An Ontological with Clustering approach for Content Based Recommendation Systems

ABSTRACT
The paper proposes a novel method to build a recommendation system based on feature vectors and ontological graphs with dynamic weighted ranking. With an ontological knowledge base and similarity measures this recommendation system builds an adaptive ranking mechanism based on user selections and preferences. Based on user feedback the ranking of entities in the same result-set changes over time and for calculating this variation two methods based on clustering techniques have been proposed. For testing these ideas, we built a tool for recipe recommendation called RecipeMiner. For most cases, the results were coherent with the expected values and the tool gave efficient and useful recommendations.

Author Keywords
Recommendation, ontology, weighted ranking, dynamic graph, feature vectors

ACM Classification Keywords
H.3.3 Information Search and Retrieval, H.5.2 User Interfaces

INTRODUCTION
A recommender system is an extension to regular search systems and takes a step forward in incorporating selection and adaptation methods based on user preferences and feedback. A major step in extending the solution of search to recommendation is the methodology of making intelligent decisions by using semantic information and considering ontological behavior. Using case based reasoning an ontological tree can be traversed in a decisive manner and starting from the root, subsequent nodes can be identified by taking into account preferences in the order of ontological importance. In existing implementation of recommender systems using semantic knowledge a major drawback is that the system fails to adapt to changes in user preference and has no way to dynamically increase the accuracy of the system: repeated input preferences give almost, and often exactly, the same recommendations each time.

The method described in this work is aimed at areas where it is possible to segment the knowledge base into groups and identify ways to order the entities within a specific group. Also, a predefined method of identifying inter-group relatedness is necessary. Such an environment also gives the added advantage to revise the ranking mechanism at any given point and make appropriate changes based on feedback because the attributes of the system are properties of individual groups. In this manner dynamism can be attained by subjective reordering of entities in selected groups.

Unlike collaborative filtering recommender systems [19], [22], [23], [26], hybrid recommendation [7] and content based collaborative recommendation systems such as 'Fab' [14] where data is collected from different users with the assumption that people with similar tastes will rate things similarly, the idea in this work is to cultivate the knowledge base for a specific user and use it to predict future selections for a specific user or a group of similar users. The work of M. Svensson et al. [17], [16] uses social markers in the realm of food recipes and focuses on recommendation which depends on the user as well the actions of peers. However in our work we focus on individual user preferences and in case of anomalies the rules of the knowledge base are suitably altered so as to best cater to future recommendations (for the specific user). This system does not interact with other similar systems running in the local environment and the possibility of contamination of user data by external factors is eliminated.

The paper is structured as follows: The next section describes the relevant work in the direction of ontological recommendation and user modeling. Subsequent sections describe the proposed method for recommendation and the experiment conducted by building a recipe recommendation system, RecipeMiner, as a test-bed for the proposed algorithms. Remaining sections explain the results obtained with respect to various constraints in the recipe recommendation system and the possible improvements and future work. Finally, the last section summarizes the salient features of the proposed method and concludes the paper.

RELATED WORK

Ontological approaches

In this section, we discuss related work which utilizes ontological knowledge for recommendation. ‘JaDaCook’ [21] is a recipe recommendation tool that utilizes ontological knowledge of food items to build an ontological tree and traverses the tree using Case Based Reasoning. It is based on the assumption that if one of the ingredients requested by the user is too generic, it will be specified (getting all their children) using the ontology created for this purpose. ‘A Wine and Food Pairing Recommendation System [9]’ provides appropriate wine recommendations based upon user inputs regarding the type of meal and the ethnicity of the recipe. Such tools necessitate the acceptance of constraints in a sequential manner which is in the form of a series of questions asked to the user. A drawback of this tree-based approach is that the path from the root to the child has to be followed and each level has to be traversed. In our work we used a graph based approach where recommendation can start from any parent and nodes have multiple parents. This gives the flexibility of having search criteria with variable number of elements and in which the order is insignificant. This also allows us to give an efficient recommendation even when all the categorizations are not given by the user. In such cases the sub-graph described by given categorical attributes is considered and the contained nodes are forwarded to the next layer for processing.

In "Overcoming incomplete user models in recommendation systems via an ontology’ [27] it is assumed that the user preferences follow an ontology. However, in our work, we used ontology to make meaningful groups in our dataset with the idea that ontological relatedness can be used to recommend items to the user in addition to those satisfying the search criteria.

Aimed at enhancing traditional recommendation the work in [6] incorporates profiling and identification modules to existing approaches. The work mentioned in [25] gives adequate importance to ontological inference and shows that it improves user profiling and other aspects or recommendation. Our work here aims to go beyond user profiling and adapt to changes in the user preferences.

User modeling and personalization

Earlier work on web based recommender systems incorporating user modeling and personalization can be witnessed in projects such as NewsDude [4] and Personal WebWatcher [5] which filter user's web-based content and provide recommendation based on personalized information using k-Nearest Neighbors and TF-IDF machine learning techniques. Example sets for both, preferred and not-preferred content are maintained for each user. ‘Semantically Enhanced User Modeling’ [18] enhances the traditional term-based user model with the WordNet-based semantic similarity techniques. Works such as ‘Ontology-Based User Modeling for Knowledge Management Systems’[12],’Semantic Modeling of User Interests Based on Cross-Folksonomy Analysis’[15] and ‘Challenges and benefits of the semantic web for user modeling’ [20] also employ semantic relatedness to user modeling.

