An Intelligent Knowledge Sharing System for Web Communities

C. Bauckhage, T. Alpcan, S. Agarwal, F. Metze
Deutsche Telekom Laboratories
10587 Berlin, Germany
http://www.telekom.de/laboratories

R. Wetzker, M. Ilic, S. Albayrak
DAI Labor, Technical University Berlin
10587 Berlin, Germany
http://www.dai-labor.de/

Abstract— **This paper presents the prototype of an expert peering system for information exchange in the knowledge society. Our system realizes an intelligent, real-time search engine for enterprise Intranets or online communities that automatically relays user queries to knowledgable specialists. According to its very nature, the system requires a sound integration of concepts drawn from various areas of Computer Science. In addition to our solutions to problems in scalable processing, data transfer, and networking, we also address issues of interface design and usability, as well as aspects of machine intelligence. Results obtained from extensive experiments demonstrate the efficiency and robustness of our system and convey its potential for next generation web services.**

I. MOTIVATION AND SYSTEM OVERVIEW

With the coming of age of Internet services conveniently labeled *Web 2.0 technologies* [1], there is a rapidly growing number of interactive portals that empower individual users or whole communities to provide content and thus allow for the exchange of information at a scale previously unheard of. However, this further adds to the familiar problem of finding the right information from an abundance of data. Therefore, Web 2.0 compliant and user friendly technologies are called for that enable the efficient retrieval of relevant information.

Addressing this need with respect to a knowledge management application, we present a novel system for knowledge sharing within web communities. It combines features of search engines, blogs, chats, and peer-to-peer technologies and extends them by using methods from machine learning, networking, usability and privacy management. Upon registration, users of our system identify themselves as experts on certain topics. Once registered, they can either address the community with problems they need assistance with, or can respond to other users' questions. Rather than mediating the communication between community members by an forum-style mechanism, our system automatically analyzes user queries and proposes appropriate experts who then can be contacted directly via, for example, instant messaging.

In order to ensure Web 2.0 compliancy and user friendliness, our implementation follows three basic paradigms. Firstly, we consider the web as the platform and the user's browser as the client; secondly, the system has a flexible and extendable architecture and allows for personalized interfaces; thirdly, it incorporates pattern recognition techniques enabling intelligent retrieval. As seen in Fig. 1, the system therefore consists of two major parts: the user's web browser as the client and a central web application server. The Ajax framework [2] on the browser provides the necessary graphics and communication capabilities allowing us to use the web browser as a client in our system. The main advantages of this approach include utilization of an already available, well-known infrastructure and interface, instantaneous and dynamic installation, and significant cross-platform support. The web application server is realized using the TurboGears framework [3] which allows for rapid application development by exploiting the Python scripting language [4]. Additionally, our server-side framework integrates a standard database, a web server, and the unique and novel matching engine described in Section II.

With respect to ease of use and user comfort, our application domain also requires to deal with problems of usability and network latency. Section III therefore briefly summarizes our approach to designing the interface of our system and section IV sketches how we optimize data transfer over the network. A summary and an outlook to future developments will close this contribution.

II. INTELLIGENT QUERY PROCESSING

In this section, we first present an ontology-based semantic model that allows for very efficient query-expert peering and then discuss performance characteristics of our approach.

A. Computational Model and Approach

Our approach to query-expert peering requires an ontology tree T whose nodes $\mathcal{N} = \{n_1, \ldots, n_N\}$ correspond to different subjects. Apart from this structural assumption, our system is independent of the nature and content of the ontology considered in any implementation.

The fundamental idea is to represent experts as well as user queries by means of subtrees of the ontology (see Fig. 2). The problem of query-expert peering thus becomes a problem of tree matching which can be dealt with efficiently if the ontology tree was serialized. Tree serialization is achieved by ordering the nodes from top to bottom and left to right, leading to an associated vector representation $\boldsymbol{v}(T) \in S(T)$ where $S(T) \subset \mathbb{R}^N$ is the *ontology-space*.

