UNIVERSITY
OF TRENTO

DEPARTMENT OF INFORMATION AND COMMUNICATION TECHNOLOGY

38050 Povo – Trento (Italy), Via Sommarive 14
http://www.dit.unitn.it

Reading between the lines: Using hierarchically structured preferences to capture query semantics

Hamza Hydri Syed, Periklis Andritsos

December 2006

Technical Report # DIT-06-094

Reading between the lines :
Using hierarchically structured preferences to capture query semantics

Hamza H. Syed, Periklis Andritsos
Department of Information and Communication Technology
University of Trento, Italy
{syed, periklis} @dit.unitn.it

Abstract

Traditionally, search engines treat the keywords entered by users as a text string, without trying to understand the sense behind the user's need. Others, employ natural language processing techniques or external dictionaries to find what the users' needs may be. At the same time there has been some recent work done in trying to understand the user better and model her interests to improve the relevancy of the results retrieved by the search engine. In this paper, we present a new hierarchical modeling of a user's interests which allows for the placing of a particular preference into context and an intuitive way to order her preferences, such that higher relevancy is achieved in the retrieved results. We discuss an approach that exploits these inherent semantics from the hierarchical structure and re-ranks the retrieved results for the particular information need. We accompany our approach with a set of experimental results, which show the merits of such an approach and open new avenues of research in this area.

1 Introduction

Alice, Bob and Charlie are average internet users and use the popular search engines for looking up for any information. Alice is a chef by profession and loves Italian cuisines. Bob is a graduate student and a big fan of Italian Football, while Charlie is writing a thesis on medieval art and is looking for information on Italian painters and sculptors. When they enter the keyword `Italy` in a search engine, traditionally they are shown results depending on the popularity of webpages (aka *pagerank* [20]), independent of their individual information needs and interests on the topic of `Italy` - Alice is interested in webpages on Italian dishes and recipes, Bob is looking for updates on Italian Football league, while Charlie is interested in articles on Michelangelo and Bernini. There has been some recent work done in trying to understand the user's need, by using her past search/click history (*Google Personalised Search*[1]) or by clustering of results in categories and allowing the user to select one section and view further results in that category (*Vivisimo*[2]). But these approaches have limitations in understanding the user's interest and the semantics behind the keyword she is searching for. Query expansion by appending terms to the keyword [15] helps in adding some semantics to the information need, but in todays world, where there is an increasing usage of handheld devices, appending terms and expanding the query adds an extra burden on the user. Personalising the search activity using a profile of her interests helps us in understanding what the user is searching for and retrieve results accordingly, without the addition of terms at query time.

[1] www.google.com/psearch
[2] www.vivisimo.com/

In this paper we present an approach towards modeling the user preferences in a hierarchical structure and utilize the relationships between the terms appearing in the *Preference Tree* to interpret the query semantics by looking at the context of the user's interests. Concretely, we propose a rooted *Preference Tree* $\mathcal{P}$ with the following features

- labels storing the *preference-tuples* on its nodes, indicating the atomic preferences and the user's degree of interest as a value
- *preference-weight* is the value used to resolve the relative preferences between the common interests of the user
- we utilize this hierarchical relationships to build a *preference-vector*, that captures the semantics of the keyword inherently, and use it to rank results based on similarity matching.

The remainder of the paper is organized as follows. In Section 2, we describe our model of user preferences and elicit its finer details. In Section 3, we describe our algorithm, its implementation and experimental results are reported in Section 4. In the end, we specify some related work and conclude in section 6 with some recommendations of future work.

2 Modeling of User Preferences

The key ingredients of a personalized information retrieval system are the techniques for capturing the preferences from the user and those for modeling these preferences in an efficient manner to be used when the user issues keyword search queries. In this work we present our approach towards modeling these preferences, proceeding under the assumption that the system has acquired the knowledge of the user preferences already. In our approach, we consider a hierarchical structure to model the user preferences,