Our work here is focused at adding the element of dynamic adaptation to ontologically created user models. This ensures personalization with time (or number of runs) which converts the search based recommendation data into a fully personalized knowledge base which can be used for recommending. The addition of dynamism allows the system to cater to the needs of the user more effectively and provides a richer and more fulfilling experience.

ONTOLOGICAL WITH CLUSTERING APPROACH

Basic Idea

The work described in this paper aims at two important aspects of recommendation systems: adaptation and ranking. The ranking of recommended results is based on semantic similarity and interests of user. The use of feature vectors and adaptive weight assignment ensure each of these respectively. The assumption is that user selections will follow a pattern and given the same set of entities, the user will tend to select a specific subset most of the time. The important issues needed to be addressed in this regard are:

- The system should offer acceptable recommendation for choices of various preferences, varying immensely with individuals.
- The system should be able to adapt to changes in individual user selections and preferences.
- Users should be provided with suitable recommendation even in the starting phase when not much training has been possible

The first aspect, namely, ‘providing efficient recommendation’ is addressed by the use of ontological graphs and sub-graphs and relying on the knowledge of semantic relationship between entities. This provides for an effective classification of data into meaningful sets where entities are connected by semantic laws and enable the recommendation algorithm to work powerfully on filtered data. Ontological relationships can be effectively drawn on data where categorization is possible and the different categories are also inter-related by certain semantic relationships. Thus, each entity in the set has a unique place for itself and is always connected to the remaining entities by at least one link.

The second aspect which deals with the change in user preferences envisages the need to push recommendation to a level where dynamism is possible. An ardent lover of history might also want to read novels at leisure. For such a user, the recommendation system should give higher ranking to history books but also be able to cope with situations when the user selects 'novel' or 'leisure reading' as a sub-category. In our work, we try to work around this problem by building a ranking mechanism over an ontological graph which deals with the alterations in user preference and helps in adapting to changes.

The third aspect, i.e. good initial recommendation, is conveniently handled by the use of attribute based feature vectors for different nodes. These feature vectors are then used to find similarity between nodes and final recommendation results contain such similar nodes in addition to the search results.

Proposed Method

The method proposed in this work tries to solve the problem of introducing adaptation in the existing recommendation approaches. Because of adaption, the results shown to a user for a specific query may change the next time the user searches for the same query because the entities finally selected by the user from the search results may introduce new information in the knowledge base regarding the preferences of the user. An example of this could be a web recommendation system where a user searches for websites and repeatedly goes only to those sites which allow the user to download a video. For subsequent results, the recommender system can be adaptive and give higher rank to such websites. Similar approach is followed in our experiments with food recipe data-sets where recipes similar to those finally selected by the user from the search results were ranked higher in subsequent recommendation results for the same queries.

To build such a system we require one more layer over the usual recommendation process. Hence, unlike earlier work utilizing single layered approaches, the recommendation method proposed here depends on two layers of relationships, categorization and ranking:

- Semantic Layer; and
- Dynamic Ranking Layer

The semantic layer is a static classification mechanism for various entities (recipes, books, shopping items, etc.) which can present themselves as the final recommendation for the user. Incorporating ideas from internet page ranking methodologies, 'The anatomy of a large-scale hyper-textual web search engine'[24] and variable and feature selection[10] this layer also builds the feature vectors required for calculating the multi-dimensional distance measures between various entities. The dynamic ranking layer is a rearrangement of the objects derived from the first layer (the semantic layer) and is dynamic and adaptive to changes in various selection criteria and feedback. Both these layers are described in the following sections.

The ontological database (Semantic layer)

The semantic layer is responsible for categorization of data into usable subsets, creating relationships among the elements within a subset and creating relationships between the subsets. The resultant of this layer is an ontological graph which depicts the relationship of each element with every other element in the knowledge base and provides a path of traversal from the parent nodes (the categories) to the root nodes (the elements in consideration). The mechanism of building this database is described below.

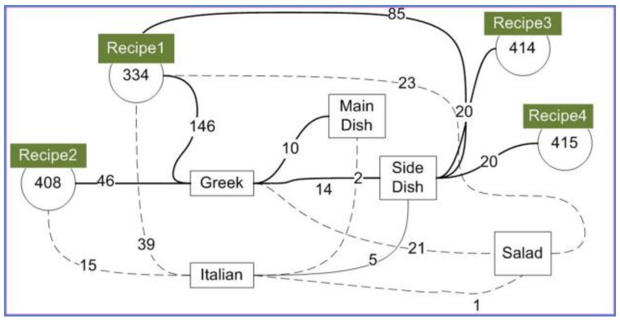


Figure 1. Graph depicting selection of Greek main dishes and Greek side dishes

Data sources

- WordNet [3] provides the semantic relationship, neighbors, parents and children nodes of a large number of entities. Synsets are interlinked by means of conceptual-semantic and lexical relations. These synsets as well as their super-ordinates can be used to classify elements of a dataset into meaningful groups. Thus, Wordnet helps in the generation of directed acyclic graphs (DAGs) which can form the basis of the ontological layer in discussion.
- Wikipedia is an excellent source for extracting ontological information and semantic relatedness between entities. Projects such as Wikitology [28] and WikiWalk [8] have used this knowledge and provided usable ideas for harvesting Wikipedia information. Internet dumps of Wikipedia are also available which can be used as a database for querying information.
- Several other sources for ontological knowledge acquisition are available on the internet. Experienced personnel, cooks and professionals have also been consulted in this regard.

