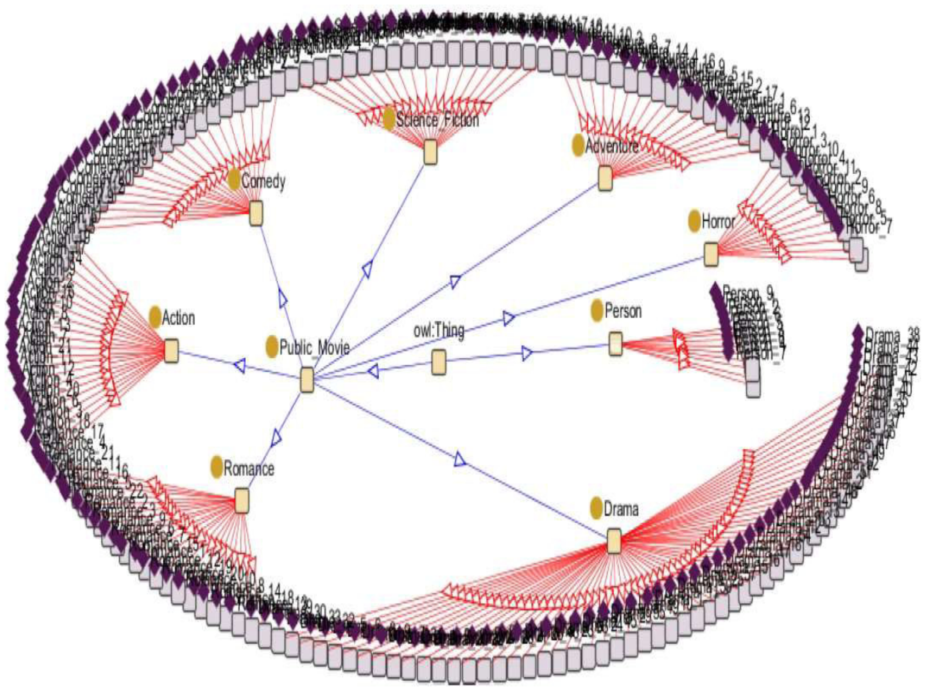


Figure 3: Movie Ontology

Code Program 1: Movie Ontology Classification by Genre

```
<owl:ObjectProperty rdf:about="&movieontology;belongsToGenre">
  <rdfs:range rdf:resource="&movieontology;Genre"/>
  <rdfs:domain rdf:resource="&www;Movie"/>
</owl:ObjectProperty>
```

3.3 Acceptance Rate

The Acceptance Rate (AR) is a threshold calculated from normalization of the user preferences [Kwona, 11]. The system collects the user preferences from each user. The data are collected are then classified into each genre of the category. In this section we use the movie category as the example. The music and book category have the same steps in calculating the acceptance rate of the user.

Class	Comedy	Adventure	Action	Romance	Drama	Horror	Science Fiction
User1	8	11	23	0	3	0	18
User2	5	8	4	5	3	15	0
User3	8	4	14	4	6	28	0
User4	0	4	22	5	16	3	1
User5	8	16	3	0	4	21	9

Table 1: User Preferences

Class	Comedy	Adventure	Action	Romance	Drama	Horror	Science Fiction
User1	0	0	1	0	0	0	1
User2	0	1	0	0	0	1	0
User3	0	0	1	0	0	1	0
User4	0	0	1	0	1	0	0
User5	0	1	0	0	0	1	0

Table 2: Normalized User Preferences

We could not combine the genre of movie, music and book category into one, because of the differences in their attributes. The three categories have the same attributes only for genre, which is one reason we chose genre as our classification method. Table 1 shows an example of user preference values in our research.

We next perform the normalization using the threshold, to decide whether the user accepts the services. First we set up a threshold value θ. When the acceptance rate value is greater than θ the value of Normalization Point (NP) result is 1, and if the value is lower than θ the NP value is 0. Equation 1 shows the normalization process, while code program 2 shows the algorithm of normalization

Code Program 2: Normalization Algorithm

```
Input: User Preference
Output: Normalization User Preference
Initialization: Threshold Value
Normalization (Preference)
Begin
        max_accept_rate←max(Preference)
        min_accept_rate←min(Preference)
        For i to Preference_Length
                Normal_i←(Preference(i) - min_accept_rate)
                Normal_i←Norma_i / (max_accept_rate - min_accept_rate)
                If (Normal_i>θ)
                        Normal_preference_i←1
                Else
                        Normal_preference_i←0
                End If
        Return Normal_Preference
End.
```

.

$$NP = \begin{cases} 0, & Normalization(\text{AR}) < \theta \\ 1, & Normalization(\text{AR}) \geq \theta \end{cases} \tag{1}$$

Setting the threshold value θ to 0.5, we take user1 as an example. User1 has a maximum acceptance rate of 23 and a minimum acceptance rate of 0. From this information the system will calculate the normalized user preference for each genre of User1. For the comedy genre: 8/23 = 0.3478. Since the value AR for the Comedy Genre is lower than the Threshold θ, we set NP=0.