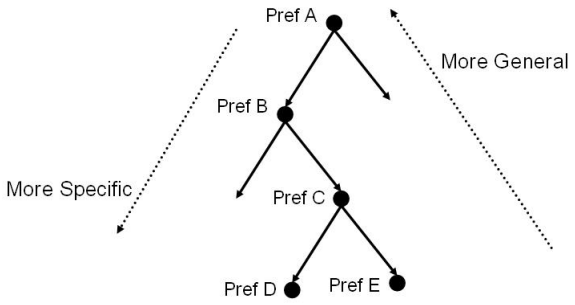

Figure 1: Model Preference Tree

as shown in Figure 1. The preference labels are placed along the nodes of a tree in a manner such that, the higher a node appears in the *Preference Tree*, the more general is the concept it represents. As she traverses deeper down the tree, the user becomes more and more specific with her preferences. This assumption is in accordance with recent work that assumes general hierarchical structures for schema matching [13] and ontology elicitation [12] purposes. More specifically, Giunchiglia et al have introduced the notion of the *concept of a node*, which, for each node in a hierarchical structure of concepts, represents the information content of this node [13]. In our model, nodes are preference labels and hence, following the

formalism in [13], when a node $preference_B$ appears below a node $preference_A$, there is a parent-child relationship between them, making the child node conceptually more specific than the parent one. In this manner, starting from a particular node in the hierarchy, its semantics are inherently implemented in our *Preference Tree* through the use of a simple conjunctive propositional formula whose elements are the nodes in the path towards the root. This way, we are able to disambiguate concepts that appear more than once in the tree and concepts which accept the same semantics. The advantage of using this idea for disambiguating the concepts is that users may define their own hierarchical structures, such as their directory tree in their e-mail application or the file structure in their system, and use it according to their needs.

Let us elaborate on the further details of our preference model. We define our *Preference Tree* as a rooted tree $\mathcal{P} = \langle \mathcal{V}, \mathcal{E}, \mathcal{L} \rangle$; where $\mathcal{V}$ is a finite set of nodes, $\mathcal{E}$ is a finite set of edges on $\mathcal{V}$ and $\mathcal{L}$ is a finite set of labels defined as *preference-tuples*. The *preference-tuple* - $\langle name, \omega \rangle$, is a combination of an atomic preference term and a number indicating the user's degree of interest. Here, *'name'* is the preference name, and 'ω' is the *preference-weight*, which we introduce as a measure of the relative preference among the siblings of a particular node.

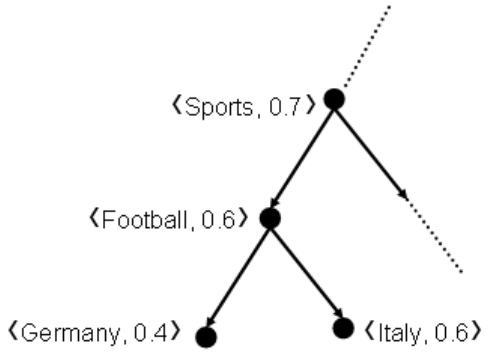

Figure 2: Sample Preference Tree

Consider a particular section of a user's Preference Tree, shown in Figure 2, describing her interests in sports. The *preference-tuples* are placed on the nodes as depicted. The hierarchical paths of preferences for `Germany` and `Italy`, both being sibling nodes with common ancestors `Sports` and `Football`, are the following

$$\texttt{Sports} \longrightarrow \texttt{Football} \longrightarrow \texttt{Germany}$$

$$\texttt{Sports} \longrightarrow \texttt{Football} \longrightarrow \texttt{Italy}$$

where $\omega_{Germany}$ and ω_{Italy} (0.4 and 0.6 respectively in the figure) weights give an indication of the relative preference between them. The assigning of numeric values to preferences/interests is built on the notions suggested in [14], [9], [8]. Using a *cardinal utility*-based approach we quantify the preference of the user for a given concept into a real number. Given a context of preference, these numbers represent the *degree of interest* the user has among the sibling preferences.

In the Preference Tree model we are using, we assume the following properties for the node weights ω **- Properties:**

- for all the sibling nodes of node *"k"*, the ω values are normalized such that they sum to unity. Hence, $\sum \omega_i = 1$, where ω_i is the weight for a node *"i"* that is a child of node *"k"*. This normalization of the sibling preferences is done to convey the relativeness amongst these preferences.

- if $\omega_A < \omega_B$, this indicates that the user prefers $preference_B$ to $preference_A$. Referring to the above example, $\omega_{Germany} < \omega_{Italy}$, i.e. the user gives higher preference to the Italian Football than the German Football, though both of them are of some interest to the user and appear in her *Preference Tree*

- the *preference-weight* for a node ω_A is independent of the *preference-weights* of its parents or children and is only dependent on the *preference-weight* of its sibling nodes. Considering the same example preference, if a user prefers the Italian Football team, we consider its relative preference as compared to its sibling preferences and the preference weight of its parent (`Football`), would not influence the relativeness of sibling preferences. For this work, we assume the preferences to be atomic terms (`Italy`, `Music`, `Sports`, etc.) and not a combination of terms (`sunset at the beach`, `South Italy`, etc)

Finally, we assume that the user-defined *Preference Tree* is embedded in the personalization module of an information retrieval system, such as a search engine, and can be utilized while performing the retrieval and ranking of the results using similarity matching methods. Further issues regarding the algorithm and its implementation are described in the following section.

