A Lightweight Tree Structure to Model User Preferences

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. 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.

1 Introduction

Alice is a chef by profession and an average internet user who relies on the popular search engines to find any information on the internet. When she enters the keyword `Italy` in a search engine, traditionally she is shown results depending on the popularity of web pages (a web search for '`Italy`' retrieves results with tourism and football related web pages ranked higher), which are not related to her interests on the topic of `Italy` (Italian cuisines in this case). There has been some recent work done in trying to understand the user's need, by using her past search/click history or by clustering of results in categories and allowing the user to select one section and view further results in that category (*Vivisimo*[1]). But these approaches have limitations in understanding the user's interest and the semantics behind the keyword she is searching for. 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 making the user to add terms at query time.

[1]www.vivisimo.com/

In this paper we present an approach towards modeling the user preferences in an intuitive hierarchical structure and utilize the relationships between the terms to interpret the query semantics by looking at the context of the user's interests. The rest of the paper is organized as follows. In section 2, we describe our model and discuss its features. In section 3, we describe our experiments and discuss the user evaluation results. In section 4, we discuss the related work and conclude in section 5.

2 Preference Model

We propose an approach to model the user preferences as a hierarchical structure, which we define as *Preference Tree*. Formally we define it 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 directed edges on $\mathcal{V}$
- $\mathcal{L}$ is a finite set of labels

Each node, $n_i \in \mathcal{V}$, is arranged in the *Preference Tree* in a hierarchical order of relationships and each directed edge, $e_{i,j} \in \mathcal{E}$, connecting the nodes n_i and n_j, indicates the parent-child relationship between them. Each child node represents a concept defined under the scope of the concept represented by its parent node. For each node $n_i \in \mathcal{V}$ there exists one and only one label $l_i \in \mathcal{L}$, which we define as *preference-tuple*s. Each *preference-tuple* has a structure $\langle name, \omega, \vec{p}_{name} \rangle$, which is a combination of an atomic term (*name*), a real number (ω) indicating the user's interest in the concept represented by the atomic term, which we define as the *preference-weight* and a vector of terms ($\vec{p}_{name}$), which we define as the *preference-vector*. For example, a sample *preference-tuple* looks like - $\langle Italy, 0.4, \vec{p}_{Italy} \rangle$, which means that the user has a preference for *'Italy'*, the ω-value of 0.4 is the user's *preference-weight* for this concept (similar in notion to the *degree of interest* in [10]) and $\vec{p}_{Italy}$ is the *preference-vector* for this node and stores the hierarchical information about its ancestors. (We shall discuss in detail these two notions in the next paragraphs). These *preference-tuples* are placed along the nodes of the *Preference Tree*, which are arranged in a manner such that,

the higher a node appears in the tree, the more general is the concept it represents and 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 [8] and ontology elicitation [7] 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 that particular node. In our model, the tree nodes have *preference-tuples* as their labels and hence, following the formalism in [8], when a node B appears below a node A, there is a parent-child relationship between A and B, making the child node more specific in the concept represented by its parent node. In this manner, starting from a particular node in the hierarchy, the semantics of the user's preferences are implemented inherently in our *Preference Tree*. We store this semantic relationship using a simple conjunctive propositional formula, which we define as the *preference-vector* whose elements are the nodes in the path towards the root. Our *Preference Tree* is analogous to the lightweight ontologies we come across in our day-to-day workplace [7]. Tree structures such as the file directory structure on our computers, the various email folders in our mail clients, the web directories etc can be considered as examples of *Preference Trees*.

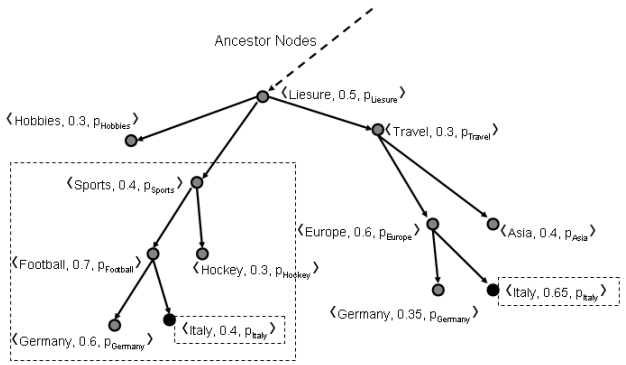

Figure 1: Sample Preference Tree

Relative Preferences

Consider a particular section of a user's Preference Tree, shown in Figure 1, describing her interests in sports. The *preference-tuples* are placed on the nodes as depicted. $\omega_{Germany}$ and ω_{Italy} (0.6 and 0.4 respectively in the figure) are *preference-weights* indicating the user's relative preference between the sibling nodes. The assigning of numeric values to preferences is built on the notions suggested in [9], [5], [4]. 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 user's relative preferences among the sibling nodes. In our Preference Tree model, we assume few properties for the *preference-weights* $(\omega - values)$.

