Weigh your Preferences!
Towards using hierarchies for building personal libraries

Hamza Hydri Syed, Periklis Andritsos

October 2006

Technical Report # DIT-06-062

Weigh your Preferences! **:** **Towards using hierarchies for building personal libraries**

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

Abstract

The existence of a vast amount of document collections both on-line or off-line leads to an overwhelming experience when users attempt to query them. This emphasises the need to develop collaborative approaches for acquiring data/knowledge from distributed and semantically heterogeneous sources of knowledge/information. Work on user preferences has recently acquired more attention in order to retrieve and rank documents which are relevant to the user's interests. In this paper, we present a hierarchical modeling of a user's interests/preferences. The proposed hierarchical structure results in a better semantic handling of the users' interests and proper matching of documents over these nodes representing the user preferences. Our preliminary experiments show the merits of such an approach and open new avenues of research in this area.

1 Introduction and Problem Description

Alice is a graduate student in Medical Sciences with interests in gardening and cooking. Bob is a photographer, who loves traveling in Europe and is a big football fan. Imagine a situation when both of them are browsing an online digital library. Alice would be pleased to find the system recommend her books/documents/journals related to her interests, while Bob would prefer the system to reflect his interests. The whole scenario is similar to searching or retrieving documents for a user, from a large noisy data set, having documents from a large domain of topics/themes/subjects. In such a case, the performance of the system would be more appreciable if it reflected the users preferences and react in a personalised manner to the user queries. The relevancy of the recommended documents to the user, would be very good if the system had prior knowledge of the user's interests.

In an IR system, the user is without doubt the most essential part of the system and implementing User Preferences in the Recommendation Systems or Data Retrieval/Search Engines has attracted a good amount of research in the recent times. In this paper, we present our approach towards implementing user preferences and make use of a running example to explain the advantages of this approach. In Section 2 we survey some related work done and in Section 3 we present our approach and elicit its finer details. In Section 4 we describe our implementation with a test collection and report some results of user evaluations in Section 5. In Section 6 we conclude with some recommendations of future work.

2 Related Work

Query personalization and modeling of user's needs have attracted interest in both IR and Database research communities. Koutrika and Ioannidis define a preference model for personalization of queries in database systems [12]. The authors present a framework which makes use of structured user profiles(storing simple, unconditional preferences) for personalizing the database queries. While the rewriting of the queries needs some optimization work, under the constraints of running time, this work provides one of the first solutions towards a good integration of databases and user personalization. Satisfying user needs could be done in an efficient manner by implementing user preferences in a Database System, at the same time personalization of the services requires a flexible modeling technique for user preferences.

There is a strong contrast in the Database queries, which are characterized by hard constraints, delivering the objects if available or rejecting the user's request and the "real world" where the personal preferences behave differently. The preferences in the real world negotiate for the best possible match and are hence treated as soft constraints as in [11]. Kießling et al propose a model which treats preferences as strict partial orders. They present several constructors and algebraic laws for modeling of preferences. To bypass the problems posed by *empty-result, exact-match query models* and *flooding effect* in e-commerce applications, a "Best Matches Only" (BMO) query model is proposed, which makes the suitable matches between wishes(preferences) and reality [11].

A similar framework has been suggested by Chomicki in [6], where preferences are specified using first-order logical formulas and embedding them into relational algebra by introducing a new operator called *winnow* that selects the most preferred tuples. The author focuses mostly on intrinsic preferences, based only on values occurring in tuples. The semantics of the *winnow* operator are similar to the BMO Query model. Several properties of preferences (intrinsic, extrinsic, etc.) including iterated preferences are discussed, however it does not mention about Pareto preferences as in [11]. The framework of [1] uses quantitative preferences in queries and discusses the issue of combining various user preferences. However, it does not provide any description of the semantics for these queries. As an extension to their work [7], presents few semantic optimization techniques for preference queries to remove redundant occurances of their *winnow* operator. But application of these techniques is limited to the authors previous work and does not cover several aspects, for example the referential integrity constraints. our work is orthogonal to the work done on utility elicitation techniques, such as the one by [5], where the authors tackle the problem of eliciting the user utility function as a classification problem.