2.1 Preference Weight Vector

In the previous section, we claimed that the semantic information of a preference is stored inherently in the hierarchical structure we have proposed, as our heuristic is substantiated by the ideas proposed by [11] and [13]. Similar to the Conjunctive Normal Forms suggested by the authors, we store the relationship between a node and its ancestors in the form of a vector, which we term as the *preference-vector* and is stored along with the *preference-tuple* on all the nodes of the *Preference Tree*. This section introduces the details of the creation and usage of these vectors.

Considering an example *Preference Tree* as illustrated in figure 4, we focus our discussion on the section of the tree describing the sporting interests of the user. As is evident, the 'preference term' `Italy` appears at 3 places (`Italy` and `Italian` would result as '`Ital`' after the word stemming process on the terms). For the purpose of discussion, consider the two occurances of `Italy` under the different paths of

`Leisure` $\longrightarrow$ `Sports` $\longrightarrow$ `Football` $\longrightarrow$ `Italy`
and `Leisure` $\longrightarrow$ `Travel` $\longrightarrow$ `Europe` $\longrightarrow$ `Italy`

Since their positions in the tree are different, their semantic meanings are also different from one another (recollect the notion of *the concept of a node* from the earlier discussion). We follow the formalism proposed by Giunchiglia et al [13] and calculate the logical formulas corresponding to these nodes as

[`Leisure`$\wedge$`Sports`$\wedge$`Football`$\wedge$`Italy`]
and [`Leisure`$\wedge$`Travel`$\wedge$`Europe`$\wedge$`Italy`]

We utilize these formulae to construct the *preference-vectors* which are stored on the nodes of the *Preference Tree* as mentioned above. For example, for a node `Italy` under the path,

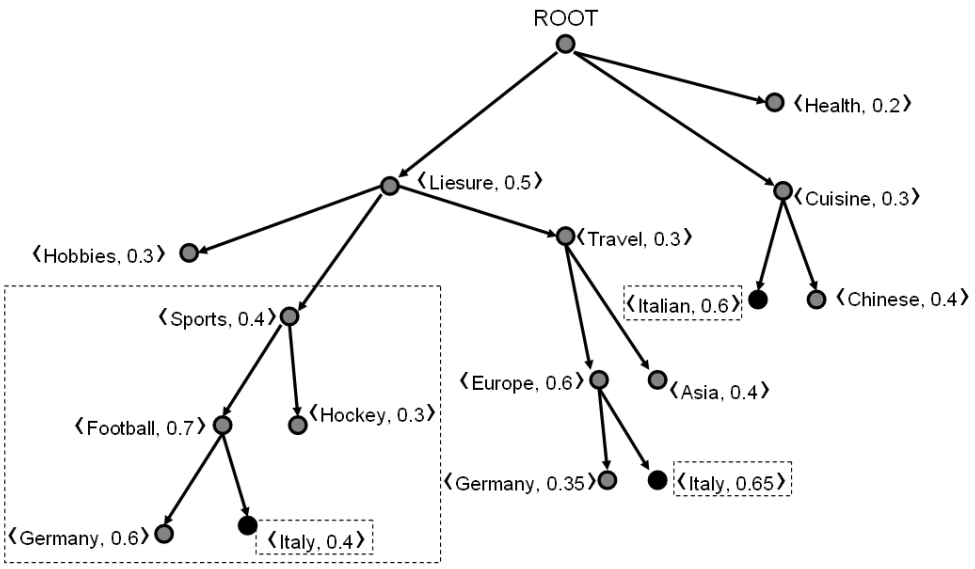

Figure 3: Sample Preference Tree

`Leisure` $\longrightarrow$ `Sports` $\longrightarrow$ `Football` $\longrightarrow$ `Italy`,

the *preference-vector* is calculated and stored on its node as

[(`Leisure`,$\omega_{Leisure}$);(`Sports`,ω_{Sports});(`Football`,$\omega_{Football}$);(`Italy`,ω_{Italy})]

When the user issues a search command for a keyword, we select those nodes/leaves of the Preference Tree which are having terms closest in similarity to the given keyword (in the above example, a search for keyword `Italy` would activate the path of preferences as described above). For the selected nodes in the *Preference Tree* we use their *preference-vectors* to perform the similarity matching and re-ranking of the documents retrieved for the queried keyword. As our experimental results suggest, this approach helps to distinguish between documents containing the same terms but having totally different semantic meanings.