Once the registered experts $e_1, \ldots, e_E$ and an incoming query q have been mapped to subtrees, we determine associated vectors $\boldsymbol{v}(e_i)$ and $\boldsymbol{v}(q)$ whose entries are set to 1 or 0 depending on whether or not the corresponding tree node is related to the expert's knowledge or the query, respectively. We then compute a matching between the query and each expert using the weighted dot product

$$m(q, e_i) = \boldsymbol{v}(q)^{\mathrm{T}} \boldsymbol{W} \boldsymbol{v}(e_i) \tag{1}$$

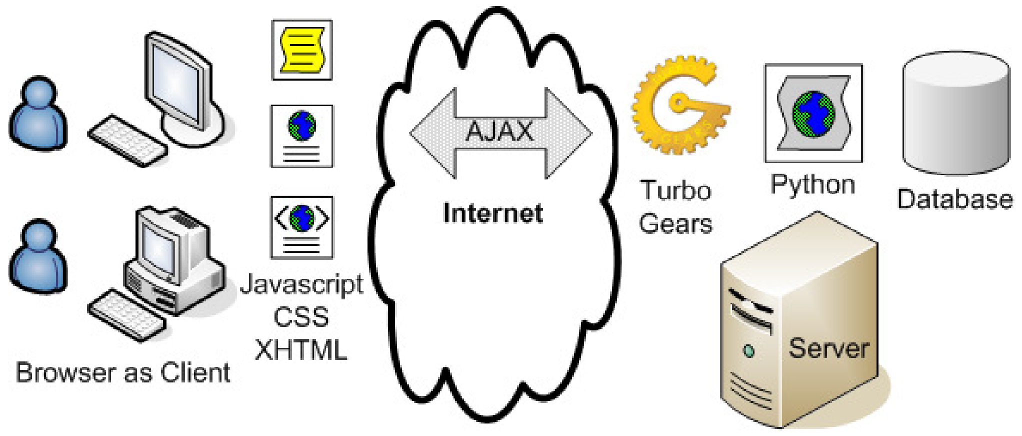


Fig. 1. Schematic sketch of the components and architectural layout of our system for delegating user queries to knowledgeable experts.

where the weight matrix $\boldsymbol{W}$ allows for incorporating further contextual relations among the topics of the underlying ontology. Subsequently, the query will be assigned to those experts with the highest ranking matchings.

While the ontology space $S(T)$ allows for using well established methods from linear algebra, its dimension is considerably smaller than the ones of the commonly used term-by-document spaces. Moreover, our approach avoids maintaining inefficient, large, and static dictionaries of possible query terms if we consider the following efficient procedure for mapping experts and queries to subtrees: In an offline preparation step, we first associate each node n of the tree with a corresponding *bag of words* $\mathcal{B}(n) = \{w_1, \ldots, w_{B_n}\}$ harvested from Internet resources related to the subject. Subsequently, we optimize all bags of words in the ontology both vertically and horizontally in order to emphasize the hierarchical nature of our data structure and to reduce redundancies, respectively. In the vertical direction, we find the union $\tilde{\mathcal{B}}(n)$ of the bag of words of a node n and the ones of its children, i.e. $\tilde{\mathcal{B}}(n) = \mathcal{B}(n) \cup [\cup_{i \in \mathcal{C}(n)} \mathcal{B}(i)]$, where $\mathcal{C}(n)$ denotes the set of immediate descendants of n. Then, we replace $\mathcal{B}(n)$ with $\tilde{\mathcal{B}}(n)$. Starting with the nodes above the leafs, this is repeated in a bottom-up manner until the root node is reached. In the horizontal direction, we find the overlapping words among all children of a node n, $\overline{\mathcal{B}}(n) = \cap_{c \in \mathcal{C}(n)} \mathcal{B}(c)$, subtract these from each $c \in \mathcal{C}(n)$ such that $\mathcal{B}(c) = \mathcal{B}(c) \setminus \overline{\mathcal{B}}(n)$, and repeat this for all $n \in \mathcal{N}$.