- for all the *child nodes* of node `"k"`, the ω values are normalized 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 *preference-weights* is done to convey the user's relative preference between them.
- if $\omega_A < \omega_B$, this indicates that the user prefers $preference_B$ to $preference_A$. Referring to the above example, $\omega_{Italy} < \omega_{Germany}$, i.e. the user gives higher preference to Germany than Italy (in the context of 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.

Semantics of User's Interests

In the above discussion, we claimed that the semantic information of a preference is stored inherently in the hierarchical structure we have proposed. Similar to the Conjunctive Normal Forms suggested by the authors [8], we store the relationship between a node and its ancestors in the form of a vector, which we term as the *preference-vector*. Considering the section of *Preference Tree* illustrated in figure 1, we observe that the 'preference term' `Italy` appears at more than one place. For the purpose of discussion, consider the two occurances of `Italy` under the different paths of `Leisure` → `Sports` → `Football` → `Italy` and `Leisure` → `Travel` → `Europe` → `Italy`. Since their positions in the tree are different, their semantic meanings are also different from one another i.e. the user's preference for 'Italy' is different for the various concepts of 'Travel' and 'Sports'. When the user is searching for 'Italy', then it would be optimum if the retrieved results reflect the similar variance. We capture these semantics of the user's needs in our *preference-vector* which is built on the formalism proposed by Giunchiglia et al (recollect the notion of *the concept of a node* from the earlier discussion [8]) and calculate the logical formulas corresponding to these nodes as [`Leisure` ∧ `Sports` ∧ `Football` ∧ `Italy`] and [`Leisure` ∧ `Travel` ∧ `Europe` ∧ `Italy`]. As we read these formulas from left to right, the preferences become more specific. We utilize these formulas 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, `Leisure` → `Sports` → `Football` → `Italy`, the *preference-vector* is computed 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 of the Preference Tree which are having terms closest in similarity to the given keyword[2]. For the

[2]For query terms not appearing in the Preference Tree, we can refer an oracle (e.g WordNet) to obtain similar terms

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.

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, which 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 along the *Preference Tree*, with the arrow direction indicating the `parent` $\rightarrow$ `child` relation, $\texttt{A}[\omega_A] \rightarrow \texttt{B}[\omega_B] \rightarrow \texttt{C}[\omega_C] \rightarrow \texttt{D}[\omega_D]$, the scaled *preference-vector* for node `D` represented as $[(A, \omega_{A'}), (B, \omega_{B'}), (C, \omega_{C'}), (D, \omega_D)]$ is calculated in this manner:

- 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. `Leisure` $\rightarrow$ `Sports` $\rightarrow$ `Football` $\rightarrow$ `Italy`; the initial *preference-vector* for the node `Italy` would be [(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`.

3 Experiments

For the first set of experiments, we used the Cranfield Test Collection [3]. In addition to the 1400 documents available in this collection, we have added some more documents, obtained after crawling and parsing the text from the web pages of BBC[3] and Wikipedia[4]. This was done to add some variance to the content and themes of the documents available, so that we could test the functioning of the algorithm for different sets of preference profiles. For each of the document in the collection corpus, we created the *term frequency - inverse document frequency* vector using the classical *TF-IDF* measure [19]. We applied the `Porter Stemmer` algorithm [16], on each of these terms to strip any suffixes present. We implemented the hierarchical trees in SQL, while the user interface and the control logic was implemented in Java. When a query is made, those nodes of the user's profile are activated which are closest in sense to the query keyword and their corresponding *preference-vectors* are used for re-ranking of the retrieved results. The similarity between these *preference-vectors* and the document vectors is calculated using the classical Cosine Similarity measure [19]. As for the ranking function, we have now implemented a simple ordering based on the similarity measure.

3.1 User Evaluation

We evaluated the effectiveness of our approach by evaluating with a set of users (15 graduate students from our University Campus). 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. They were asked explicitly for their interests on a wide range of topics. The retrieved documents were re-ranked using these Preference Trees and distributed among the users for evaluation. We collected explicit relevance feedback from the users. The feedback questionnaire asked explicitly for the users opinion on the relevancy for each of the document - by marking it as either `Not Relevant`, `Relevant`, or `Highly Relevant`. The users were also asked for their general opinion on the retrieved documents, about the overall relevancy to their interests in general.