For instance, consider a document describing some touristic information about Italy, with some travel tips etc. The term `Italy` appears prominently in its text. Usually with the regular document retrieval methods involving similarity measure of document and term frequency vectors, this document would find a similarity match for the node `Italy` under `Sports` as well as for the node `Italy` under `Travel`. But with our approach, it becomes easier to disambiguate the semantic sense of the query ('`Italy`') and retrieve this document if the user has specified higher preference for node `Italy` under `Travel` as compared to the node `Italy` under `Sports`. In this way, if the similarity matching is done using the *preference-vector*, it becomes easier for the searching algorithm to distinguish between two documents (both having `Italy` as a prominently appearing term, but having different semantic senses - one describing the travel tips for Italy and the other describing the Italian football team) and rank them appropriately reflecting the relative preferences of the user.

At present we limit ourselves for searching only the keywords (terms) appearing in the *Preference Tree*

of a user. The issue of semantic matching of any given keyword, not present in the user's preferences, to the terms mentioned explicitly by the user and the issue of resolving the results of searches on keywords that do not appear at all in the *Preference Tree*, are currently being looked into using the ideas proposed in [13].

2.2 Scaling of ω values

One would argue that while doing similarity matching between the *preference-vector* of a node with the document vectors from the collection corpus, there could be some false positives in the results, since the ancestor node terms appear along with their ω-values in the *preference-vector* of a particular node. For this reason we introduce the scaling of the ancestor ω-values for a given node.

The scaling of the ω values of the ancestors is necessary to reflect the hierarchical relationship between the nodes and to diminish the influence of the *preference-weights* of the ancestor nodes, relative to the distance of the ancestor node from the particular node. It emphasizes the specificity of the term being searched.

The *preference-vector* for a node A, contains its preference weight ω_A and the scaled value of its ancestors preference weights.

For example, given a preference node `A` with *preference-weight* ω_A,

- if node `B` is an ancestor then its scaled *preference-weight* is

$$\omega_{B'} = \omega_B * 1/m^k$$

- *where m* $\geq$ *2* and *k* is its path distance, along the tree, towards the root node starting from node `A`

For each step `k` one goes up the tree along the path of the ancestors, the scaled weight is $1/m^k * \omega_A$ where ω_A is the original *preference-weight* of node `A`.

Consider a sample path as shown below, with the arrow direction indicating the `parent` $\longrightarrow$ `child` relation

$\texttt{A}[\omega_A] \rightarrow \texttt{B}[\omega_B] \rightarrow \texttt{C}[\omega_C] \rightarrow \texttt{D}[\omega_D]$

and the *preference-vector* for the node `D` which is (calculated by procedure explained above)

$[(A, \omega_{A'}); (B, \omega_{B'}), (C, \omega_{C'}), (D, \omega_D)]$

this is how the calculations are done:

- for node `A`, *k* = 3 $\Rightarrow \omega_{A'} = \omega_A * 1/m^3$
- for node `B`, *k* = 2 $\Rightarrow \omega_{B'} = \omega_B * 1/m^2$
- for node `C`, *k* = 1 $\Rightarrow \omega_{C'} = \omega_C * 1/m^1$

In this manner the influence of the preference weight of the ancestors is slowly diminishing as we are moving up from the node's position in the direction of the root node, in the *Preference Tree*. Considering an example similar to the one mentioned before.

$\texttt{Leisure}[\omega_{Leisure}] \rightarrow \texttt{Sports}[\omega_{Sports}] \rightarrow \texttt{Football}[\omega_{Football}] \rightarrow \texttt{Italy}[\omega_{Italy}]$

the *preference-vector* for the node `Italy` would be

$[(Leisure, \omega'_{Leisure}); (Sports, \omega'_{Sports}); (Football, \omega'_{Football}); (Italy, \omega_{Italy})]$

Considering the *Preference Tree* in figure 3,

$[(Leisure, 0.5); (Sports, 0.4); (Football, 0.7); (Italy, 0.4)]$

with m = 2, the scaled *preference-weight* (ω) values would be

$[(Leisure, 0.031); (Sports, 0.1); (Football, 0.35); (Italy, 0.4)]$

As we see, though `Football` had the highest ω-value in this *preference-vector* for `Italy`, after scaling its effect on the similarity matching process (while querying and ranking) is reduced, such that results having the maximum similarity with the term `Italy` but influenced strongly by the term `Football` are given higher priority. Without this scaling, the term `Football` would have a stronger influence on the results as compared to `Italy`.