Table 2 shows normalized user preference of the acceptance rate for each user. The system performs this calculation identically for other categories. Music and books also will be normalized using the same method.

3.4 Similarity Rank

After the system calculates all the normalized user preferences AR, the next step is to obtain the candidate users that have the same interests as the user. We use the XOR formulation equation to search for such candidate users. We use equation 2 to compare the entire user's similarity rate. M is defined as the number of classes, while m is the index of the class. The smaller the Similarity Rate (SR), the closest the user is to the candidate users' preferences. The process calculates the XOR equation:

$$SR_{i,j} = \sum_{m=1}^{M} XOR(NP_{im}, NP_{jm}) \tag{2}$$

The system will sort the entire value of SR, in equation 3, where *P* is an integer value that stands for the number of users. The SR_Ranking is the set of order of candidates recommended in order by the value of $SR_{i,j}$

$$SR\_Rank_i = \{SR_{i,1}, SR_{i,2}, \dots, SR_{i,p}\} \tag{3}$$

Code Program 3: Similarity Rank Algorithm

```
Input: User Preference
Output: Similarity Count
Similarity(User, Member)
Begin
        User_Preference ← Select from User
        Member_Preference ← Select from Member
        Normalize_User ← Normalization (User)
        Normalize_Member ← Normalization (Member)
        For i in Every Member m
            Similar_Count ← ∑XOR(Normalize_User_{i,m},Normalize_Member_{i,m})
        Return Similar_Count
End.
```

Code program 3 is used to calculate the similarity rank. We develop a candidate decision algorithm to classify the community members into candidate rank based on the similarity candidate decision and the normalization method

We classify the user into one of three types of candidates. The first candidate has a similarity value equal to that of the user, the second candidate has a similarity value equal to 1. All others are classified into the third class of candidate. Code program 4 shows the candidate decision algorithm

3.5 Recommendation List

This sub-section will explain our method to generate a dynamic rule to give the recommendation to users who have the highest similarity rank. This step begins with finding the preferences of the user. These preferences consist of long-term and short-term preferences. Long-term preferences are defined as watching a movie, listening to music, and reading a book for a long period of time, while short-term preferences mean that the user is watching a movie, listening to music, or reading a book at the moment. There are other factors that affect user preferences such as user interest influences, which are their activities in social networks. What the community likes might be affecting the way the user makes decisions. The system can use this information to make the inference engine to decide what rule could be generated from the process.

Code Program 4: Candidate Decision Algorithm

```
Input: User Preference
Output: Candidate Decision List
Candidate_Decision
Begin
        User←rand(member_list)
        Remove(user, member_list)
        While (member_list!=NULL)
                Member_?←rand(member_list)
                Similar_count←Similarity(User, Member_?)
                If (Similar_count=0)
                        First_candidate_list ← Member_?
                Else If (Similar_count=1)
                        Second_candidate_list← Member_?
                Else
                        Third_candidate_list ← Member_?
        Return  First_candidate_list, Second_candidate_list,
                  Third_candidate_list
End.
```

Code Program 5: Recommendation List Algorithm

```
Input: Short_list, Long_List, Else_List
Output: Recommendation List
Initialization: Short_list_Size, Long_List_Size, Else_List_Size
Recommend
Begin
        Synchronization_Server_Update(Personal_Ontology)
        Reasoner.load(Personal_Ontology)
        If (Short_preference!=NULL)
              Candidate_decision()
        Reasoner (Short_List_Size, First_Candidate_List,
                Short_Preference, Recommend_Short_List)
        Reasoner (Long_List_Size, Second_Candidate_List,
                Long_Preference, Recommend_Long_List)
        Reasoner (Else_List_Size,Third_Candidate_List,
                Long_Preference, Recommend_Long_List)
End.
```

Commonly ontology and inference engine are static and could not be changed. So the results generated from this also fix. In this study, we try to find the way to modify these disadvantages, so ontology and inference engine could reuse the knowledge and make them update their ontologies automatically. We proposed the dynamic rule algorithm to update the ontologies after the system and user get a feedback. However the rules that are generated automatically might be irrelevant to the user, especially when cold start user without previous knowledge is using the system. Code program 5 shows our dynamic rule generation algorithm.