3 Modeling of User Preferences

In [15], the authors list out the key issues related to implementing preference-based search systems. Among these, in this paper, we deal with the *preference modeling* assuming that the system has prior knowledge of users preferences.

We consider a hierarchical structure to model the user preferences (as shown in Figure 1). 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. In our Preference Tree model, each of the node consists of a tuple - $\langle name, \omega \rangle$.

- where *'name'* is the preference name, and 'ω' is the *preference-weight*, which we introduce as a measure of relative preference among the siblings of a particular node

For example consider a section of a user's preference tree as shown in figure 2. The hierarchy of preferences for `Germany` and `Italy`, both being sibling nodes with common ancestors `Sports` and `Football`, are as shown

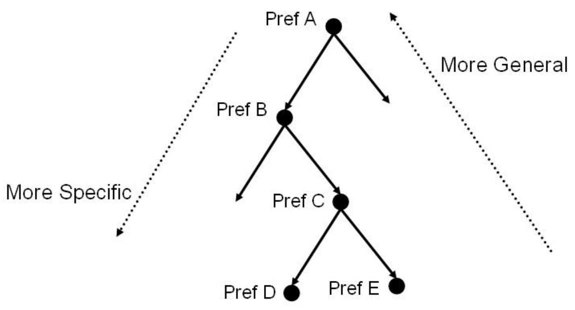


Figure 1: Model Preference Tree

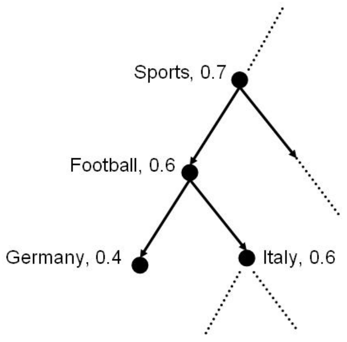


Figure 2: Sample Preference Tree

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

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

where $\omega_{Germany}$ and ω_{Italy} (0.4 and 0.6 respectively in the figure) parameters give an indication of the relative preference between them.

ω **- Properties:** for all the sibling nodes on one level the ω values are normalised to 1.

- $\sum \omega_i = 1$ where ω_i is the weight for a node $"i"$. We do this to normalize the sibling preferences, to quantize the relative preference among them.
- if $\omega_A < \omega_B \Rightarrow$ the user prefers B as compared to A. Referring to the above example, $\omega_{Germany} < \omega_{Italy}$, i.e. the user gives higher preference to the Italian Football team as compared to the German Football team, though both these preferences are of some interest to the user.
- the preference weight for a node ω_A is independent of the preference weights of its parents or children and is only related to 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, as the preference weight of its parent (Football), would not influence in anyway the relativeness of sibling preferences.

These user preferences are stored in a database and are loaded into the runtime environment while doing similarity matching with the documents from the collection corpus. Further issues regarding how we implement it are dealt with in the next section.

3.1 Preference Weight Vector

We claim that the semantic information of a preference is stored latently in the hierarchical architecture we propose, as our ideas are based on the ones proposed by [9] and [10]. Similar to the Conjunctive Normal Forms suggested by the authors, we store the relationship between a node and its ancestors in the form a *preference-weight-vector*, which is stored along each of the nodes in the tree. Considering the example illustrated in figure 3, let us look at the two different sections of the preference tree having a common root node.

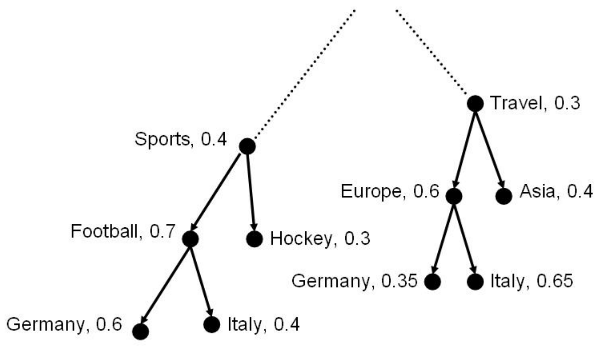


Figure 3: Sample Preference Tree

The node `Italy` under the path
`Sports` ⟶ `Football` ⟶ `Italy`
is semantically different from the node `Italy` under the path
`Travel` ⟶ `Europe` ⟶ `Italy`,
since their position in the tree and hence their semantic meanings are different from one another i.e. if we follow the formalism proposed by Giunchiglia et al. the logical formulas corresponding to the node would be

