CARSA - An Architecture for the Development of Context Adaptive Retrieval Systems

Korinna Bade, Ernesto W. De Luca, Andreas Nürnberger and Sebastian Stober

Information Retrieval Group
Institute for Knowledge and Language Engineering
Otto-von-Guericke-University of Magdeburg
Universitätsplatz 2, D-39106 Magdeburg, Germany
{bade,deluca,nuernb,stober}@iws.cs.uni-magdeburg.de

Abstract. Searching the Web and other local resources has become an every day task for almost everybody. However, the currently available tools for searching still provide only very limited support with respect to categorization and visualization of search results as well as personalization. In this paper, we present a system for searching that can be used by an end user and also by researchers in order to develop and evaluate a variety of methods to support a user in searching. The CARSA system provides a very flexible architecture based on web services and XML. This includes the use of different search engines, categorization methods, visualization techniques, and user interfaces. The user has complete control about the features used. This system therefore provides a platform for evaluating the usefulness of different retrieval support methods and their combination.

1 Motivation

Searching the Web has become an every day task for almost everybody, at work as well as at home. Nevertheless, it is still difficult to find specific knowledge and requires some expertise and intuition in query formulation. One of our main objectives is to make this process a little easier.

Currently, multiple algorithms and systems are being developed for this purpose from many researchers. With CARSA (Context Adaptive Retrieval System Architecture), we want to provide a flexible architecture that allows for experimenting with different methods and their combination in a real world retrieval setting. Various methods for searching, classification, clustering, ranking and visualization can be easily integrated and combined to examine their effects on improving web and local search. We focus especially on the integration of methods that support the adaptation of the system interface and the output to the current search context. This includes methods that consider the *semantic search context* based on search results and user interests, but also the *physical search context* defined, e.g., by the device and its interaction components a user is using for searching.

2 Related Work

One way to support the user when browsing the Web are browsing assistant systems, which suggest links of interest, while the user is browsing. One such system is WebWatcher [26], which suggests a "tour" when browsing a collection of documents by highlighting links. Letizia [14] attempts to anticipate items of interest by exploring links from the user's current position. This is inferred from browsing behavior. In [6], an automatic approach for rating how interesting a web page is for a specific user is proposed.

When handling a search request, search performance can be improved, if more knowledge about the context of the search is available. Knowledge of the context can either be gained by explicit user input, or learned automatically. The Watson System [25] tries to infer context information from what the user is currently doing on his PC, e.g. documents he is typing or reading. The Persona System [18] uses explicit feedback and an underlying taxonomy to learn the user interests. This information is then used to re-rank search results. In [19], an approach for mapping user interests to categories is proposed. When a user sends a query, the system tries to map it to the right category based on the terms used. This information provides means to disambiguate the query since the category can be understood as the users intention.

Presenting the results structured in categories can also help the user to find information he is seeking faster [17]. This is done, e.g., by the Vivisimo search engine [24], which clusters the results and assigns labels to the found clusters. The clusters are presented as a folder structure. AISEARCH [1, 2] also builds categories over search results and then displays it in a graph. Ontologies provide another way to group similar documents together, see, e.g., [30].

Bookmarks of a user give clues about his topics of interest [10]. In [5], using bookmarks in collaborative filtering is studied. In [15], it is tried to learn themes in a community of surfers with overlapping interests by using the bookmarks.

One main problem of almost all approaches mentioned above is that they had been implemented in quite diverse systems and thus an evaluation or comparison of the diverse techniques with respect to usability and retrieval performance in an interactive setting is hardly possible. Especially the effects of the combination of different retrieval support methods cannot be studied due to the systems incompatibilities. A similar integration idea (however in another field of application) is underlying the CANDELA system for video retrieval [4] and the system for media management described in [12].

3 System Architecture

It was our main goal to develop a retrieval system, which is as flexible as possible, to provide a platform for combining different methods for all parts of the multimedia retrieval process. So far, we mainly consider the retrieval of text documents. However, the system is designed such that multimedia data could also be handled. An overview of the system architecture is shown in Figure 1.