For our experimental tool 'RecipeMiner' we classified the data using semantic relationships based on common sense, WordNet and through cooks whom we knew personally. For example, the Wordnet search results of 'Raita' show that its hypernym (superordinate) is 'side dish' and its synset (set of ontologically related terms) contains 'Mushy Peas'. Hence, both 'Raita' and 'Mushy Peas', if contained in our recipe list, can be categorized under the course 'side dishes'. Such data for 7104 recipes was collected and the recipes were categorized in terms of the following:

- Ethnicity
- Course
- Type of meal
- Time required for cooking
- Calorie content.

Building the Ontological graph

Once the dataset was identified and the required attributes for categorization were added to the nodes, an ontological graph was generated. In this paper we shall refer to the attributes required for building this ontological graph as ‘categorical attributes’. Considering a food recipe database the categorical attributes for each recipe would be ‘Ethnicity’, ‘Course’, ‘Duration of cooking’ etc. The sub-graph for a sample selection is shown in Figure 1.

Distance Calculation using Feature Vectors

Apart from the ontological classification based on categorical attributes, the non-categorical attributes of each node were used to build feature vectors for determining similarity. In the case of a food recipe database, the non-categorical attributes can be the list of ingredients for each recipe and a feature vector can be created for every recipe based on the presence or absence of each of the ingredients in the dataset. Once the feature vectors for each node were built the multi-dimensional Euclidean distance was calculated between every pair of entities. This distance was used for calculating inter-node similarity across the ontological graph.

Assigning the weights

Every parent node in the ontological graph is connected to each of its children by a weighted link. This weight is decided by the number of times the user selected the particular child node compared to the other sister nodes in previous runs for the particular ontological sub-graph. For elements with same weight the order is decided randomly each time. For the entities that satisfy all the categorical constraints, a table of weights with respect to the different categories is generated from the ontological graph. The summation of weights with respect to all the selected categories gives the final weight of the entity.

During initialization of the system every entitiy is linked to its parent categories by a weight of ‘1’unit. Since entities having same weight are ordered randomly each time, non-biasing of any set of entities is ensured. Once the weights have been initialized the ranking mechanish governs their change according to user feedback.

Overall weight

The overall weight of an entity with respect to a set of categories is the summation of the weights of all the selection categories in that set. For example, when the user selects books of language ‘English’ and content ‘History’ the final weight of a book would be the sum of the weights in the categories ‘English’ and ‘History’. This weight is a direct measure of the relative rank of an entity in the recommendation results.

Final result-set

The final recommendation result-set consists of nodes satisfying the categorical constraints provided by the user and the related nodes determined by feature vectors. The order of the nodes in the final result-set is decided by the weight of the link: the greater the weight, the better is the rank.

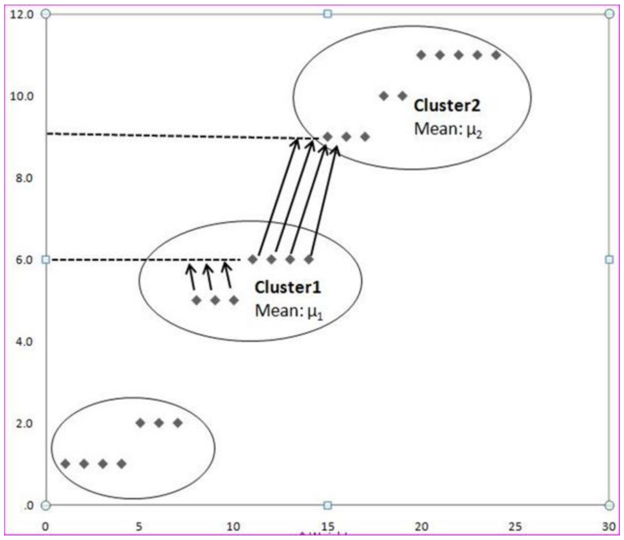


Figure 2. Cluster-Minimum method: On positive feedback, if the entity lies in subset S2, it jumps to the lowest weight of the next highest cluster. In all other cases, its weight is incremented by one unit

The ranking mechanism (Dynamic ranking layer)

Once a useful subset of entities has been obtained using the ontological graph method and weights have been initialized the next task is to incorporate adaptation to user preferences. For this purpose the system should be able to do the following:

- Find a set of potential entities for recommendation : the entities closest to the ones matching the search criteria
- Reorder the entities obtained above by assigning new weights if required

When the user selects an entity out of the several recommended ones, a positive feedback action on that entity is triggered. This feedback is used to incorporate the latest user selection into the knowledge base and use it along with the previous user data to predict future recommendations. For this task, we have formulated two methods based on clustering:

1. Cluster Minimum method; and
2. Cluster Mean method

In both these methods, Support Vector Clustering (SVC) [1] has been used. Unlike k-means clustering, support vector clustering does not require the number of clusters (or means) as initial input and we can conveniently group the outliers and take necessary steps during our recommendation process. The attributes used for clustering are the weights of the nodes with respect to each of its parents. These two methods are described in the following sections.

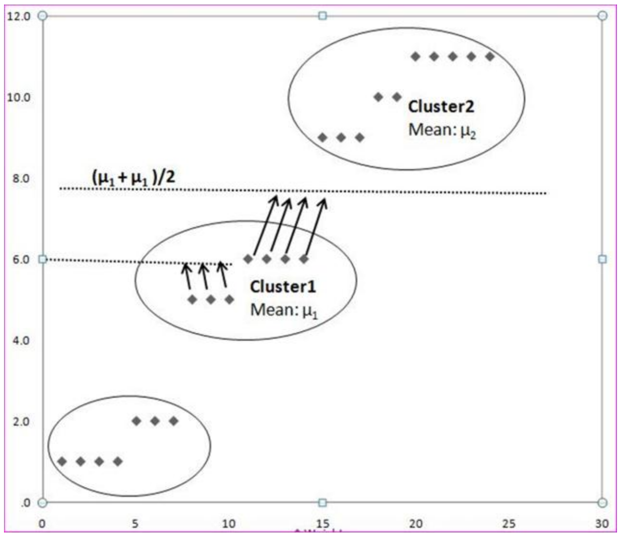


Figure 3. Cluster-Mean method: On positive feedback, if the entity lies in subset S2, its new weight is equal to the mean of the mean of its own cluster and that of the next higher cluster. In all other cases, its weight is incremented by one unit.