Obviously, expert profiles (provided by means of resumes, webpages, blogs, etc.) and user queries alike can be converted into bags of words, too. Mapping experts and queries to the ontology therefore requires a similarity measure $r(i,j)$ that compares two entities i and j given their respective bags of words $\mathcal{B}(i)$ and $\mathcal{B}(j)$. Unlike in previous work, where we considered angles between word occurrence vectors [5], our current implementation adopts the *term frequency/inverse document frequency* (TF/IDF) measure from information retrieval [6]. Term frequencies are measured with respect to individual subject nodes, while inverse document frequencies are computed taking into account all children of a node, too.

Given a suitable similarity measure, the following procedure efficiently maps an entity i, i.e. an expert or a query, onto the ontology-space:

1) consider the root node n of the ontology tree T
2) for the children $n_{c_1}, \ldots, n_{c_C}$ of n, compute the similarities $r(i, n_{c_j})$
3) compute the mean μ and the standard deviation σ of the resulting similarities
4) consider all nodes $\{n_k\}$ in the current set of children for which $r(i, n_k) > \mu + \alpha\sigma$, where $\alpha \geq 0$ is a fixed *branching parameter*
5) for each node n in the set $\{n_k\}$, add n to the resulting subtree, set the corresponding vector entry to 1, and continue with step 2, until the lowest level of the tree is reached

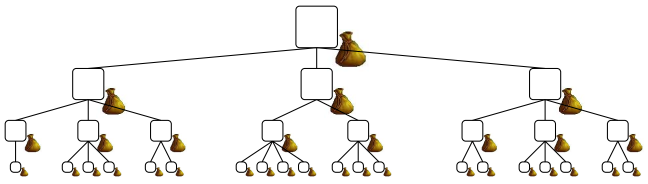
Due to its top-to-bottom iterative nature and its usage of the ontology's hierarchical structure, our algorithm is inherently robust. At each level of the tree, it solves classification problems of increasing specificity but of decreasing importance.

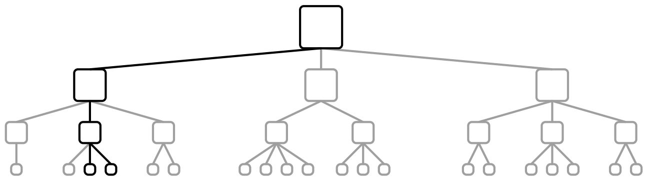
B. Experimental Setup

For a quantitative performance evaluation of our semantics-based matching of queries and experts, we conducted a series of experiments. In contrast to a preliminary study where we experimented with an ontology of about 200 nodes only [5], the integrated prototype described in this paper was tested with the database of knowledge collected by the open directory project[1].

This ontology was pruned to a tree of 1318 nodes distributed over three levels. The resulting version of the tree mainly focused on topics in the areas of business, science, and computer technology and therefore simulated the use of our system for knowledge management in enterprise environments. Several million topic related keywords were obtained from harvesting corresponding HTML documents yielded by the YAHOO search engine. They were converted into ASCII format and further processed using the Natural Language Toolkit [7] by tokenization, stop word removal, and

[1]http://www.dmoz.org


(a) Didactic example of an ontology tree where each node is associated with a corresponding bag of words obtained from a suitable collection of documents. In order to increase the robustness of expert-query matching, the lists of keywords of higher level nodes also contain the union of keywords of all their descendants.


(b) An expert on two topics represented by leafs of the ontology tree corresponds to a subtree.