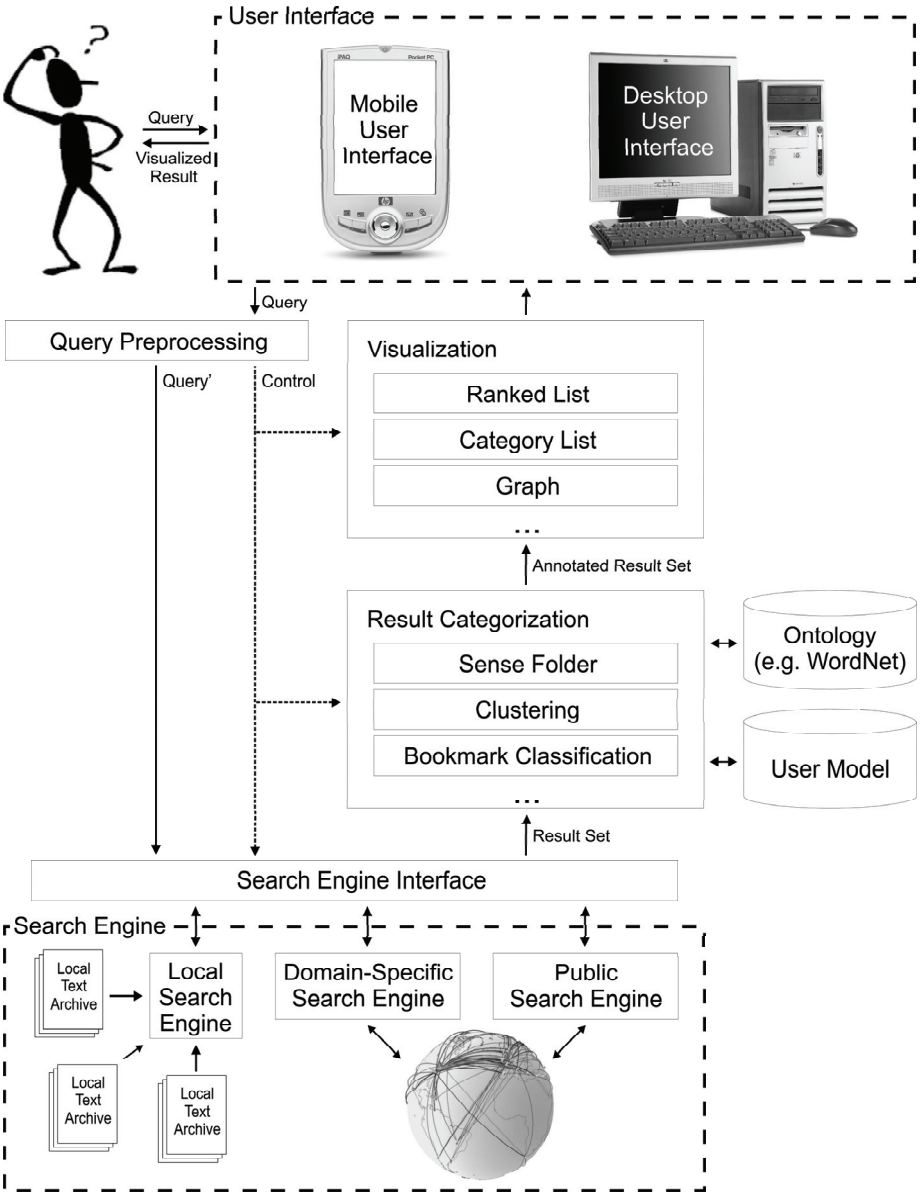

Fig. 1. System Architecture

The system should be accessible from different computers as one user usually accesses the web through different computers, e.g. at home, at the office, or on mobile devices. Therefore, the main application runs on a server. User access can be gained either through a web browser or through a locally installed application (see Sect. 3.1 for details).

The main application handles a pool of plug-ins, each containing one specific method for a specific part of the search process. This comprises plug-ins for accessing document collections for indexing and searching (e.g. local archives, benchmark collections or direct access to search engines like Google), plug-ins that provide methods to structure (e.g. clustering and classification algorithms) or to mine (e.g. methods to extract descriptive concepts) result sets as well as methods that preprocess the data such that it can be easily visualized by a client tool connected to the meta-searcher (e.g. creating graph structures or computing an overview of the result set using self-organizing maps [11]). Furthermore, plug-ins for user profiling (e.g. to access profiles obtained by the proxy based logging mechanism of the CARSA system) and plug-ins that provide access to ontologies (e.g. WordNet) that are required by other plug-ins (e.g. for multilingual support or semantic categorization) can be integrated. Each plug-in can run either on the same server as a dynamically linked java class or on another server accessible through a web service. The first method increases computational efficiency. However, it might not always be possible to run everything on the same server, especially when external resources like the Google API are used. Plug-ins can be registered by a web service or locally with a configuration file.