Cluster-Minimum Method

This method provides for rigorous adaptation and incorporates the changes in user preference speedily into the system. In this method, three subsets of entities are identified:

- S1: It contains all the entities having the maximum overall categorical weight.
- S2: It contains those entities which have the maximum weight in their own cluster (the cluster identified by support vector clustering) and do not fall in S1.
- S3: It contains all the remaining entities.

When the entity lies in S2, any increment in the value of its weight would not make any difference in the relative ranking till it becomes equal to the minimum weight of the next higher cluster. Using this idea, the following ranking scheme is developed: If the entity whose weight is to be increased lies in the subset S2, its new weight is the minimum weight of the next higher cluster whereas, if the entity lies in S1 or S3, the increment in weight is the integral value '1'. Hence, for an entity E_i with initial weight W_i and belonging to the cluster C_j, its new weight W_i' is given by

$$W_i{}' = \begin{cases} W_i + 1; & W \in S1 \text{ or } W \in S3 \\ \min(C_{j+1}); & W \in S2 \end{cases}$$

Here C_{j+1} is the next higher cluster after C_j, and $\min(C_{j+1})$ gives the minimum weight in the cluster just higher than C_j. The advantage of this method is that the system readily uses the latest knowledge about user selection in the future recommendations by increasing the weight of the entity.

A disadvantage of this method is that even if much iteration was previously required for entities of the next higher cluster to attain their weights, the maximum element in the cluster immediately jumps to the next higher cluster. Although this ensures robust adaptation, it disregards several important factors such as the number of times the user would have selected the same recipe in cases when an immediate predecessor receives positive feedback.

Cluster-Mean method

This method is similar to the Cluster-Minimum method with the difference in the way the weight of an entity lying in S2 is modified. When an entity lies in the subset S2, its new incremented weight is the mean of the means of its original cluster and that of the next higher cluster. Hence, for an entity E_i with initial weight W_i and belonging to the cluster Cj, its new weight Wi' is given by

$$W_i' = \begin{cases} W_i + 1; & W \in S1 \text{ or } W \in S3 \\ \dfrac{\operatorname{avg}(C_j) + \operatorname{avg}(C_{j+1})}{2}; & W \in S2 \end{cases}$$

Here C_{j+1} is the next higher cluster after C_j. This method preserves the past knowledge obtained by use feedback to a much greater extent than the cluster-minimum method but gives preference to the order in which the entities were selected in the course of several runs. The entities selected by user in the initial runs have the potential to attain greater weights than the entities that were given positive feedback in the later runs.

RECIPEMINER: A RECIPE RECOMMENDATION SYSTEM

The experiment deals with building a recipe recommendation system called RecipeMiner. RecipeMiner incorporates all the discussed features like ontological classification, distance measure using feature vectors and dynamic weighted ranking approach. It works by allowing user input for a set of preferences like ethnicity, course of food, calorie count, etc. and recommending food recipes to the user. The architecture of the recommendation system is depicted in Figure 4 and the work flow of RecipeMiner and its working is explained in the following section.

Construction of RecipeMiner

Data collection

The database for this experiment was collected through cooks whom we knew personally in and around the locality. The information was consolidated and attributes such as ethnicity, course, type of meal, time required for cooking, calorie content and group were added.

Generating ontological graph

Using the attributes listed in the previous section, the entire database was structured into an ontological graph. This graph forms the basis of selection of recipes for a given set of user preferences

Building Feature Vectors

For each food recipe a feature vector was built using the ingredients of the recipe. Although ingredients such as water and salt ranked very high because of their presence in most of the recipes, they were identified as irrelevant and the remaining top 200 ingredients in our database were used as the fields and the presence or absence of each of these ingredients was represented the food recipe's feature vector.

Run No. (1-20)	**Top ten match**	**Error Magnitude**	**Run No. (481-500)**	**Top ten match**	**Error Magnitude**
1	No	57	481	No	26
2	No	66	482	Yes	3
3	No	23	483	Yes	4
4	No	11	484	Yes	9
5	No	46	485	No	19
6	No	127	486	Yes	2
7	No	109	487	Yes	3
8	No	46	488	Yes	7
9	No	113	489	Yes	5
10	No	43	490	No	37
11	No	27	491	Yes	2
12	Yes	3	492	Yes	9
13	No	14	493	Yes	8
14	No	28	494	Yes	3
15	No	28	495	Yes	7
16	Yes	5	496	Yes	1
17	Yes	3	497	No	65
18	No	16	198	Yes	1
19	Yes	2	199	Yes	8
20	Yes	7	500	No	11
	Total Matches = 5	Average error Magnitude = 38.7		Total Matches =15	Average error Magnitude = 11.5

Table 1. Evaluation results for top-ten match and error magnitude recorded for a single user.