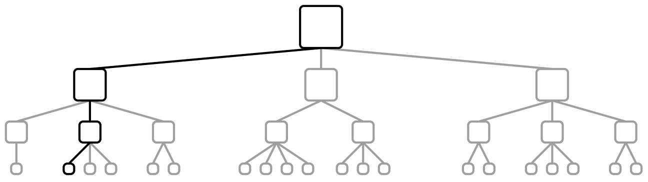
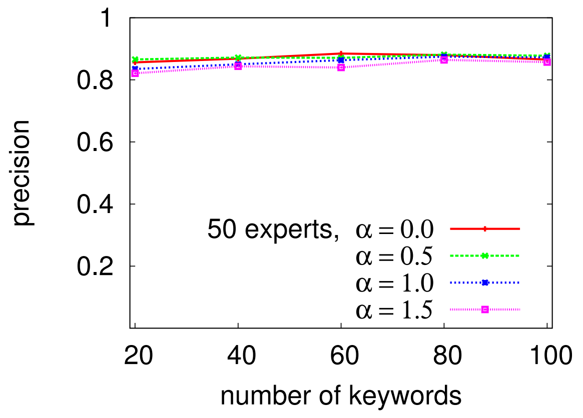
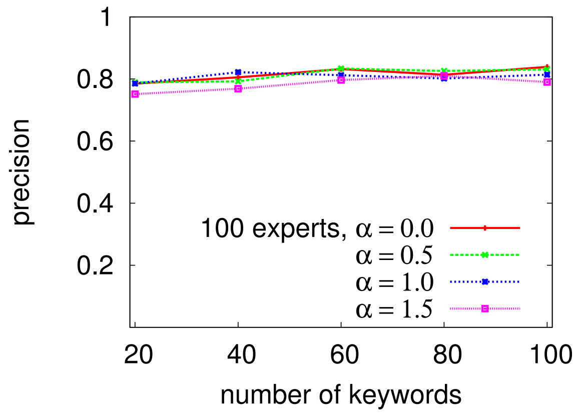

(c) A query on a topic represented by a leaf of the ontology tree corresponds to a subtree, too.

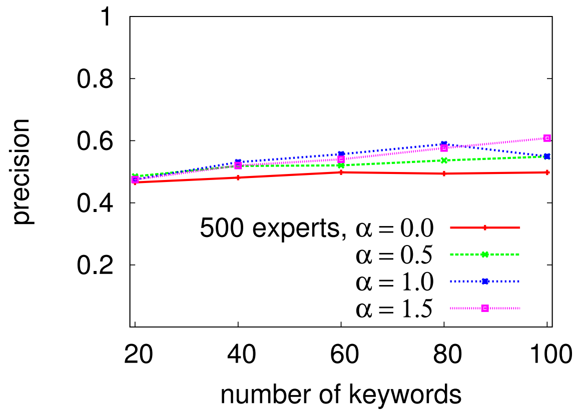
Fig. 2. Illustration of the data structure for expert-query matching and visualization of the idea of mapping experts and queries to subtrees in order to determine a suitable peering. For instance, a query on electromagnetism (a topic represented by a leave node) is, in a wider sense, a query on theoretical physics which, in turn, is a query in the area of physics in general. Therefore, even if there is no expert on electromagnetism, an expert on quantum mechanics and computational physics might be able to help.



(a) 50 registered experts



(b) 100 registered experts



(c) 500 registered experts

Fig. 3. Precision values for different parameters in query-expert matching.

stemming with Porter's stemmer [8]. 80% of the documents harvested for each node were used for populating the tree while the remaining 20% served as an independent test set.

Exemplary queries were created by randomly choosing a topic and a number of keywords from the corresponding bag of words in the test set. Mapping these to subtrees provided estimates of the associated vector representations. Since we knew to which node of the ontology the keywords belonged, we were able to provide "ground truth" vectors for each query. These were obtained by traversing the tree from the selected leaf to the root. The associated vectors of the the resulting subtrees represent the query vectors one would obtain from a set of keywords, if the automatic mapping of queries to subtrees was perfect. Also, we generated different sets of experts, where each expert was assumed knowledgable in one to five topics which were randomly selected from the leafs of the tree.

For each query q, we computed the expert peering, i.e. the set of experts $R(q)$ with the highest matching scores. The ground truth vectors served to determine the set of experts $A(q)$ that should have been returned for that query. In order to assess *how many of the appropriate experts were retrieved* and *how many of the retrieved expert were appropriate*, we averaged *recall* and *precision* over $M = 1000$ queries:

$$recall = \frac{1}{M} \sum_{i=1}^{M} \frac{| A(q_i) \cap R(q_i) |}{| A(q_i) |} \qquad (2)$$

$$precision = \frac{1}{M} \sum_{i=1}^{M} \frac{| A(q_i) \cap R(q_i) |}{| R(q_i) |}. \qquad (3)$$

We considered $k = 20, 40, \ldots, 100$ query words, the sizes of the sets of experts were drawn from $\{50,\ 100,\ 500\}$, and the branching parameter α was varied between 0.0 and 1.5. Also, in order not to be overly restrictive or too pessimistic, the set $A(q)$ was said to be given by those experts from the ground truth set whose matching scores were amongst the two highest ranking scores; the same applied to the set $R(q)$ of retrieved experts. Finally, whenever a query q yielded a matching $m(q, e_i) \leq 1$ for all i, it was discarded and replaced by another one. The server we used for our experiments was installed on a dual AMD Opteron 2214 workstation (2.2MHz) and did run a non-optimized Python implementation of our matching engine.