[...$\texttt{Sports} \wedge \texttt{Football} \wedge \texttt{Italy}$]

[...$\texttt{Travel} \wedge \texttt{Europe} \wedge \texttt{Italy}$]

With this approach it becomes easier to distinguish between documents having the same terms but having totally different semantic meanings. For instance, a document is describing some touristic information about Italy, with some travel tips etc. The term `Italy` appears prominently in its text. Usually with the regular document matching methods involving similarity measure of vectors, this document would find a match on the node `Italy` under `Sports` as well as at the node `Italy` under `Travel`. But with our approach, it becomes easier to disambiguate the semantic sense, perform proper matching and place this document at the node `Italy` under `Travel`. In this way, if the document matching is done using the preference weight vector, it becomes easier for the algorithm to distinguish between two documents having `Italy` as a prominently appearing term, but both being different semantically - one describing the travel tips for Italy and the other describing the Italian football team.

For any given node in the preference tree, we store this preference weight vector (containing the given node term along with its ancestors) at each node of the tree.

One would argue that while doing similarity matching between the preference weight 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 weight vector of a particular node. For this reason we introduce the scaling of the ancestor ω-values for a given node.

3.2 Scaling of ω values

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

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

if node `B` is a parent/ancestor then its scaled *preference-weight* is

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

where m $\geq$ *2*, (for the experiments in this work, we consider *m* = 2, we are currently studying the relation of *m* over the retrieved results and shall report any interesting observation in our follow-up work)

k is its path distance, along the tree, from the node `A`

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.

For each step `k` one goes up the tree along the path of the ancestors, the scaled weight is $1/2^k * \omega_A$ 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

... $\mathtt{A}[\omega_A] \longrightarrow \mathtt{B}[\omega_B] \longrightarrow \mathtt{C}[\omega_C] \longrightarrow \mathtt{D}[\omega_D]$

so the *preference-weight-vector* for the node `D` looks like

$[..., A = \omega_{A'}, B = \omega_{B'}, C = \omega_{C'}, D = \omega_D]$

this is how the calculations are done:

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

- for node C, $k = 1 \Rightarrow \omega_{C'} = \omega_C * 1/2^1$

In this manner the influence of the preference weight of the ancestors is slowly diminishing as we are moving away from the node. Considering an example similar to the one mentioned before.
... $\texttt{Sports}[\omega_{Sports}] \longrightarrow \texttt{Football}[\omega_{Football}] \longrightarrow \texttt{Italy}[\omega_{Italy}] \longrightarrow \texttt{Buffon}[\omega_{Buffon}]$

the *preference-weight-vector* for the node Buffon would be

$[..., Sports = \omega_{Sports'}, Football = \omega_{Football'}, Italy = \omega_{Italy'}, Buffon = \omega_{Buffon}]$

In this case the documents talking about Buffon and having semantics of Italian Football would have the highest similarity match to this node, while documents on Italian football, Football in general or Sports in general sense 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 occuring caused by the ancestor terms, while doing similarity matching with term weight vectors for the documents.

4 Experiments

For our implementation work, we adapt the Model-View-Controller[1] paradigm - separating the application's data model, user interface and the control logic into three distinct components so that modifications to one component can be made with minimal impact to the others. The Model encapsulates the core application data and functionality, the View obtains data from the model and presents it to the user and the Controller receives and translates input to requests on the Model or the View.