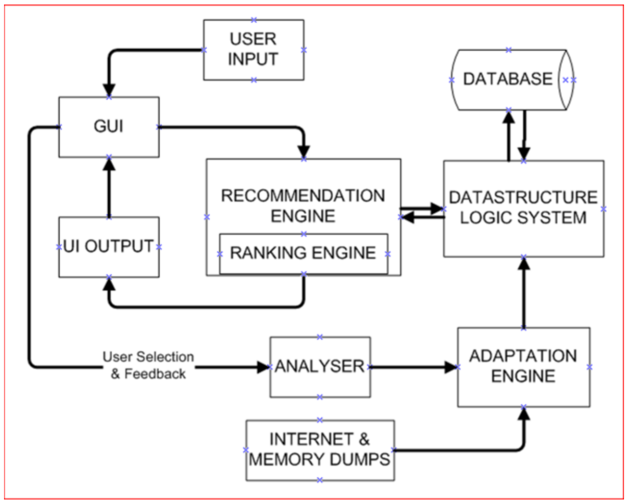


Figure 4. Basic architecture of RecipeMiner

It should be noted that the quantity of an ingredient in a recipe was not considered while building the feature vectors since this does not ensure a normalized data. A higher quantity of ingredients would unfairly bias certain recipes and it is a difficult task to compare mixed quantities of ingredients by volume and by weight per volume or weight of the cooked recipe. Figure 5 shows a sample feature vector for a recipe.

After feature vectors for every recipe were built the Euclidean distance between each pair of recipes was calculated. For our data set of 7104 recipes and 200 ingredients, tables were created for each food recipe storing distance measures with every other recipe and the food recipes with least distances from a recipe (maximum similarity to a recipe in terms of ingredients) were used as the potential recommendation items for that recipe. A sample table is shown in Figure 6. These recipes were then passed to the dynamic ranking layer which decided their ordering based on user feedback and selected preferences.

The user interface

The interface and functioning of RecipeMiner was designed

RID	Ing2	Ing2	Ing3	…	…	…	Ing200
217	1	0	0	…	…	…	1

Figure 5. Sample feature vector for a food recipe

RID	217	218	219	…	…	…	9108
217	0	5.65	3.41	…	…	…	0.05

Figure 6. Sample table of distances for a food recipe

to be simple and intuitive. After the welcome screen the user gives her preferences for different categories such as 'Ethnicity', 'Course', 'Calorie Count', etc. The user can also skip any or all the categories. When 'Get Recommended' is clicked the recommendations for the user selection are shown. The user is allowed to go back and change the selection or select a recipe out of the recommended items. The selected recipe is shown to the user in full detail and is identified as a recipe with positive feedback. The option of bookmarking favorite recipes is also provided to the user and the entire recommendation process is made as hassle-free and user-friendly as possible.

Working of RecipeMiner

Initialization

When the tool is provided to the user for the first time, the weight of all recipes with respect to all the categories is initialized to 1. This baseline initialization ensures that the system is ready to start recording patterns without being biased towards any entities or following any generalization.

User selection

The first step in is to take user input. The user can provide values for any or all of the categories mentioned in Table 2.

Analyzing user input

The user selects from the available options in the various categories as depicted in Table 2. One example can be {Ethnicity: Greek, Course: null, Time: Less, Calorie: Low, Meal: Breakfast} which means the user is looking for Greek Breakfast which has low calorie content and takes less time to cook.

Searching & Recommending

Apart from the recipes belonging to search result, the recipes having least distances are also added to the final recommendation result which is ranked according to the weights. A result-set is outputted to the user and the option of either going back to change selections or selecting a recipe out of the recommended ones is provided.

Post-processing

When the user selects a recipe and the details are displayed adaptive ranking system is triggered and the weights of the selected recipe are appropriately changed according to the cluster-minimum method.

EVALUATION AND DISCUSSION

Dimensions and metrics

To test the usability of this method in real world situations and to measure the effectiveness of recommendation, usage logs of users were studied over a period of time and the preferences as well as the final selected recipes were recorded. The usage statistics of 37 users were recorded out of which 6 were professionals in the food industry. Two of the important metrics that were measured in the experiment from were 'top-ten match' and 'magnitude of error'.

Top-ten match

For the same set of selected categories, it was recorded whether the recipe selected by the user was actually in the top ten recommended recipes. A correct match signified that the user selected a recipe from his/her top ten recommendations.

Magnitude of error

This dimension was determined by the difference in the ranks of the topmost predicted recipe and the actual recipe selected by the user.

The recordings of both these metrics are shown in Table 1. It can be seen that both, the error magnitude and the top-ten-match, gave better results gradually over a period of time as the knowledge base was enriched with quantitative feedback.

Qualitative feedback

Apart from quantitative measures as mentioned above, a feedback questionnaire was prepared to the test the 'interestingness' and 'accuracy' of the tool. Various questions were asked in both these areas and the results showed a substantially good feedback in terms of 'accuracy'.

Result

The system performed well with the set of real world recipe database of 7041 recipes. Governed by the cluster-minimum method, the ranks of the recipes changed and provided dynamic ranking to the user. The usage of ingredient based feature vectors proved to be a good method of finding similarity between recipes. The system started with near random outputs in the beginning and gradually adapted to user preferences and selection and started giving acceptable results after a series of runs.