In this case the documents talking about Italian Football and having semantic sense of Italy as a Football team would have the highest similarity match to this node, while documents on Football in general, Sports in general sense or any other Leisure related documents would have lower similarity match. In this way, though we retain the semantic information from the ancestor nodes, we reduce the possibility of false positives occurring caused by the ancestor terms, while doing similarity matching with term weight vectors for the documents.

3 Algorithm

The pseudo code of our algorithm is as described below. The $CollectionCorpus$ is our Test Collection (more details about it are mentioned in the following section). $PreferenceTree$ is the hierarchical tree of user preferences. The variable parameter *'k'* can be set at run time. For our tests we select (an arbitrary choice of) the top 30% of the retrieved results. The end result is a list of documents ranked in an order reflecting users interests in that context.

For each of the document in the collection corpus, we create the $term - weight - frequency$ vector using the classical *TF-IDF* measure [22], where wgt of a term in the particular document is given by

$wgt = tf * idf$

where

tf is the Term Frequency

idf is the Inverse Document Frequency

We apply the `Porter Stemmer` algorithm [21], on each of these terms to strip any suffixes, if present. The similarity measurement is done using the classical cosine similarity given by the formula

$sim(pwv,twv) = \frac{|pwv \bullet twv|}{|pwv| * |twv|}$

where *pwv* is the $preference - vector$ and *twv* is the $term - weight - vector$

As for the ranking function, we have now implemented an ordering based on the similarity score of the documents, where the document having the maximum similarity with the semantics of the query keyword (captured in the *preference-vector*) is ranked higher. Currently we are working with modifications of this ranking, based on the ideas proposed in [3].

Algorithm 1 Weight-based Preference matching

Input: $PreferenceTree$, k
Output: List of Documents
  for all $document \in CollectionCorpus$ **do**
    form the $term - weight - vector$
    for all $term \in term - weight - vector$ **do**
      apply `Porter Stemmer` to remove any suffix
    end for
  end for
  for all $node \in PreferenceTree$ **do**
    get the ancestors
    form the $preference - vector$
    scale the ω values of the ancestors
    perform similarity measure with the $term - weight - vectors$
    sort documents based on similarity score
    return top-k documents
  end for
  return overall ranked list of documents for the entire $PreferenceTree$

4 Implementation and Experience

For our experiments we made use of the Google API [3], for issuing keyword queries on the web. Using a wrapper around it we query for terms given by users and store the retrieved results as a test collection. The user preferences were obtained explicitly from the users (a group of 15 graduate students from our University campus) and stored in the form of a hierarchical Tree structure. The *Preference Trees* were different for each of the user (depending on the respective interests and preferences of the users), with an average of 75 nodes and a depth of 5 levels. We implemented the hierarchical trees in SQL, using the Adjacency List Model prescribed by [5]. The user interface and the control logic was implemented in Java. At present the user interface is in elementary stage, improvising the GUI to interact effectively with the user - to capture the users preferences/interests implicitly and to present the retrieved results in a better way is part of our future work.

In these experiments we consider $m = 2$, for scaling of the *preference-weights* of the ancestors.

[3] http://code.google.com/apis/ajaxsearch/

4.1 Experimental Results

We asked for 3 keyword queries from the users (whose preferences were collected a priori) and queried the Google API. For each query term, we retrieve the top-500 ranked results and stored them as a collection of documents. We store the snippets of the results retrieved by the Google search engine as the contents of the documents. We believe the summary of a webpage reflects its contents and is a sufficient metric for its classification [19]. We then repeat the search procedure using the user's *Preference Tree* for searching in this collection and rank these documents based on her preferences.