3.2 Results

All the users (with one exception) reported that the documents returned by the system were relevant to their interests (figure 2). A look into the relevancy percentage of the documents show some positive results. We observed one interesting pattern - the users, who had elicited in detail their preferences, by giving an exhaustive list of their interests and relative choices among them, had found a larger percentage of the retrieved documents to be relevant to them, as compared to those users who were very brief in describing their preferences.

[3] http://www.bbc.co.uk/

[4] http://en.wikipedia.org/wiki/Main_Page

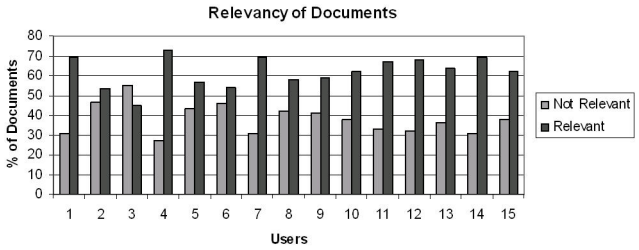

Figure 2: Relevancy of Documents

If we make a deeper analysis of the documents labeled as relevant, we find some skewed results (Figure 3). Many users found a majority of the documents to be just relevant and not highly relevant. We interpret this behavior due to two reasons.

- Firstly, our approach at present is to find documents relevant to a user by matching her preferences. We have not done any work on evaluating the *interestingness* of the document, before recommending it to her. Interestingness of a document is a measure of the information content or quality of information contained in the document, and is an altogether different research problem in the data mining community.
- Secondly, the nature of the Test Collection we have used could have influenced the results. The Cranfield Collection is not very huge and lacks in thematic variety. The pages we added to the collection, after having crawled from the websites mentioned were not sufficient in adding some diversity to the content of the collection.

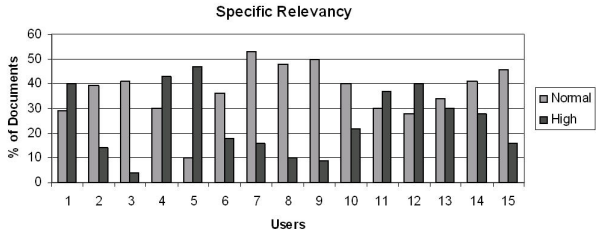

Figure 3: Specific Relevancy

Hence in a second set of experiments we implemented our algorithm over the search results retrieved by a search engine. We collected keyword queries from the users (whose preferences were collected a priori) and queried Google through its web search API. For each query term, we retrieved the top-300 ranked results and stored them as a collection of documents. We stored the snippets of the retrieved results as contents of the documents. We believe the summary of a web page reflects its contents and is a sufficient metric for its classification [13]. We constructed the *tf-idf* vectors for these documents and by measuring their similarity with the *preference-vectors* from the user's *Preference Tree* we re-ranked these top-300 results. We compared the two sets of rankings using the Precision@N metric. Precision@N indicates the precision values for N number of retrieved results. We observed that in the case of simple ranking of Google, the precision values were not good initially, but were improving monotonically as the result set increased (figure 4). While re-ranking the results by utilising our *preference-vector* rated all the results relevant to the user to a higher rank and thus achieving precision values close to unity (1.0) in the initial sets of results itself. The user's opinion was sought if these revised results indicated their preferences for that keyword. 80% of the users agreed that the revised rankings reflected their choice of documents for the given keyword query. Thus, by re-ranking the results retrieved from a search engine using our *preference-vector*, the system could give a higher rank to the documents/webpages which are more relevant to the user's interest in that context.

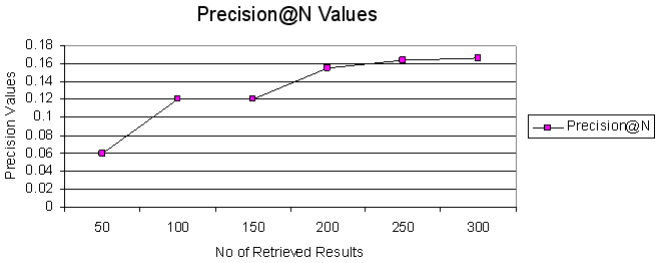

Figure 4: Precision@N results

4 Related Work

Personalized systems and services have two major aspects involved - (i)Preference Elicitation or capturing the user's interests and preferences and (ii)Preference Modeling or managing the knowledge of the user's preferences obtained a priori by the system. Capturing of the user's interests (through explicit feedback from the user or applying implicit interest indicators), is a vast field of research in itself. There have been several approaches towards capturing the user's interests such as [2], [6], [20]. Our *Preference Tree* can be used to model the knowledge of the user's preferences obtained by the above mentioned methods and the *preference-weight* values can be calculated by the techniques described by them. For example, by observing the user's browsing history (as in [6]) we can modify a reference ontology (eg. ODP[5]) and construct our *Preference Tree*, assign *preference-weights* to the nodes and construct the *preference-vectors*.