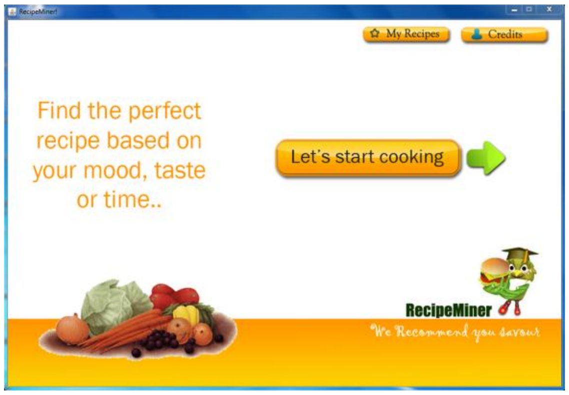
An important issue that cannot be overlooked in the process of adaptation is that of over-fitting. As the number of runs increases to a great extent, the system will tend to assign relatively large weights to a set of entities most selected in the course of usage. The issue of over-fitting will come into picture and in our scenario it will be governed mostly by two factors:

- Number of runs; and
- Diversity of selection

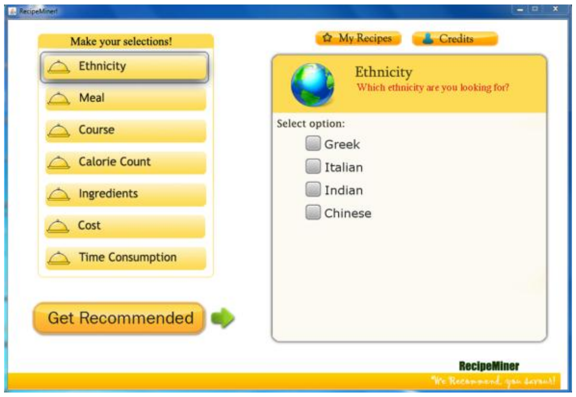
As the number of runs increases, certain entities might gain

Ethnicity	Course	Time consumption	Calorie Content	Meal
Greek	Cocktail	Long	High	Breakfast
Chinese	Appetizer	Short	Low	Lunch
Italian	Soup			Snacks
Indian	Salad			Dinner
	Main Dish			
	Side Dish			
	Dessert			

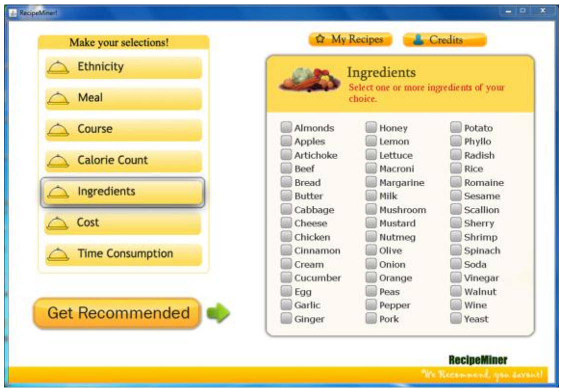
Table 2. Options for user selection in RecipeMiner



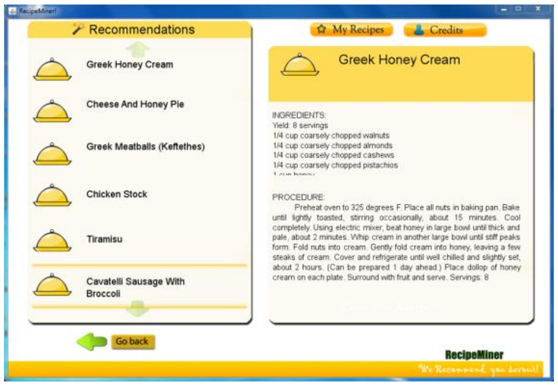
(a). Welcome screen



(b). Ethnicity selection



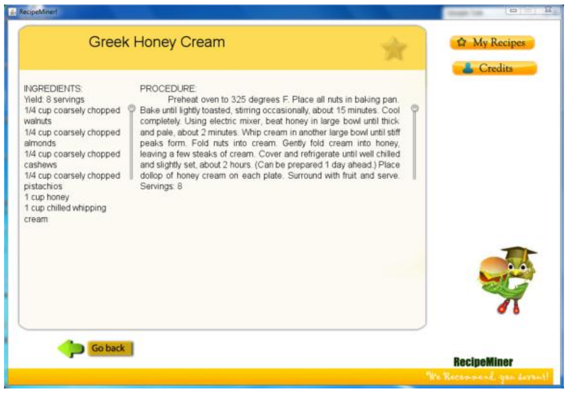
(c). Ingredients selection



INGREDIENTS:
Yield: 8 servings
1/4 cup coarsely chopped walnuts
1/4 cup coarsely chopped almonds
1/4 cup coarsely chopped cashews
1/4 cup coarsely chopped pistachios

PROCEDURE:
Preheat oven to 325 degrees F. Place all nuts in baking pan. Bake until lightly toasted, stirring occasionally, about 15 minutes. Cool completely. Using electric mixer, beat honey in large bowl until thick and pale, about 2 minutes. Whip cream in another large bowl until stiff peaks form. Fold nuts into cream. Gently fold cream into honey, leaving a few steaks of cream. Cover and refrigerate until well chilled and slightly set, about 2 hours. (Can be prepared 1 day ahead.) Place dollop of honey cream on each plate. Surround with fruit and serve. Servings: 8