The plug-ins that should be used for a specific search session can be dynamically selected, i.e., when sending a search request, the user interface can also send information about what plug-ins to use. Otherwise a server defined default setting is used. The search query is forwarded to the selected search engine via the search engine interface (see Sect. 3.2). It returns a ranked list of results. This list, possibly together with an ontology or user model, is used to categorize the results (see Sect. 3.3). After this, the altered list is processed by a visualization module (see Sect. 3.4), which presents the results to the user. Several categorization methods can be called in parallel and will be visualized separately.

All data transfer is done using XML structured data packages. Therefore, the information exchange between different modules can be easily revised or extended. An example of such an XML string, describing the annotated results of a search query, is shown in Fig. 2.

3.1 User Interface

The CARSA architecture enables the development of user interfaces for different contexts (e.g. problem and device specific). These interfaces not only allow the user to specify a search query for a selected search engine, but they can additionally provide the possibility to select and configure the plug-ins registered at the system to suit the user's current needs. Predefined user- and system-related presets help to ease the configuration. These presets are stored on the central server

```
<?xml version="1.0" encoding="UTF-8"?>
<Result DocumentFiltering="false" EndIndex="10" EstimateIsExact="false"
 EstimatedTotalResultsCount="3870000" SearchComments=""
 SearchTime="0.031655" SearchTips="" StartIndex="0">
 <Query QueryString="universitaet"
  SessionID="7F881092EC34CBA2F421C955A2EC3BDD" UserSessionID="357021445">
 ...
 </Query>
 <RE CachedSize="19k" DirectoryTitle="Freie <b>Universitaet</b> Berlin"
  HostName="" RelatedInformationPresent="true" Snippet="Hauptsite.
  Aktuelle Meldungen, Informationen ueber Studium, Forschung, Links zu<br>
  den Einrichtungen der <b>Universitaet</b>, Online-Vorlesungsverzeichnis
  sowie <b>...</b>" Summary="Hauptsite. Aktuelle Meldungen, Informationen
  ueber Studium, Forschung, Links zu<br>  den Einrichtungen der
  <b>Universitaet</b>, Online-Vorlesungsverzeichnis sowie Hinweise zu
  <b>...</b>" Title="Freie <b>Universitaet</b> Berlin"
  URL="http://www.fu-berlin.de/">
  <PluginResultList>
   <PluginResultElement
    JavaClassName="de.unimd.irgroup.carsa.classification.SetClassification"
    Plugin="BayesClassifier">
    <CE ClassName="Uni" Probability="1.0"/>
    <CE ClassName="Berlin" Probability="0.9999999999984375"/>
   </PluginResultElement>
   <PluginResultElement
    JavaClassName="de.unimd.irgroup.carsa.classification.SetClassification"
    Plugin="kNN">
    <CE ClassName="Berlin" Probability="0.84"/>
    <CE ClassName="Magdeburg" Probability="0.16"/>
   </PluginResultElement>
  </PluginResultList>
 </RE>
 ...
</Result>
```


Fig. 2. Sample XML-string describing annotated results

to provide maximum accessibility. In order to give more insight into the ideas and capabilities of CARSA, we describe in the following briefly some interfaces that have been already implemented using this architecture.

First, there is a java servlet based web interface (Fig. 3) that requires no special software to be installed on the client's side. The user can simply access the system with a web browser of his choice and use it like any "ordinary" internet search engine. A login provided at the web site enables access to a stored user profile. Furthermore, the interface provides transparent access to select and configure the installed plug-ins (see Fig. 4). This user interface can be used as both, mobile and desktop user interface.