Code Program 6 Reasoner Algorithm

```
Input: Short_list, Long_List, Else_List
Output: Recommend_List
Initialization: Short_list_Size, Long_List_Size, Else_List_Size
Reasoner(recommend_size, candidate_list, preference, recommend_list)
Begin
        While (Recommend_sort_list<recommend_size)
              If (Fail>Fail_count)
                    Break
              If (candidate_list!=NULL)
                    Candidate←rand(candidate_list)
                    Services←Inference(Candidate, preference)
                    Remove(Candidate, candidate_list)
              If (Services!=NULL)
                    Recommend_list.add(services)
              Else
                    Fail++
End.
```

Code Program 7: Inference Algorithm

```
Input: candidate_list, preference
Output: Services
Inference(candidate_list, preference)
Begin
        Candidate ← rand(Candidate_list)
        Rule ← "(User Like preference)(candidate like preference)
              (service belong preference) → Recommend(User, Services)"
        Services ← Reasoner.inference(Rule)!=NULL
        Return Services
End.
```

The dynamic generation rule algorithm consists of a reasoner and inference sub program. The system compares the personal ontology with its candidate list. Code program 6 shows the process of the reasoners. The Inference algorithm is used to enable the inference engine to generate a rule for the recommendations for the candidate list. Its return services are used by the reasoner. Code program 7 shows the inference algorithm. After the reasoner step is finished, the results of the recommendation list are presented to the user.

Figure 4 shows the processes of the dynamic generate rule algorithm process to recommend a movie for the user based on the user's interests. Each user that has same interests can influence other users to choose the same movie. This study uses the relationships between users with social media data, personalized ontology, and an inference engine to generate rule used for recommendation system.

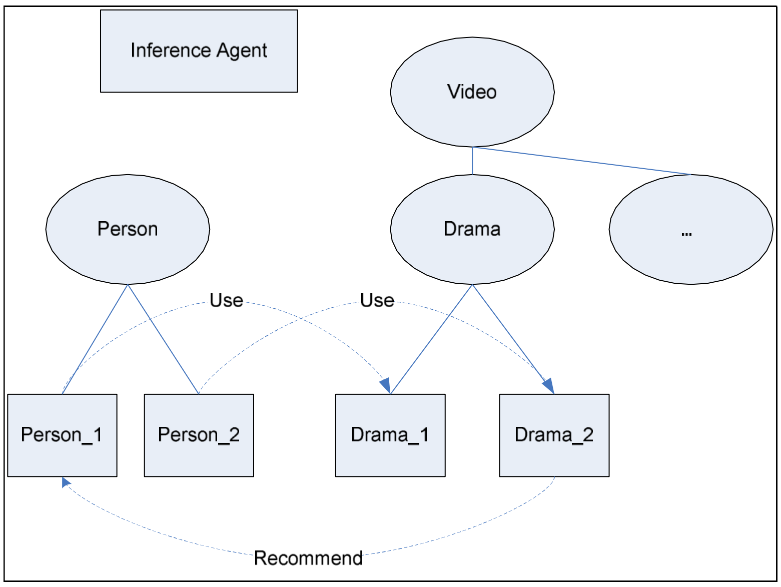


Figure 4: Recommendation System Process

In Figure 4, Person_1 likes to watch the movies with Drama_1 category; Person_2 likes to watch the movies with Drama_2 category. Then, the recommendation process calculates similarity of Person_1 and Person_2 users' preferences. If the two users' preferences are similar, according the calculation of similarity rank algorithm, Drama_2 is recommended to the Person_1.

4 Experiments

In this section, we conduct a series of experiments to evaluate the performance of our proposed method. We use a dataset from Facebook pages such as movies, music, and book categories that are followed by our Facebook user. Note that we modify the dataset to match our experimental method and construct the relationships of the dataset.