We made use of the Cranfield Test Collection [8] to test our algorithm. In addition to the 1400 documents available in this collection, we have added some more documents, obtained after crawling and parsing the text of the web pages collected from the websites of BBC[2] and Wikipedia[3]. 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 create the term frequency - inverse document frequency vector using the classical *TF-IDF* measure [14], where wgt of a term in the particular document is given by

$$wgt = tf * idf$$

where

$$tf = \frac{n_i}{\Sigma_k n_k}$$

n_i is the frequency of occurance for the i^{th} term and k is total number of terms appearing in the document.

$$idf = \frac{\log|D|}{|(d_i \supset t_i)| - |D|}$$

where $\log|D|$ is the total number of documents in the corpus, $|(d_i \supset t_i)|$ is the number of documents in which the term appears.

[1]http://www.phpwact.org/pattern/model_view_controller
[2]http://www.bbc.co.uk/
[3]http://en.wikipedia.org/wiki/Main_Page

We apply the `Porter Stemmer` algorithm [13], on each of these terms to strip any suffixes, if present.

The user preferences were obtained explicitly from the users and stored in the form of a hierarchical Tree structure. We implemented the hierarchical trees in SQL, using the Adjacency List Model prescribed by [4]. The SQL tables containing the tree nodes and their relations were stored in a PostGres database.

The user interface and the control logic was implemented in Java. At present we have an elementary user interface. In this paper we are introducing our inceptual ideas, 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.

The pseudo code of our algorithm is as described below. The $CollectionCorpus$ is our Test Collection as described above. $PreferenceTree$ is the hierarchical tree of user preferences which is loaded from the data base into the Java environment. 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 ranked list of documents which are relevant to the users interests.

Algorithm 1 Weight-based Preference matching

```
Input: $PreferenceTree, k$
Output: Matching documents
  for all $document \in CollectionCorpus$ do
    form the $term - frequency - vector$
    for all $term \in term - frequency - vector$ do
      apply Porter Stemmer to remove any suffix
    end for
  end for
  for all $node \in PreferenceTree$ do
    get the ancestors
    form the $preference - weight - vector$
    scale the $\omega$ values of the ancestors
    perform similarity measure with the $term - frequency - vectors$
    rank documents with highest match
    return top-$k$ documents
  end for
  return overall ranked list of documents for the entire $PreferenceTree$
```

The similarity measurement is done using the classical cosine similarity given by the formula

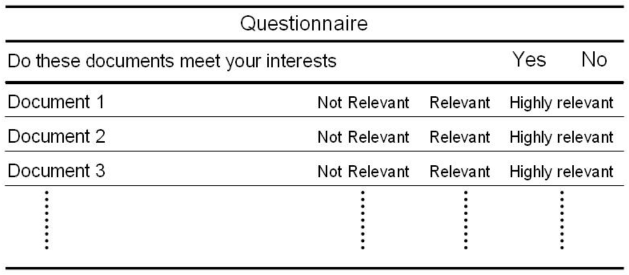
$$sim(pwv,tfv) = |pwv \bullet tfv| / |pwv| * |tfv|$$

where *pwv* and *tfv* are $preference-weight-vector$ and $term-frequency-vector$ respectively

As for the ranking function, we have now implemented a simple ordering based on frequency of occurence. Currently we are exploring some better ways to rank our retrieved results, based on the ideas proposed in [2] . In our follow up work, we shall implement a better ranking algorithm.

5 Evaluation with Users

We evaluated the effectiveness of the results retrieved by our algorithm, by doing a study involving 15 users[4]. The users were asked explicitly for their interests and preferences on a wide range of topics. These preferences were later stored as personal preference trees for each of the users and given as an input to the retrieval system. The documents retrieved by the system were ranked and distributed among the users for evaluation. We asked for their feedback on the relevancy of these retrieved documents to their interests.



Questionnaire			
Do these documents meet your interests		Yes	No
Document 1	Not Relevant	Relevant	Highly relevant
Document 2	Not Relevant	Relevant	Highly relevant
Document 3	Not Relevant	Relevant	Highly relevant
⋮	⋮	⋮	⋮

Figure 4: Questionnaire

The feedback questionnaire was as depicted in the figure above, asking 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. The evaluation results show some interesting results.

5.1 Results

All the users (with one exception) reported that the documents returned by the system were relevant to their interests (as depicted in figure 5). 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.