We collected the top 300 results for the keyword `Italy` (using Google search engine on 12th December,2006) and the documents, which were created using the above described method. Indicatively, Table 1 shows the top 10 results retrieved by Google and their page summaries. The top 300 documents were used as a Test Collection for re-ranking under the user's preferences. We compare the revised ranking given to the documents by our Weight-based Preference matching algorithm (described in section 3) and the original ranking given by Google. We used a sample *Preference Tree* (similar to the one depicted in figure 3), in which the user had a preference for Italian cuisine (recipes, dishes, etc.) and for Italian art (paintings, music, etc.) - but upto varying degrees. In this case, the user was more interested in food as compared to music (depicted by the *preference-weights* for these concepts in her *Preference Tree*).

As this user preferred Italian cuisine and was interested in Italian art, the preferences-based ranking clearly rated these documents higher. If we compare the documents from Table 2, we see that the result ranking based on our weight-based Preference matching algorithm clearly understands the user's needs better and gives higher priority to the results related to them. Webpages describing Italian food and Italian art are rated higher, while the rankings without preferences rates webpages based on their popularity (football, tourism etc). We also see that the relative preferences between the interests are also understood well by the algorithm and reflected in the ranking order. The effect of relative scaling of ancestors was visibly evident. Without the scaling, the algorithm was returning some false-positives (pages most similar to the ancestor node's semantics, were rated higher), but by relatively scaling the ω-values of the ancestors, we can achieve greater resolution on the semantics of the queried keyword.

We then presented to the user 2 sets of results - the top 30 documents retrieved by Google and another top 30 set of documents re-ranked using their respective *Preference Trees*. The user's opinion was asked if these rankings were reflecting their preferences/interests for that keyword. 80% of the users agreed that the revised rankings (using our weight-based Preference matching algorithm) reflected their choice of documents for a given keyword query.

As has been observed, by modeling the user's preferences in an hierarchical structure (*Preference Tree*), and building a wrapper around any search engine (Google in our case) to manage these preferences, we can achieve greater understanding of the semantics behind the user's information need. In an earlier set of experiments [23], we had observed that using hierarchical models of user preferences, we could improve the effectiveness of the results retrieved for a user query from a large collection of documents.

5 Related Work

As we have seen modeling of the preferences is an essential technique to implement any preference based search. The other major feature is the manner in which these preferences are captured from the user. The authors in [24] have described some interesting techniques to elicit these preferences from the user and

Table 1: Rankings by Google Search Engine

Webpage Summary	Google Rank
Italian tourist board site (North America). Background, tourism, events in Italy	1
Italy footballers went into the 2006 FIFA World Cup against the difficult backdrop of a match-fixing scandal yet ended up rekindling their nation love ...	2
News and views on Italy, property in Italy, holidays in Italy, hotels in italy, weddings in italy, culture in italy, fashion and style in italy - From the ...	3
Displays current conditions for all major cities on one page, with five-day forecast and more detailed information for each available	4
Travel information about Italy, Hotel booking and accommodations, complete tourist guide of main cities, all you need for your next travel to Italy	5
Italy: The Wonderful Country: Tourism and Culture	6
You will soon be zipping around like a local, thanks to this map of Italy provided by Lonely Planet	7
Italy is among the top teams in international football and the second most successful ... Italy's predominance on international football in the 1930s was ...	8
It has become obvious that I don't have time to maintain this page. Yahoo is an excellent starting place for information of all kinds	9
Information and on-line reservation on hotel, Spas, thermal sites, ground transportation services, religious tourism, school tourism, amenities and tour ...	10

Table 2: Comparison of Rankings

Webpage Summary	Google (simple)	Google + Preferences
An introduction to Italian shopping with information about fashion and design, food, wine, and travel companies. Also featuring a shopping guide to Italy ...	40	1
Italy. italian composers; italian culture; italian history; italian language; italian literature	137	2
Italytour is a directory of italian tourist resources organized for categories, contain information about italy, italian hotels and restaurants, travel, ...	226	3
Bite size portions of information for the independent visitor to Italy. Food, accommodation, cultural events, links and daily press releases.	57	4
Italy's rich history of Food and Wines can be studied through a variety of museums, which allow the public first hand to observe the gastronomic delights. ...	231	5
Maps, photographs and historical information on various Italian towns.	79	6
This page provides information on the main websites of the Italian government.	11	7
Photographic tour of Italy and home to Palimpsest: An Anthology of Italian History and Culture, which covers historical Italian people, places, and events.	86	8
I normanni nel Sud: The history of the Normans in Italy; Medioevo Italiano: Medieval Italian ... General Internet Resources: Italy and Italian Language ...	124	9
Italy. The Global Gourmet features daily updates, international recipes, cookbook profiles, regular columnists, food news, cooking tips, wine product ...	98	10