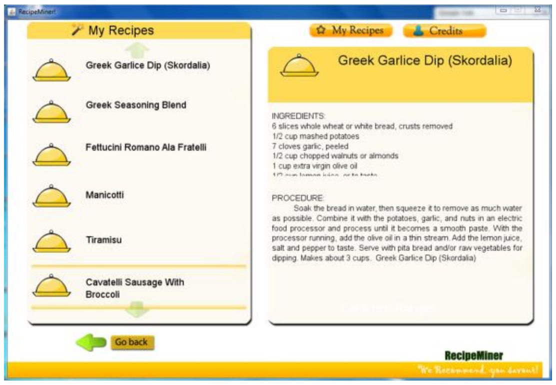
(d). Recommendation results



INGREDIENTS:
Yield: 8 servings
1/4 cup coarsely chopped walnuts
1/4 cup coarsely chopped almonds
1/4 cup coarsely chopped cashews
1/4 cup coarsely chopped pistachios
1 cup honey
1 cup chilled whipping cream

PROCEDURE:
Preheat oven to 325 degrees F. Place all nuts in baking pan. Bake until lightly toasted, stirring occasionally, about 15 minutes. Cool completely. Using electric mixer, beat honey in large bowl until thick and pale, about 2 minutes. Whip cream in another large bowl until stiff peaks form. Fold nuts into cream. Gently fold cream into honey, leaving a few steaks of cream. Cover and refrigerate until well chilled and slightly set, about 2 hours. (Can be prepared 1 day ahead.) Place dollop of honey cream on each plate. Surround with fruit and serve. Servings: 8

(e). Complete Recipe


RecipeMiner!
My Recipes
Greek Garlice Dip (Skordalia)
Greek Seasoning Blend
Fettucini Romano Ala Fratelli
Manicotti
Tiramisu
Cavatelli Sausage With Broccoli
Go back
My Recipes
Credits
Greek Garlice Dip (Skordalia)
RecipeMiner
We Recommend, you savour!

INGREDIENTS:
6 slices whole wheat or white bread, crusts removed
1/2 cup mashed potatoes
7 cloves garlic, peeled
1/2 cup chopped walnuts or almonds
1 cup extra virgin olive oil

PROCEDURE:
Soak the bread in water, then squeeze it to remove as much water as possible. Combine it with the potatoes, garlic, and nuts in an electric food processor and process until it becomes a smooth paste. With the processor running, add the olive oil in a thin stream. Add the lemon juice, salt and pepper to taste. Serve with pita bread and/or raw vegetables for dipping. Makes about 3 cups. Greek Garlice Dip (Skordalia)

(f). Bookmarked recipes

Figure 7 (a) – (f). The various application windows of RecipeMiner

a lot more relative weight than the others. However, since the adaptation mechanism controls only the relative ranking of the entities, the diversity in user selection will hinder the conditions leading to over-fitting: if the user selections have been diverse and have ranged over a variety of categories and clusters, and a large number of entities the ranks of all important entities would have elevated by some extent. Because of the distribution of total increment over a large and diversified subset of the knowledge base, the individual increment in weights would be relatively less and the factors resulting in over-fitting will be slowed down. Normalization of feature vectors and distance measures can also address this issue as the initial distances calculated on the basis of feature vectors can be normalized in due course of time depending on the user preferences. All these modifications can be made in the future when the tool is accessible to a larger audience and greater number of runs can be recorded.

FUTURE WORK

The work discussed in this paper utilizes ontological knowledge and user feedback to provide recommendation in the realm of Recipes. The same concepts and ideas can be used to build multivariate, heterogeneous recommendation systems in areas such as recommendation

Language	Title	Author	Content	Publisher
English	Culture	Hamilton Mabie	Medieval History	Penguin
Chinese	Travel	Arthur Conan Doyle	Image Processing	Addison-Wesley
Hindi	Computers	C. Collodi	Turbines	Barefoot Books
French	Programming	Mark Twain	Evolution	Catbird Press
Russian		Lucy Maud Montgomery	Biology	CRC Press
Bengali		Jack London		
Arabic		Thomas Paine		

Table 3. Attributes for Book Recommendation

of books and articles, targeting specific advertisements and advertisement placement, product recommendation, video recommendation and more. The idea can also be applied to domains such as targeting specific advertisements and advertisement placement.

Recommendation of books is synonymous to recipe recommendation as the attributes of the entities are easily identified and hence, it provides a defined model to work upon. Some major deciding factors in the case of book recommendation are shown in Table 3. With the attributes identified, the user is allowed to make choices like {Language: English; Content: Biology; Publisher: CRC Press}. This preference would generate an ontological graph with the subsets decided by language, content and publisher. The initial weights of the books with respect to various categories can be initialized to a predefined number and be subsequently increased based on user feedback and the clustering methods.

Benchmarking the proposed approach with the existing algorithms is important and requires the side-by-side evaluation of the different methods with a larger user-group. For accurate results the metrics for evaluation and the techniques need to be made as coherent as possible across the different methodologies.

CONCLUSION

Ontological relationships provide wealth of knowledge about an entity's properties and can be harvested for categorizing generic data. The use of ontological knowledge for recommendation is a promising approach and we tried to enhance it further by the use of feature vectors to determine similarity between nodes across the generated ontological graph. The final addition to the recommendation system was the dynamic ranking layer which allowed for the result-set of a particular query to change gradually as more user feedback was incorporated over time. The work in this paper exploits the quantitative feedback from user to build an ordering mechanism for the entities falling in a particular sub-graph. The results of RecipeMiner show how the system adapts well to user selections and incorporates the knowledge of past user preferences in future recommendations. This adaptation can be further enhanced by finding more accurate methodologies of re-ranking the entities based on user feedback. With the proposed algorithms the system performed better with increased user feedback and adapted well to the changes in user preferences.