4.1 Dataset

We conduct the experiments with the data from social network Facebook. This system uses FacebookAPI to collect information such as movie, music, and book pages on Facebook. We collected data from 100 users as a training dataset and then validate the system with 30 users as the testing dataset. Table 3 shows the information about the data that the system collects from Facebook. In practice, since social network data has a scalability problem, we collected the data based on the relationship between users who have voted for movie, music and book pages only. Participant datasets collected include their basic profiles, their explicit interests (such as movie, music and book pages), and content of social activities. Social activities on Facebook about movies, music and book pages means: (1) the pages on Facebook they like and (2) the Facebook pages they comment on.

We define the dataset as a combination of user interest in movies, music, and book pages. These categories are already defined on Facebook. The dataset consists of likes of movie pages divided into 83 genres, music pages (26 genres), and book pages (22 genres). The average number of comments and likes is around 20 for each genre of movies, 30 for each genre of music, and 10 for each genre of books. Table 4 shows the field names and data format of the dataset.

User	**Movie**		**Music**		**Book**	
130	Total movie	261	Total music	887	Total book	90
	Total genre	83	Total genre	26	Total genre	22
	Total comment	8,921,437	Total comment	74,186,225	Total comment	3,361,022

Table 3: Dataset Collected from Facebook

Tb_user_tree		**Tb_movie**	
Field	**Type**	**Field**	**Type**
Id	Varchar (20)	Id_movie	Varchar (50)
Name	Varchar(100)	Title	Varchar (50)
Parent_id	Varchar(20)	Genre	Varchar (100)
Level	Float	Release_date	Date
Weight	Float	Movie_artist	Varchar (100)
Quantity	Float	Studio	Varchar (50)
Node_weight(NW)	Float	Director	Varchar (50)
Node_quantity(NQ)	Float	Cover	Varchar (255)
NW_multiply_NQ	Float	Talking_about_count	Int (11)
Edge_quantity(EQ)	Float		
NW_multiply_EQ	Float		
Similarity_weight	Float		

Table 4: Data Format of Data Set

Figure 5 shows that the top three genres for movies are **Drama/Romance**, **Drama,** and **Animation**. Note that we manually fix the missing data for the genre, according to the genre we collected from the producers which we crawled from the web. Each users comment or like on movie pages we count it as frequency based on the movie genre. We just add the frequency by 1, whether the user comment frequently, like and post in the movie pages.

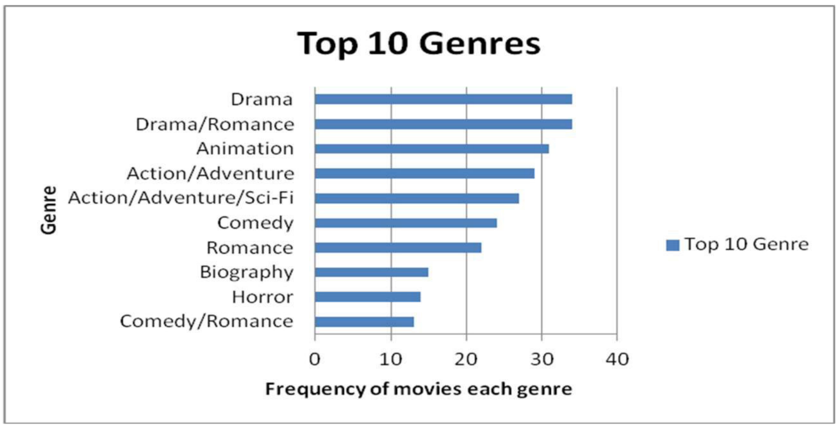


Figure 5: Top 10 movies genres

Figure 6 shows that the top three genres for music are **POP**, **Acoustic/Rock/Pop,** and **Alternative/Rock**. We manually fixed the missing data for the genre using information obtained by crawling the web. Each users comment or like on music pages we count it as frequency based on the music genre. We just add the frequency by 1, whether the user comment frequently, like and post in the music pages.