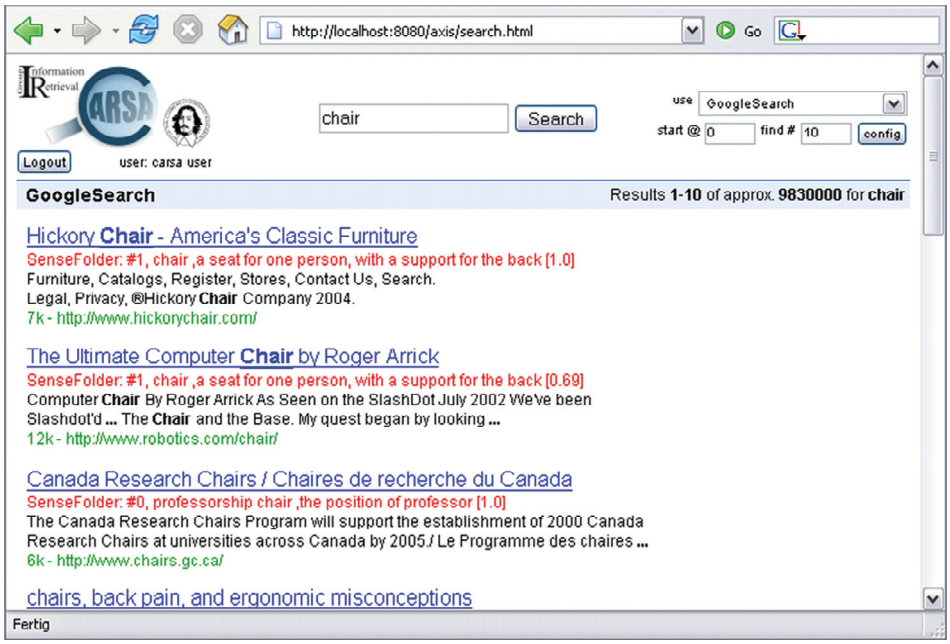

Fig. 3. Web Based Interface: Searching

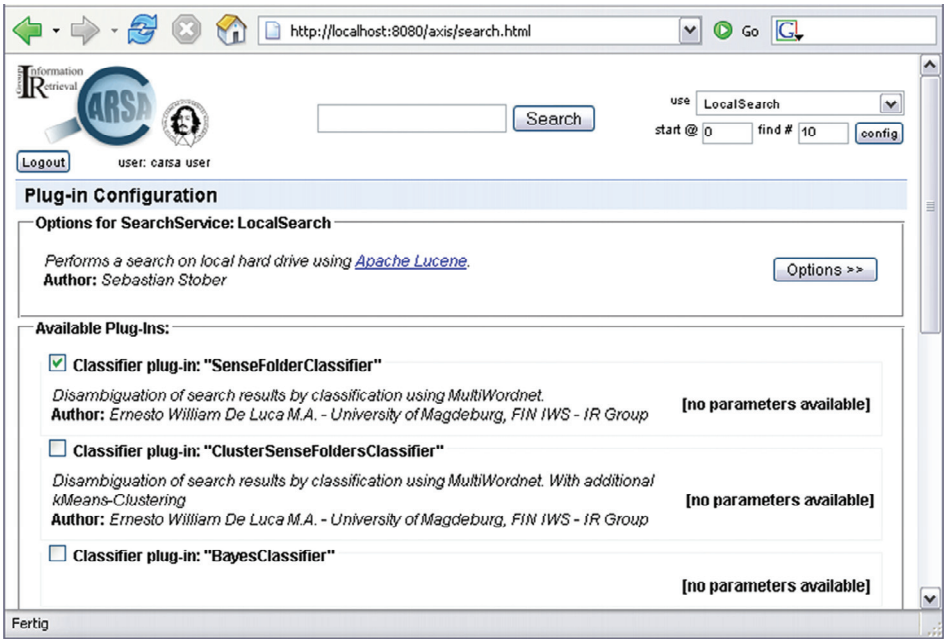

Fig. 4. Web Based Interface: Configuration Dialog

Furthermore, a client based interface that contains means to manage bookmarks stored on the server as well as providing access to our search plug-ins (Fig. 5) was implemented. Results of a search request are shown in the web browser using the java servlet (Fig. 3) described above. However, based on the bookmark management, this tool enables the use of bookmark based classification methods (see also Sect. 3.3).