Most of the preference modeling approaches use a flat '*bag-of-words*' kind of profile [1], [11], [14], [21]. There have been other approaches towards modeling the user's preferences in a hierarchical structure [10], [17], [18]. Storing the user profiles in a hierarchical manner, gives us many advantages over a flat structure, as we can utilise the inherent relationships between the nodes of

[5]http://dmoz.org

the hierarchy to better understand the user's interest on a given context.

Our personalization technique uses the same notion that query augmentation and result processing are the primary ways to personalize the search for an active user [15]. Our approach is orthogonal to theirs as we combine the process of query augmentation (using the atomic preferences stored as node labels) and result processing (by utilising the *preference-vector*) to interpret the semantics of the user's information need more effectively. Our approach differs from the other hierarchical approaches [10], [17], as we introduce certain metrics that help us to disambiguate the relative preferences between the sibling nodes of the hierarchical user preference model. Our model also provides a heuristic (*preference-vector*) to understand the semantics of the keyword from the concept represented by its position in our *Preference Tree*. Our work is complimentary to the works of [6] and [12], in the sense that our *Preference Tree* could be used to model the user knowledge obtained by the approaches defined in their work. Their approach may appear similar to us, as they also employ weights to indicate the user's preferences. But the major difference is that they build a vector or matrix of terms and assign weights to these terms under each category in the user's profile, while our *preference-vector* is built using the relationship between a node and its ancestor nodes in the *Preference Tree*. The scaled-hierarchical relationship encoded in our *preference-vector* helps us in resolving the ambiguity arising out of duplicity in the user's preference terms and a better understanding the semantics of the user's query.

5 Conclusion

We have presented an approach towards modeling the user preferences in a hierarchical structure what we term as *Preference Tree*. We have introduced notions (*preference-weight* and *preference-vector*) which help us to understand the semantics of the user's information need in a better manner. From the experimental results we have observed that by re-ranking of results using our *preference-vector*, we can improve the precision performance of search engines.

References

[1] E. Benaki, K. A. Vangelis, and C. D. Spyropoulos. Integrating user modeling into information extraction: The UMIE prototype. In *Proceedings of UM*, pages 55–57. Springer, 2-5 1997.

[2] M. Claypool, P. Le, M. Waseda, and D. Brown. Implicit interest indicators. In *Intelligent User Interfaces*, pages 33–40, 2001.

[3] C. W. Cleverdon. The significance of the cranfield tests on index languages. In *SIGIR*, pages 3–12, 1991.

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

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

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

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

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

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

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

[11] K. Lang. NewsWeeder: learning to filter netnews. In *Proceedings of the 12th International Conference on Machine Learning*, pages 331–339. Morgan Kaufmann publishers Inc.: San Mateo, CA, USA, 1995.

[12] F. Liu, C. Yu, and W. Meng. Personalized web search for improving retrieval effectiveness. *IEEE Transactions on Knowledge and Data Engineering*, 16(1):28–40, 2004.

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

[14] D. Nichols. Implicit rating and filtering. In *Proceedings of 5th DELOS Workshop on Filtering and Collaborative Filtering*, pages 31–36. ERCIM, 1998.

[15] J. Pitkow, H. Schütze, T. Cass, R. Cooley, D. Turnbull, A. Edmonds, E. Adar, and T. Breuel. Personalized search. *Commun. ACM*, 45(9):50–55, 2002.

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

[17] A. Pretschner and S. Gauch. Ontology based personalized search. In *ICTAI*, pages 391–398, 1999.

[18] J. Rucker and M. J. Polanco. Siteseer: personalized navigation for the web. *Commun. ACM*, 40(3):73–76, 1997.

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

[20] J. Teevan, S. Dumais, and E. Horvitz. Personalizing search via automated analysis of interests and activities, 2005.

[21] T. Yan and H. Garcia-Molina. SIFT—A tool for wide-area information dissemination. In *Proc. 1995 USENIX Technical Conference*, pages 177–186, New Orleans, 1995.