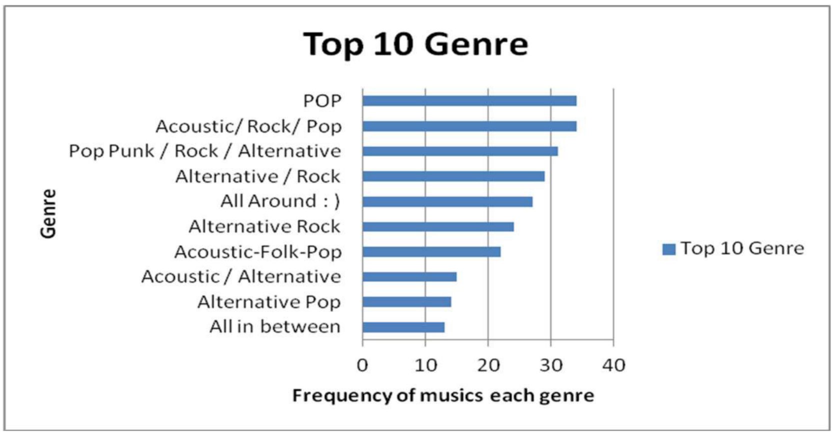


Figure 6: Top 10 of music genres

Figure 7 shows that the top three genres for books are **fantasy**, **drama,** and **comedy**. We manually fixed the missing data for the genre using information obtained by crawling the web. Each users comment or like on book pages we count it as frequency based on the book genre. We just add the frequency by 1, whether the user comment frequently, like and post in the book pages.

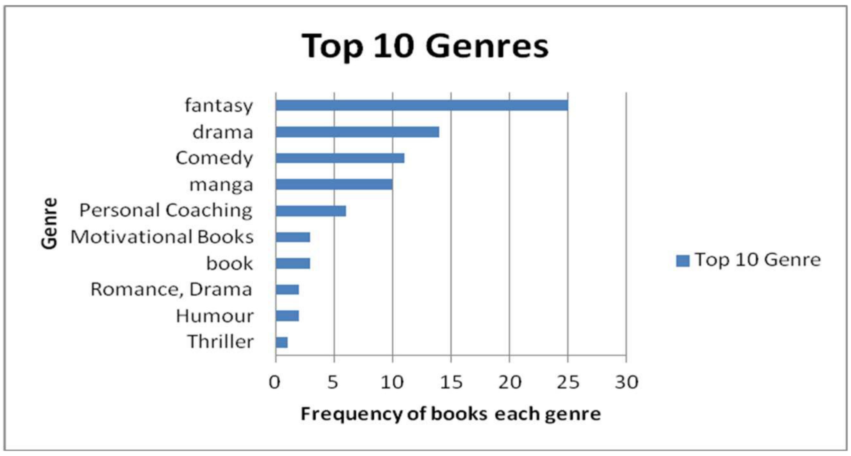


Figure 7: Top 10 book genres

4.2 Performance Measure

We conducted the experiments using 100 users as training data, with 30 users as the testing dataset. We collected all the movie, music, and book data from 100 users as training data for the system learning. The evaluation metric we used in our experiments is Mean Absolute Error (MAE), as this is a popular metric for measuring the accuracy in the recommendation system [Jiang, 12]. MAE is defined as follows:

$$MAE = \frac{\sum(m,i) \in M \left| R_{m,i} - \hat{R}_{m,i} \right|}{|M|} \tag{4}$$

Where $R_{m,i}$ is a real item from user m on item i. $\hat{R}_{m,i}$ is the predicted recommendation item generated by the system. M is the set of all user items in the test set.

4.3 Evaluation

4.3.1 Parameter Setting

This section focuses on the setting of the acceptance rate threshold value use for the parameters of our algorithm in our model. We set the parameter θ value from 0.1 to 0.9 and evaluate the accuracy of the results. We compare the results of each category from the first candidate to the third candidate to obtain the best threshold value for our objective.

4.3.2 Results Comparison

In this section, we compare the results from our experiments for each of our categories. We compare the results from first candidate list through the third candidate list. Table 5 lists the results for the movie category. The first candidate list has the best accuracy based on MAE, because the recommendations are generated using similar user interest. The best result for the first candidate list in the movie category is for θ = 0.6, with MAE = 4.98. The second candidate list shows the best results when θ is 0.8, with MAE = 16.63. Best results for the third candidate list are when θ = 0.9 with MAE = 56.11. Note that the first candidate list has the highest similarity value and the smallest MAE value.

Threshold θ	C1 MAE	C2 MAE	C3 MAE
0.1	5.65	31.37	59.6
0.2	4.99	17.84	61.67
0.3	5.80	17.50	59.66
0.4	5.88	17.54	61.67
0.5	5.78	19.47	61.675
0.6	**4.98**	19.76	61.673
0.7	5.40	17.48	61.672
0.8	6.43	**16.63**	60.66
0.9	5.85	21.48	**56.11**

Table 5: Accuracy Comparison between Threshold Values in Movie Category

The results for the music category are shown in Table 6. For all values of θ in first candidate list, MAE = 16.44. The second candidate list shows the best results when evaluation = 0.5, with MAE = 45.55, while the third candidate list shows the best results when θ = 0.8, with MAE = 50.87. Since all threshold values have the same MAE in the first candidate list, which has the highest similarity value, the θ that is suitable for this category is the average of all candidate lists. Its value is 0.5 with an average MAE of 39.97.