C. Numerical Results and Discussion

Fig. 3 displays precision values we obtained from different experimental settings. While for a fixed number of experts the branching parameter α does not seem to impact the precision achieved by our system, there is an effect correlated with a growing number of registered experts. If more experts have to be searched for appropriate ones, we observe a gradual decline in performance. However, for larger numbers of experts, larger values of α yield better performance than smaller ones. Fig. 4 indicates a similar behavior for the corresponding recall values. Again, for a fixed number of experts, the branching parameter did not exert any significant influence; we therefore only present the results for $\alpha = 1.5$ and omit the others due to lack of space.

With respect to Figs. 3 and 4, it is worth noting that the precision yielded by our matching approach consistently exceeds the respective recall. Regarding our application, this is in fact a desirable property. Consider the fact that if a high recall was the predominant performance criterion, returning *all* registered experts would be the best strategy for any query. This, of course, would lead to weariness and frustration for users who are seeking for assistance. Their chances of contacting an appropriate expert would diminish even if the experts were returned in a ranked order (frequent users of state-of-the-art search engines will be familiar with this phenomenon). However, since our system includes a high percentage of relevant experts among the ones it presents to the user looking for assistance, his or her chances of quickly finding someone who may help increase considerably.

Since the matching task becomes more difficult if more entities have to be searched, a declining performance for a growing number of experts comes with little surprise. However, choosing the larger $\alpha = 1.5$ value for our algorithm still yields precision values of up to 60%. Since this parameter affects the branching behavior in the mapping of queries to subtrees of the ontology we conclude that increasing it restricts unnecessary branching and hence oppresses noise.

From the figures we also note that, although precision and recall both slightly increase with an increasing number of query terms, the curves are rather flat. This demonstrates that, regardless of the totaly random fashion in which the queries in our experiments were created, our system performs almost independent of the amount of given information.

Fig. 5 compares average processing times obtained from different parameterizations in our query-expert matching experiments. Here, the branching parameter α noticeably influences the results even for a fixed number of experts. Although even for the less favorable settings the processing times resulting from our straightforward Python implementation do not threaten real time constraints, we note a gain in computation time if α increases. Again, this is because choosing larger values for α restricts the breadth of subtrees considered in the mapping stage. Therefore, while more readily branching (smaller values of α) leads to a computational overhead if our system has to process a growing number of query terms, restricted branching (larger values of α) barely affects processing times. Still, it yields comparable results in terms of precision and recall (see Figs. 3 and 4).

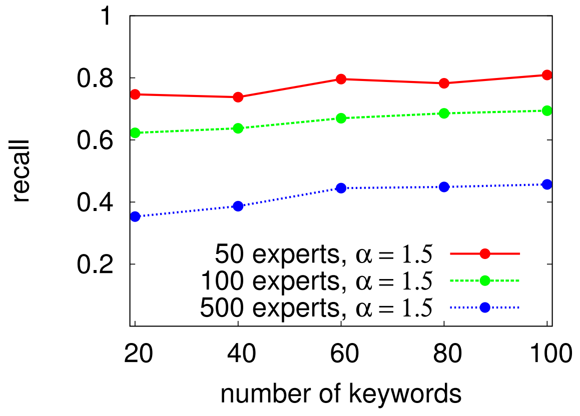


Fig. 4. Recall values for different parameters in query-expert matching.