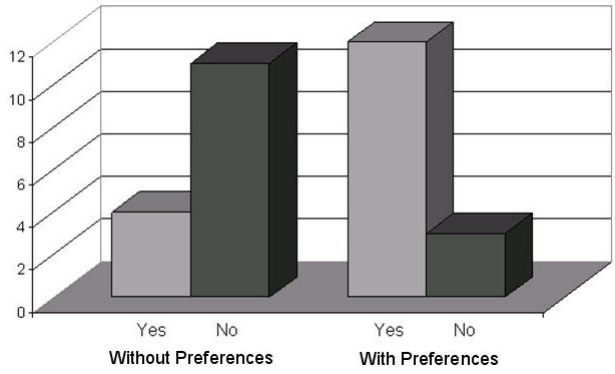

Figure 4: Ranking of Documents

discuss the influence on the accuracy of the results retrieved. Another approach towards extracting user knowledge is discussed in [2]. The authors describe an ontology-based dialog agent, called OWL-OLM, which is a framework to elicit and maintain a model of the user's conceptualization using an ontology driven dialog. Our approach is orthogonal to the work done on utility elicitation techniques, such as the one by [6], where the authors tackle the problem of eliciting the user utility function as a classification problem.

The re-ranking of the retrieved results based on the user interests helps in achieving greater efficiency in satisfying the user's information need, as the rankings reflect her preferences on that context. The literature presents several approaches towards ranking of results to achieve this efficiency. Lukasiewicz and Schellhaze [18] present a ranking scheme based on the conditional preference bases, which consist of a description logic knowledge base and a finite set of conditional preferences and use this scheme for ranking objects in ontologies. Pretschner and Gauch [10] consider the preference profile as an ontology and use this to retrieve webpages based on the user's interest. They provide an alternative approach towards the ranking by measuring the similarity of the document to the concept appearing in the user's preferences.

Query personalization and modeling of user's needs have attracted interest in both IR and Database research communities. The preferences in the real world negotiate for the best possible match and are hence treated as soft constraints as in [16], their "Best Matches Only" (BMO) query model treats preferences as strict partial orders. A similar framework has been suggested in [7], where preferences are specified using first-order logical formulas and embedding them into relational algebra by introducing a *winnow* operator that selects the most preferred tuples. A preference model for query personalization in database systems is defined in [17]. The authors present a framework which makes use of structured user profiles(storing simple, unconditional preferences). Similar approach of implementing quantitative preferences in Databases was proposed earlier in [1], which provides a framework for expressing and combining preferences. As in our technique, both these approaches used a numeric score between [0,1] to indicate the user's degree of interest in the atomic concept represented by the preference term.Our approach differs by normalising these *numeric interests* and using them to disambiguate the relative preferences between the sibling nodes of our *Preference Tree*. In comparison to these approaches, our model provides a heuristic to understand the semantics of the keyword from the concept represented by its position in the *Preference Tree*

6 Conclusion and Future Work

As discussed, the existence of a vast amount of information on the Internet leads to an overwhelming experience for users who attempt to browse through them, while searching for an information need. Work on user preferences has recently acquired more attention in order to retrieve and rank the documents relevant to a user. We have presented an approach which models the user's interests/preferences in an hierarchical structure called as *Preference Tree* and utilizes the context of the user's interests to understand the semantics behind a query keyword. We have also seen that by introducing a small metric of *preference weight* on each of the nodes of the *Preference Tree*, we were able to handle the relative preference between common interests (which appear as sibling nodes in the tree). On comparing the ranking of results using our weight-based Preference Tree algorithm with the rankings obtained without any preferences, we observed that, with our technique its easier to achieve greater effectiveness in satisfying a user's information need. Our preliminary experiences demonstrate the merits of such an approach and open new avenues of research in this area. Though our work may appear similar to the one proposed by [4], as our *Preference Tree* may be inferred as the user ontology used by them, we claim that our approach is different in the sense that we introduce the *weight parameter* to quantize the relative preferences and we implement semantics latently in the *Preference Weight Vector* stored at the nodes of the tree representing the user preferences.

As stated before, there are several issues we need to address to. We have to test our algorithm with a large set of people from different backgrounds, so that we can observe how the algorithm reacts to different groups of people with varied sets of preference profiles. In these experiments, our Preference Tree had about 80 nodes and 5 levels of depth. We want to test it, with a much deeper Preference Tree, with a large number of nodes. The issue of semantic matching of a query keyword not appearing in the user's *Preference Tree* to the existing nodes needs to be resolved.