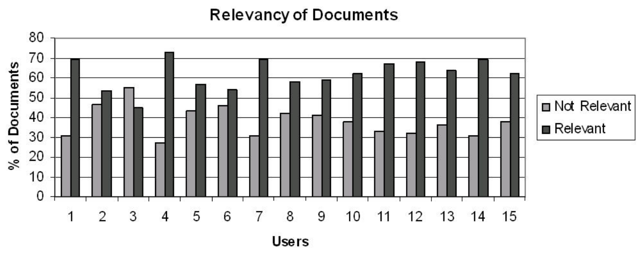


Figure 5: Relevancy of Documents

[4]graduate students from our University Campus

If we make a deeper analysis of the documents labeled as relevant, we find some skewed results (as seen in Figure 6). 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 in this paper is to find documents relevant to a user by matching her preferences. We are not doing 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. For instance, in one user evaluation feedback, the student found most of the documents were not highly relevant to her. According to her, the information content in the recommended documents was very general in tone, as she was not gaining any new information. Though these documents were relevant to her preferences, she did not find them to be of high information value. (In this case, since this student was passionate about photography, she said that though the documents about photography in general were relevant to her, they did not give any detailed information about any specialised form of photography). However, we make a note of it as a probable extension for our work. Since this paper presents our introductory work, we leave this issue for future work.

- Secondly, the nature of the Test Collection we have used could have influenced the results. The Cranfield Collection is not very huge and does not cover a variety of themes. The pages we added to the collection, after having crawled from the websites mentioned were also very general in content, with not much specialised information. We think by adding a much larger number of documents to our collection, or probably by retrieving results by searching on the Internet using the user preferences (which is one of our major extensions for future work), we can retrieve documents which would be highly relevant to the user.

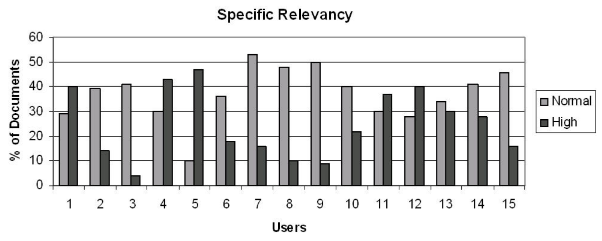


Figure 6: Specific Relevancy

6 Conclusion and Future Work

This is our preliminary work in this direction and the implementation discussed is inceptual if not comprehensive. We present our work as a proof-of-concept in this paper.

As discussed, the existence of a vast amount of document collections leads to an overwhelming experience for users who attempt to browse through them. Work on user preferences has recently acquired more attention in order to retrieve and rank the documents relevant to a user, from a large corpus of documents. We have presented an approach based on hierarchical modeling of a user's interests/preferences.

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 choice between common interests (which appear as sibling nodes in the tree). The results and evaluation study with users, strengthen the claim that our proposed hierarchical structure efficiently handles the semantics of the users' interests and performs proper gathering/matching of documents over these nodes. 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 [3], 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. As an extension work, we are now looking into ways and methods to capture the users preferences implicitly, implement the whole system over an existing search engine and perform comparative analysis with any existing state of the art preference handling systems. For this purpose we are now working on a GUI interface which the user can interact and the results can be displayed to her. We also need to look into the scalability issues, and optimize the run-time efficiency of the retrieval engine.

We want to expand our test collection and implement our algorithm over a much larger collection of documents. As for the evaluation of the algorithm, we need to test it 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.

The overall ranking of the total retrieved documents should also reflect the relative hierarchy of preferences among the users interests. We are exploring methods to implement this intuition, such that the hierarchy of the user preferences is maintained in the overall retrieved document set. As mentioned earlier, we also hope to implement some techniques to improve the *interestingness* of the content in the document labeled as relevant to the user.

References

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

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

[3] 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.

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

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

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

[7] J. Chomicki. Semantic optimization of preference queries. In *CDB*, pages 133–148, 2004.

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

[9] 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.

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

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

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

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

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

[15] P. Viappiani and B. Faltings. Design and implementation of preference-based search. In Springer, editor, *The 7th International Conference on Web Information Systems Engineering*, LNCS4255, Wuhan, China, October 2006.