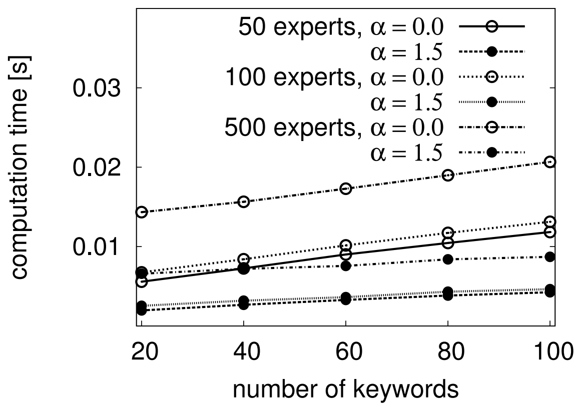


Fig. 5. Average query processing times (in seconds) for different parameters in query-expert matching.

III. Interface Design and Usability

Apart from the quality of expert-query peering, the success of our system crucially depends on the quality of the user interface. This is because the system should be *easy to use* for users of all levels of experience. Also, the interface should convey they idea of *symmetry*, i.e. users should not only submit questions but also feel encouraged to register as experts who may answer questions. Finally, experts need to be *available*. The interface therefore must entice them to be logged in to the system so that they can be reached by the system while they carry on with their everyday work.

A. *User Interface Design Principles*

Since we envision our expert peering system and its interface to run as a background process while the users continue with their usual work routine, the design of the interface follows a few simple design considerations, which roughly mimic some common *Web 2.0 principles* [1]:

a) The Web as a Platform: our system acts as an intelligent broker, hopefully fostering an ethic of cooperation.

b) Harnessing Collective Intelligence: users are encouraged to document their knowledge in blogs, which are then indexed by the system to identify relevant experts.

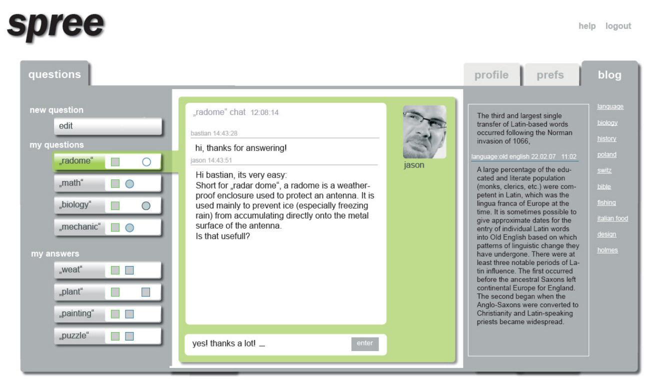
c) Extensibility and Modular Design: while our current interface implementation is tailored towards PC-based information workers, our architecture also supports inclusion of (VoIP) phones, mobile phones, text messaging, etc.

d) Rich User Experiences: our implementation uses Ajax technologies in order to deliver a full scale interactive application that can be run in modern web browsers.

B. *Functionalities and Implementation*

The current installation is to be used by a group of peers within a company whom we expect to be computer-literate and comfortable with internet use. We are currently evaluating three different interfaces which provide the same functionality but differ in the layout and interaction facilities of the users' personal start pages. The functionalities are:

- personal start page: allows for submitting queries or answering or declining queries delegated to the user; users can choose among the following appearance themes:
  1) *web search interface*: queries are entered in a standard text box and query assignments are presented as individual tabs; the user has to choose manually if (s)he wants to search in blogs or get connected to an expert (see Fig. 6 for a mockup design).
  2) *chat client*: incoming and outgoing queries are presented as several tabs of a chat window and queries are entered in an empty chat window; the system automatically connects the user to an expert or, if more appropriate, displays a relevant blog.
  3) *alternative interface*: we are also evaluating less common search interface metaphors, which, for example, are inspired by the controls of current, portable MP3 music players, which assist in seamless browsing for a particular style of music.
- set preferences: choices and settings for alerts about incoming queries
- set expert profile: either by uploading papers, URLs, or resumes for indexing (adding or replacing existing information) or by inspecting and manually editing the competence tree, i.e. the personal ontology subtree
- answer search: either by typing into a chat window or by dragging a link from a blog entry into the chat window
- rate experts: after direct interaction with an expert the knowledge seeking user can rate the session, giving



„radome" chat 12:08:14

bastian 14:43:28

hi, thanks for answering!