We want to expand our test collection and implement our algorithm over a much larger collection of heterogeneous documents. At present we are experimenting with a Weblog dataset of 400,000 documents [4]. We are also improving our GUI interface such that we could embed our algorithm as a plug-in into a web browser and help the user in her searching tasks.

7 Acknowledgments

The authors would like to thank Fausto Giunchiglia for the useful inputs in reporting this work. We also thank Vaishak Belle for helping in the implementation work of the experiments.

References

[1] R. Agrawal and E. L. Wimmers. A framework for expressing and combining preferences. In *SIGMOD Conference*, pages 297–306, 2000.

[2] L. Aroyo, R. Denaux, V. Dimitrova, and M. Pye. Interactive ontology-based user knowledge acquisition: A case study. In *ESWC*, pages 560–574, 2006.

[3] B. T. Bartell. *Optimizing ranking functions: a connectionist approach to adaptive information retrieval*. PhD thesis, La Jolla, CA, USA, 1994.

[4] Buzzmetrics Dataset - http://www.nielsenbuzzmetrics.com/

[4] D. Caragea, J. Z. 0002, J. Bao, J. Pathak, and V. Honavar. Algorithms and software for collaborative discovery from autonomous, semantically heterogeneous, distributed information sources. In *ALT*, pages 13–44, 2005.

[5] J. Celko. *SQL for Smarties: Advanced SQL Programming*. Morgan Kaufmann, 1995.

[6] U. Chajewska, L. Getoor, J. Norman, and Y. Shahar. Utility elicitation as a classification problem. In *UAI*, pages 79–88, 1998.

[7] J. Chomicki. Querying with intrinsic preferences. In *EDBT*, pages 34–51, 2002.

[8] P. Fishburn. *Utility Theory for Decision Making*. Huntington, NY. Robert E. Krieger Publishing Co., 1970.

[9] P. Fishburn. Preference structures and their numerical representations. *Theor. Comput. Sci.*, 217(2):359–383, 1999.

[10] S. Gauch, J. Chaffee, and A. Pretschner. Ontology-based personalized search and browsing. *Web Intelli. and Agent Sys.*, 1(3-4):219–234, 2003.

[11] F. Giunchiglia, M. Marchese, and I. Zaihrayeu. Towards a theory of formal classification. In *Technical Report DIT-05-048*, Informatica e Telecomunicazioni, University of Trento, 2005.

[12] F. Giunchiglia, M. Marchese, and I. Zaihrayeu. Encoding classifications into lightweight ontologies. In *ESWC*, pages 80–94, 2006.

[13] F. Giunchiglia, P. Shvaiko, and M. Yatskevich. S-match: an algorithm and an implementation of semantic matching. In *ESWS*, pages 61–75, 2004.

[14] R. Keeney and H. Raiffa. *Decisions with multiple objectives: Preferences and value tradeoffs*. J. Wiley, New York, 1976.

[15] J. Kekäläinen and K. Järvelin. The impact of query structure and query expansion on retrieval performance. In *SIGIR '98: Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 130–137, New York, NY, USA, 1998. ACM Press.

[16] W. Kießling. Foundations of preferences in database systems. In *VLDB*, pages 311–322, 2002.

[17] G. Koutrika and Y. E. Ioannidis. Personalization of queries in database systems. In *ICDE*, pages 597–608, 2004.

[18] T. Lukasiewicz and J. Schellhase. Variable-strength conditional preferences for ranking objects in ontologies. In *ESWC*, pages 288–302, 2006.

[19] I. Mani. *Advances in Automatic Text Summarization*. MIT Press, Cambridge, MA, USA, 1999.

[20] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford Digital Library Technologies Project, 1998.

[21] M. F. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, July 1980.

[22] G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Inf. Process. Manage.*, 24(5):513–523, 1988.

[23] H. Syed and P. Andritsos. Weigh your preferences!: Towards using hierarchies for building personal libraries. In *Technical Report DIT-06-062*, Informatica e Telecomunicazioni, University of Trento, 2006.

[24] P. Viappiani and B. Faltings. Implementing example-based tools for preference-based search. In *ICWE '06: Proceedings of the 6th international conference on Web engineering*, pages 89–90, New York, NY, USA, 2006. ACM Press.