Threshold θ	C1 MAE	C2 MAE	C3 MAE
0.1	16.44	79.90	54.91
0.2	16.44	79.90	52.89
0.3	16.44	79.90	58.95
0.4	16.44	69.79	68.04
0.5	**16.44**	**45.55**	57.94
0.6	16.44	69.79	54.91
0.7	16.44	68.78	65.16
0.8	16.44	72.82	**50.87**
0.9	16.44	66.76	67.03

Table 6: Accuracy Comparison between Threshold Values in Music Category

Results for the book category are shown in Table 7. The best threshold value with the lowest MAE is 0.9, with an MAE value of 18.76 for the first candidate list. For the second candidate list, the best threshold value is 0.9 with a MAE value of 45.49. For the third candidate list, the best threshold value is 0.3 with a MAE value of 52.36.

Threshold θ	C1 MAE	C2 MAE	C3 MAE
0.1	20.86	66.76	61.78
0.2	20.87	66.75	57.37
0.3	20.88	66.79	**52.36**
0.4	20.88	68.81	58.49
0.5	20.87	66.79	61.78
0.6	20.88	68.81	55.21
0.7	20.86	54.59	61.78
0.8	20.87	45.60	61.78
0.9	**18.76**	**45.49**	61.78

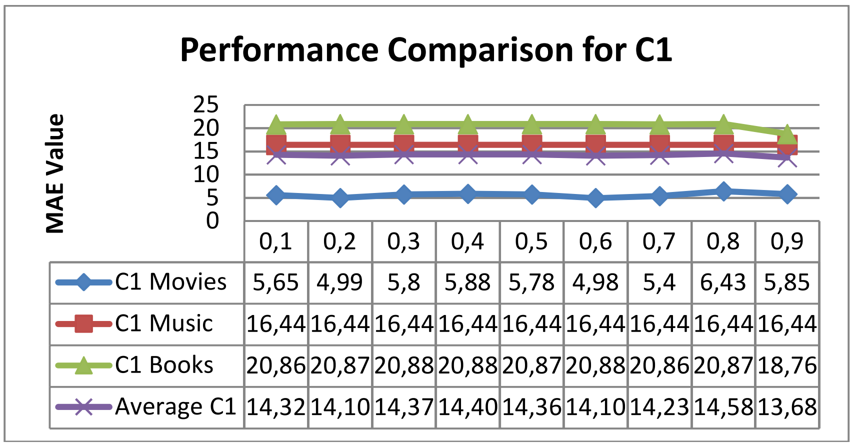
Table 7: Accuracy Comparison between Threshold Values in Music Category

4.4 Discussion

Comparing the results for each category, we discuss how to obtain the best threshold value θ in our proposed method. Figure 7-9 show the comparison for each candidate list in each category. We compare the initialization of the threshold value with the MAE value, and trade-off to determine the best value. We explain for each candidate list as follows.

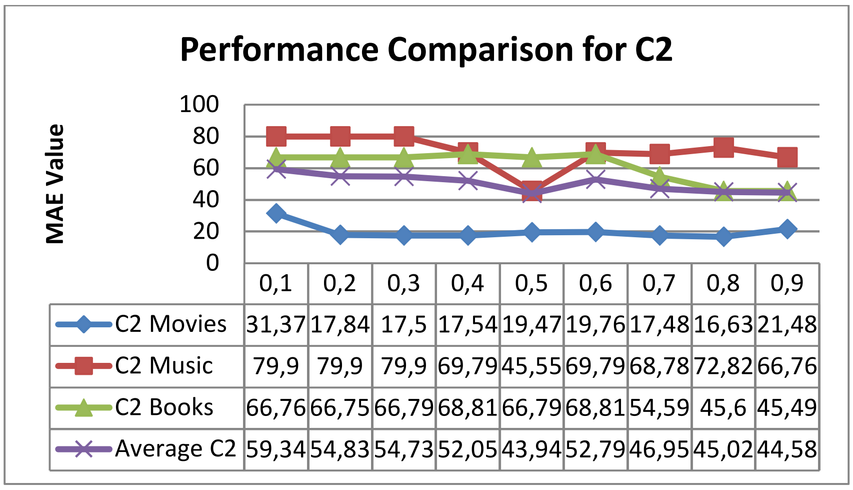
We compare the results for the first candidate list for the movie, music, and book categories. The threshold is set from 0.1 to 0.9. The average MAE value is calculated by the system in order to search for the best threshold value. Figure 8 shows that the first candidate list for movies has the best value, followed by the music and book categories. The best threshold for C1 movies is 0.6, for C1 music it is any value, and for C1 books it is 0.9. The best threshold for all C1 is 0.9 with a MAE value of 13.68