jason 14:43:51

Hi bastian, its very easy:
Short for „radar dome", a radome is a weather-proof enclosure used to protect an antenna. It is used mainly to prevent ice (especially freezing rain) from accumulating directly onto the metal surface of the antenna.
Is that usefull?

yes! thanks a lot! _

The third and largest single transfer of Latin-based words occurred following the Norman invasion of 1066,

language:old english 22.02.07 11:02

A large percentage of the educated and literate population (monks, clerics, etc.) were competent in Latin, which was the lingua franca of Europe at the time. It is sometimes possible to give approximate dates for the entry of individual Latin words into Old English based on which patterns of linguistic change they have undergone. There were at least three notable periods of Latin influence. The first occurred before the ancestral Saxons left continental Europe for England. The second began when the Anglo-Saxons were converted to Christianity and Latin-speaking priests became widespread.

Fig. 6. Artist's depiction of the User Interface.

  credits to the answering expert; the expert can store the chat to a blog where (s)he can edit it and make it available as an answer to future searches in order to avoid manually answering them again; at the end of a chat, the user can also rephrase the question if the answer was not satisfactory

C. *Evaluating the User Interface*

The different versions of the user interface and the overall system will be evaluated according to common *usability criteria*. According to our experiences with evaluating intelligent interactive systems (cf. e.g. [9–12]), we will focus on three aspects: *effectiveness* considers questions like: Does the system perform correctly? Can its performance be improved? With regard to *efficiency*, we ask: How long does a query take? How many other resources does the user need to fulfill the task (equipment, people, etc)? Finally, *satisfaction* concerns aspects such as: Does the use enjoy working with the system? Is is easy to operate the system? While the first two criteria can be assessed using technical measurements only, the last criterion will be evaluated based on questionnaires handed to our test users.

Following [13], we already have started a series of preliminary, iterative tests with only a few users which allow us to correct simple usability problems as early as possible in the ongoing system design process. Once the different versions of the interface have stabilized, we will perform an extensive evaluation of the overall system in to determine how the design of the user interface influences the overall performance of the system. We expect to find interdependence between the way a query is formulated (either as a normal text-based web-search as proposed in [14] or as a query in a human-to-human chat) and the result of the matching algorithm.

IV. NETWORKING

Our system presents users with a rich and engaging interface over the web browser using state-of-the-art web technologies such as Ajax [2] and dynamic web pages. These technologies use intelligent pre fetching of data from the server in order to make the network latency transparent to the user by preemptive caching of data that is likely to be requested in the future. Moreover, the web browsing screen is continuously updated in place, i.e. pages are not redrawn completely when some parts of the page change, but instead the document object model (DOM) [15] of the page is updated using dynamic HTML.

A major contribution of our work is in optimizing how much communication occurs between the web-browser and the server when *active pre fetching* is used. While active pre fetching reduces perceived network latency, it comes at the cost of larger bandwidth usage (and wastage when the pre fetched data is not accessed in the future). The server also services more HTML "GET" requests as web browsers preemptively request more information. In addition, pre fetching makes data transfer much more bursty [16] and this adversely affects router queues when the underlying network is congested. Hence there is scope for optimizing the amount and content of pre fetched data to mitigate the issues cited above.

We employ sophisticated optimization techniques to intelligently pre fetch data. In the case when multiple data objects are pre fetched on a web page, our algorithms adapt to user behavior and make intelligent choices about scheduling pre fetching requests based on which data object is being serviced at what rate in the browser. The algorithm quickly adapts to user behavior, thus reducing the number of erroneous pre fetched data objects, and improving performance significantly.

While our algorithms are obviously beneficial to the expert peering system, we have also successfully tested our optimization approaches for pre fetching optimization in several other commercially deployed web sites on the Internet. Our results indicate that our approaches improve pre fetching throughput (cache hits on preemptive content), scalability (server and network friendly), and usability (smoother end-user experience) on Web 2.0 web sites. We have also setup a test bed to capture real web traffic out of a major ISP router and use this real data to analyze and improve the pre fetching algorithms.