REFERENCES

1. A. Ben-Hur, David Horn, Hava T. Siegelmann, Vladimir Vapnik: Support Vector Clustering. Journal of Machine Learning Research 2: 125-137,(2001)
2. A. Broder. Graph structure in the web. Computer Networks, 33(1-6):309–320, June 2000.
3. Christiane Fellbaum (1998, ed.) WordNet: An Electronic Lexical Database. Cambridge, MA: MIT Press.
4. D. Billsus and M. J. Pazzani, "User modeling for adaptive news access" User Modeling and User-Adapted Interaction, vol. 10, no. 2-3, pp. 147-180, 2000.
5. D. Mladenic, 1996. "Personal WebWatcher: design and implementation." Technical Report IJS-DP-7472, Department for Intelligent Systems, J. Stefan Institute.
6. D. W. McDonald and M. S. Ackerman, "Expertise recommender: a flexible recommendation system and architecture," in Proceedings of the 2000 ACM conference on Computer supported cooperative work, ser. CSCW '00. New York, NY, USA: ACM, 2000, pp. 231-240.
7. E. Rojsattarat and N. Soonthornphisaj, "Hybrid recommendation: Combining content-based prediction and collaborative filtering," in Intelligent Data Engineering and Automated Learning, 2003, pp. 337-344.
8. Eric Yeh, Daniel Ramage, Christopher D. Manning, Eneko Agirre, Aitor Soroa. WikiWalk: Random walks on Wikipedia for Semantic Relatedness. ACL-IJCNLP TextGraphs-4 Workshop 2009.
9. Felix Santiago, MIT Open Courseware. A Wine and Food Pairing Recommendation System
10. I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," J. Mach. Learn. Res., vol. 3, pp. 1157-1182, Mar. 2003.
11. J. M. Kleinberg, "Authoritative sources in a hyperlinked environment," Journal of the ACM, vol. 46, pp. 668-677, 1999.
12. L. Razmerita, A. Angehrn, and A. Maedche, "Ontology-based user modeling for knowledge management systems," in User Modeling 2003, ser. Lecture Notes in Computer Science, P. Brusilovsky, A. Corbett, and F.

Rosis, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, June 2003, vol. 2702, ch. 29, p. 148.

13. M Fernández. CORE: A Tool for Collaborative Ontology Reuse and Evaluation, Proceedings of the 4th International EON Workshop EON 2006 at the 15th International World Wide Web Conference WWW 2006 (2006).

14. M. Balabanović and Y. Shoham, "Fab: content-based, collaborative recommendation," Commun. ACM, vol. 40, no. 3, pp. 66-72, March 1997.

15. M. Szomszor, H. Alani, I. Cantador, K. O'Hara, and N. Shadbolt, "Semantic modelling of user interests based on cross-folksonomy analysis," in ISWC '08: Proceedings of the 7th International Conference on The Semantic Web. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 632-648.

16. M. Svensson, K. Hö"ok, and R. Cöster, "Designing and evaluating kalas: A social navigation system for food recipes," ACM Trans. Comput.-Hum. Interact., vol. 12, no. 3, pp. 374-400, Sep. 2005.

17. M. Svensson, K. Höök, J. Laaksolahti, and A. Waern, "Social navigation of food recipes," in CHI '01: Proceedings of the SIGCHI conference on Human factors in computing systems. New York, NY, USA: ACM, 2001, pp. 341-348.

18. P. Achananuparp, H. Han, O. Nasraoui, and R. Johnson, "Semantically enhanced user modeling," in SAC '07: Proceedings of the 2007 ACM symposium on Applied computing. New York, NY, USA: ACM, 2007, pp. 1335-1339.

19. P. Brusilovsky, A. Kobsa, and W. Nejdl (Eds.): The Adaptive Web, LNCS 4321, pp. 291 – 324, 2007. © Springer-Verlag Berlin Heidelberg 2007.

20. P. Dolog and W. Nejdl, "Challenges and benefits of the semantic web for user modelling." [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.13.591

21. P. Javier Herrera, Pablo Iglesias, David Romero, Ignacio Rubio, Belén Díaz-Agudo. JaDaCook: Java Application Developed and Cooked Over Ontological Knowledge.

22. P. Massa and P. Avesani, "Trust-aware collaborative filtering for recommender systems," 2004.

23. R. Bell, Y. Koren, and C. Volinsky, "Modeling relationships at multiple scales to improve accuracy of large recommender systems," in KDD '07: Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining. New York, NY, USA: ACM, 2007, pp. 95-104.

24. S. Brin and L. Page, "The anatomy of a large-scale hypertextual web search engine," Computer Networks and ISDN Systems, vol. 30, no. 1-7, pp. 107-117, 1998.

25. Stuart E. Middleton, Nigel R. Shadbolt, and David C. De Roure: Ontological user profiling in recommender systems. ACM Trans. Inf. Syst., 22(1):54–88, January 2004.

26. T. H. Roh, K. J. Oh, and I. Han, "The collaborative filtering recommendation based on som cluster-indexing cbr," Expert Systems with Applications, vol. 25, no. 3, pp. 413-423, October 2003.

27. V. Schickel-Zuber and B. Faltings, "Overcoming incomplete user models in recommendation systems via an ontology advances in web mining and web usage analysis," ser. Lecture Notes in Computer Science, O. Nasraoui, O. Zaïane, M. Spiliopoulou, B. Mobasher, B. Masand, and P. Yu, Eds. Berlin, Heidelberg: Springer Berlin / Heidelberg, 2006, vol. 4198, ch. 3, pp. 39-57.

28. Z. Syed, T. Finin and A. Joshi, Wikitology: “Wikipedia as an ontology”, Proceedings of the Grace Hopper Celebration of Women in Computing Conference, October 2008.