Figure 9 shows that for the second candidate list of movies have the best value, followed by the music and book categories. The music category is better than the book category only when the threshold value is 0.5, otherwise the MAE value for C2 music is lower than that of C2 books. The best threshold for C2 movies is 0.8, for C2 music it is 0.5, and for C2 books, 0.9. The best threshold for all C2 is 0.5 with an MAE value of 43.937



	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9
◆ C1 Movies	5,65	4,99	5,8	5,88	5,78	4,98	5,4	6,43	5,85
■ C1 Music	16,44	16,44	16,44	16,44	16,44	16,44	16,44	16,44	16,44
▲ C1 Books	20,86	20,87	20,88	20,88	20,87	20,88	20,86	20,87	18,76
× Average C1	14,32	14,10	14,37	14,40	14,36	14,10	14,23	14,58	13,68

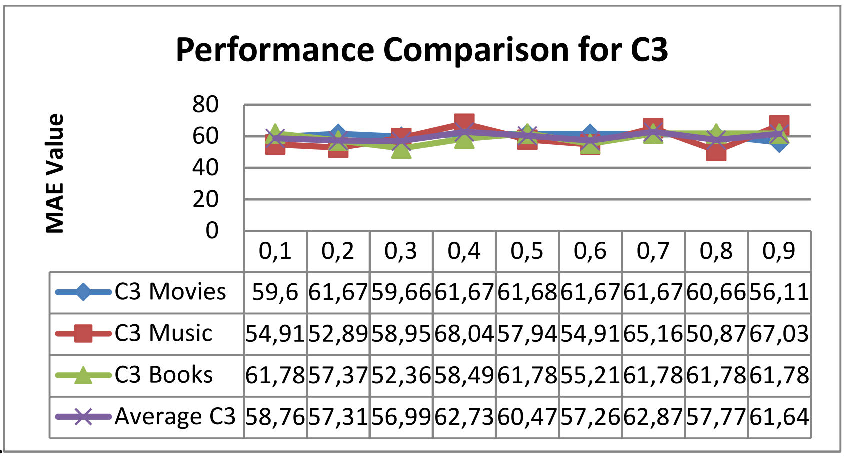
Figure 8: Performance Comparison of the First Candidate List



	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9
◆ C2 Movies	31,37	17,84	17,5	17,54	19,47	19,76	17,48	16,63	21,48
■ C2 Music	79,9	79,9	79,9	69,79	45,55	69,79	68,78	72,82	66,76
▲ C2 Books	66,76	66,75	66,79	68,81	66,79	68,81	54,59	45,6	45,49
× Average C2	59,34	54,83	54,73	52,05	43,94	52,79	46,95	45,02	44,58

Figure 9: Performance Comparison for the Second Candidate List

Figure 10 shows that for the third candidate list the music category have the best value, followed by books and movies. The best threshold for C3 movies is 0.9, for C3 music it is 0.8, and for C3 books, 0.3. The best threshold for all C3 is 0.3, with a MAE value of 56.99.



	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9
C3 Movies	59,6	61,67	59,66	61,67	61,68	61,67	61,67	60,66	56,11
C3 Music	54,91	52,89	58,95	68,04	57,94	54,91	65,16	50,87	67,03
C3 Books	61,78	57,37	52,36	58,49	61,78	55,21	61,78	61,78	61,78
Average C3	58,76	57,31	56,99	62,73	60,47	57,26	62,87	57,77	61,64

.

Figure 10: Performance Comparison for Second Candidate List

5 Conclusions

This research combined ontology, social networks, and similarity method to calculate the recommendation list and address the cold start problem for new users. Personalized ontology is used to retain information from the user. The system is developed to generate a dynamic rule from the inference engine so that the information can be updated automatically and can be saved in the personal ontology. The results of the recommendation list are classified into first, second, and third candidate lists. In the first candidate list a member has characteristics which have the highest similarity to those of the user. From the experiments we found that our best system accuracy for the movie category is 95.02% (MAE = 4.98%) when the threshold value is 0.6. For the music category it is 83.56% (MAE = 16.44%) when the threshold value is 0.5, and for book category it is 81.24% (MAE = 18.76%) when the threshold value is 0.9. Total accuracy of the recommendation list generated by the system is 86.32% (MAE = 13.68%) when the threshold value is 0.9. Note that our main concern is the first candidate list only, based on the greatest similarity between users.