V. Summary and Outlook

In this paper, we presented the first prototype of a system for automatically peering knowledge seekers with knowledgable experts. It realizes an intelligent, real-time search engine for web clients that can effortlessly be deployed in enterprize Intranets or general interest online communities. Our algorithm for matching queries and appropriate experts is based on an ontology tree that covers various areas of knowledge. Mapping expert profiles and user queries to subtrees of the ontology allows for addressing the matching problem by means of dot products in a semantics-induced vector space that results from tree serialization. Extensive experimentation with an exemplary ontology demonstrated the efficiency and usefulness of our approach.

Apart from machine intelligence, our system concept also addresses interface design and usability in order to warrant Web 2.0 compliancy. We therefore also sketched how an ongoing evaluation of several usability aspects of our system guides the implementation process. Also, we briefly summarized an approach to optimizing client-server communication within our system. It makes use of intelligent pre fetching in order to alleviate network latencies and therefore further increases user comfort.

Apart from realizing and evaluating different interface implementations, we are currently technically tweaking our prototype. In addition to re-implementing parts of our system in C, this includes implementing similarity measures for bags of words that make use of efficient bloom filters. Also, we are researching extensions of our matching algorithm for query-expert peering. Here, we currently focus on investigating the use of different weighting schemes in the computation of matching scores. Future work will consider applications of dynamic ontology approaches that would allow our system to adopt to frequent usage patterns or changes in the user constituency.

References

[1] J. Musser and T. O'Reilly, "Web 2.0 – Principles and Best Practices," O'Reilly Radar, Tech. Rep., Nov. 2006.

[2] N. Zakas, J. McPeak, and J. Fawcett, *Professional Ajax*. Wiley, 2006.

[3] M. Ramm, K. Dangoor, and G. Sayfan, *Rapid Web Applications with TurboGears*. Prentice Hall, 2006.

[4] G. van Rossum, *An Introduction to Python*. Network Theory Ltd., 2003.

[5] T. Alpcan, C. Bauckhage, and S. Agarwal, "An Efficient Ontology-Based Expert Peering System," in *Proc. IAPR Workshop on Graph-based Representations*, 2007, to appear.

[6] G. Salton, *Introduction to Modern Information Retrieval*. McGraw-Hill, 1983.

[7] S. Bird, E. Klein, and E. Loper. (2007) The Natural Language Toolkit. [Online]. Available: http://nltk.sourceforge.net

[8] M. Porter, "An Algorithm for Suffix Stripping," *Program*, vol. 14, no. 3, pp. 130–137, 1980.

[9] C. Bauckhage, J. Fritsch, K. Rohlfing, S. Wachsmuth, and G. Sagerer, "Evaluating Integrated Speech- and Image Understanding," in *Proc. Int. Conf. on Multimodal Interfaces*, 2002, pp. 9–14.

[10] T. Käster, M. Pfeiffer, C. Bauckhage, and G. Sagerer, "Combining Speech and Haptics for Intuitive and Efficient Navigation through Image Databases," in *Proc. Int. Conf. on Multimodal Interfaces*, 2003, pp. 180–187.

[11] G. Heidemann, I. Bax, H. Bekel, C. Bauckhage, S. Wachsmuth, G. Fink, A. Pinz, H. Ritter, and G. Sagerer, "Multimodal Interaction in an Augmented Reality Scenario," in *Proc. Int. Conf. on Multimodal Interfaces*, 2004, pp. 52–60.

[12] M. Hanheide, C. Bauckhage, and G. Sagerer, "Combining Environmental Cues & Head Gestures to Interact with Wearable Devices," in *Proc. Int. Conf. on Multimodal Interfaces*, 2005, pp. 25–31.

[13] J. Nielsen and T. Landauer, "A Mathematical Model of the Finding of Usability Problems," in *Proc. InterCHI*, vol. 5. ACM, 1993, pp. 206–213.

[14] J. Nielsen. (2005) Mental Models For Search Are Getting Firmer. [Online]. Available: http://www.useit.com/alertbox/20050509.html

[15] W3C-Consortium. Document Object Model. 2006. [Online]. Available: http://www.w3.org/DOM/

[16] M. Crovella and P. Barford, "The Network Effects of Prefetching," in *Proc. IEEE INFOCOM*, 1998, pp. 1232–1239.