6 Future Work

In the near future, we will explore external factors such as the popularity of the item on social media. Popularity means how often the item is commented on or liked. We will explore whether this parameter has a significant effect on recommendation system accuracy.

Acknowledgements

This paper is supported by Ministry of Science, Taiwan, with project number: MOST-103-2221-E-324-028 and MOST-104-2221-E-324 -019 -MY2

References

[Ahn, 12]Ahn, D., Kim, T., Hyun, S.J., and Lee, D. (2012). Inferring user interest using familiarity and topic similarity with social neighbors in facebook. Proceedings International Conferences on Web Intelligence and Intelligent Agent Technology, IEEE/WIC/ACM.

[Boley, 03] Boley, H. (2003). Object-oriented ruleml: user-level roles, URI-grounded clauses, and order-sorted terms. Rules and Rule Markup Languages for the Semantic Web, Lecture Notes in Computer Science, Springer Berlin Heidelberg.

[Boley, 06] Boley, H. (2006). The ruleml family of web rule languages. Proceedings of the 4th International Conferences on Principles and Practice of Semantic Web Reasoning (PPSWR'06).

[Friedman, 03] Friedman-Hill, E.(2003). Jess in action: rule-based systems in Java, Manning.

[Fong, 12] Fong, A.C.M., Baoyou Zhou, Siu C. Hui. (2012). Generation of Personalized Ontology Based on Consumer Emotion and Behavior Analysis. IEEE Transactions on Affective Computing, 3(2): 152-164.

[Jena, 11] Jena. (2011). https://jena.apache.org/documentation/inference/

[Jiang, 12] Jiang, M. (2012). Social contextual recommendation. Proceedings of 21st ACM International CIKM, New York, USA.

[Juang, 05] Juang, C.H. (2005). Collaborative filtering recommendation Groups. National Central University Department of Information Management Master's Thesis, Taiwan.

[Juliana, 06] Juliana Lucas de Rezendel, Vinicios Batista Pereira, Geraldo Xexeo, Jano Moreira de Souza. (2006). Building a personal knowledge recommendation system using agents, learning ontologies and web mining. Proceedings of the 10th International Conference on Computer Supported Cooperative Work in Design.

[Kaplan, 2010] Kaplan, A. M., Haenlein, M. (2010). Users of the world, unite! The challenges and opportunities of social media. Business horizons 53(1): 59-68.

[Kwona, 11] Kwona, O., Leea, Y. and D. Sarangibc. (2011). A Galois lattice approach to a context Aware privacy negotiation service. Expert Systems with Applications, 38(10):12619-12629.

[Liu, 11] Liu, K., Tang, L. (2011). Large-scale behavioral targeting with a social twist. Proceedings of the 20th ACM International Conference on Information and Knowledge Management.

[Mariam, 14] Mariam Rady. (2014). Generating an excerpt of a service level agreement from a formal definition of non-functional aspects using OWL. Journal of Universal Computer Science, 20(3):366-384.

[Merrill, 11] Merril, T., Latham, K., Seantelesa R., Navetta, D. (2011). Social media: the business benefits may be enormous, but can the risks -- reputational, legal, operational -- be mitigated? http://www.acegroup.com/us-en/assets/ace-progress-report-social-media.pdf.

[Qian, 13] Qian, X., Liu, X., Zheng, C., Y. Du, and Hou, X. (2013). Tagging photos using users vocabularies". Neurocomputing, (111):144-153.

[Quan, 06] Quan, T.T., Hui, S.C., Fong, A.C.M. and Cao, T.H. (2006). Automatic fuzzy ontology generation for semantic web. IEEE Trans. Knowledge and Data Eng, 18(6): 842-856.

[Studer, 98] Studer, R., Benjamins, V. R., and Fensei, D. (1998). Knowledge engineering: principles and methods. *Data & Knowledge Engineering*, 25(1-2): 161-197.

[Xueming, 14] Xueming, Q., He, F., Guoshuai, Z., and Tao, M. (2014). Personalized Recommendation Combining User Interest and Social Circle, IEEE Transactions on Knowledge and Data Engineering, 26(7): 1763-1777.