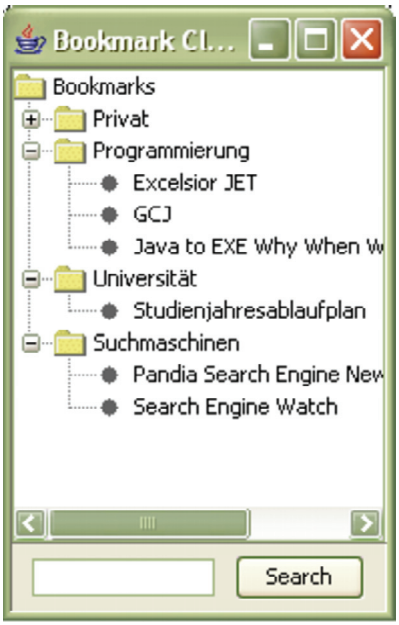

Fig. 5. Bookmark Client

The mobile user interface shown in Figures 6 – 8 has been especially designed to meet the requirements that go along with the limited display size of mobile devices such as PDAs and the specific type of interaction using a pen [20]. It has been implemented in Macromedia Flash. Hence, in addition to the application, the Macromedia Flash Player has to be installed on the user's mobile device. The user interface is divided into three views. In the first view (Fig. 6), the user can specify his query and select a page of search results that then is displayed in the second view (Fig. 7). Results may be added to bookmark categories that are displayed in the third view (Fig. 8).

There is also a desktop version of the mobile user interface available that puts the three views of the mobile user interface together into a single frameset. It also requires installation of a Macromedia Flash Player.

All user interfaces presented here still provide a ranked list of search results. Further information gained through one or several categorization methods are simply added as additional information to the document snippets. However,

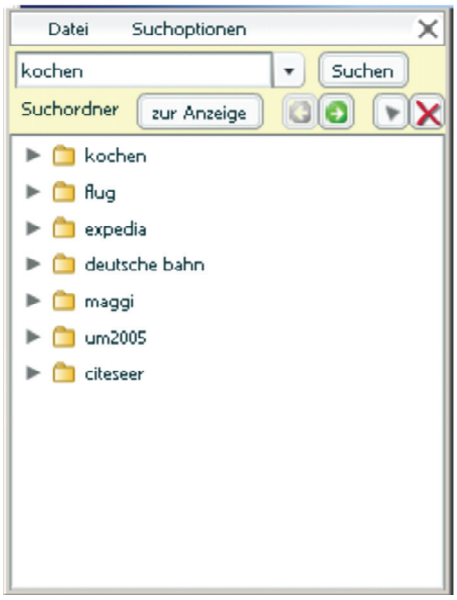

Fig. 6. PDA Client - Search

Fig. 7. PDA Client - Results

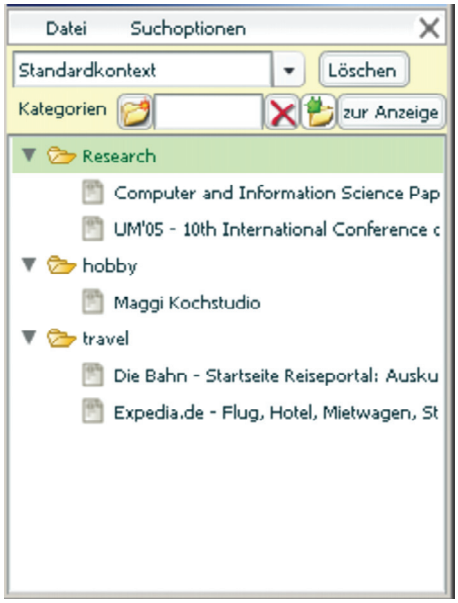

Fig. 8. PDA Client - Bookmarks

we are currently working on other visualization techniques, e.g. for visualizing classification of search results into an existing class structure.

3.2 Search Engine Interface

The search engine interface allows searching for data over different search engines. Each possible search engine connection is handled by plug-ins. Each plug-in modifies the input to fit the specific search engine and ensures that the output of the search engine is converted to the CARSA specific internal format.

Furthermore, for web based searching the CARSA system automatically downloads, indexes, and caches the web pages of the search hits and provides these information to the plug-ins for further processing, e.g., for classification or re-ranking. Due to the multithreaded architecture and the cache this approach is reasonably efficient also for online searching.

The following search plug-ins have been implemented so far:

- To search the whole web, we currently use the Google API [8].
- Furthermore, we have implemented our own webcrawler, which runs on a cluster PC. It is designed to crawl a domain-specific subset of the web. The domain is described by linguistic quantifiers.
- We also have developed a local searcher working on a stored document collection. On the one hand, this enables testing on a defined data set (benchmark collection). On the other hand, the local search can be used to search data distributed on several computers in the Intranet.

3.3 Plug-ins for Categorization, Clustering and Re-ranking

In order to allow a simple post-processing of result sets, plug-ins for categorization, clustering and re-ranking can be easily installed. These plug-ins have full access to user specific information (e.g., bookmarks and user profiles) and to further information resources like ontologies. The plug-ins receive a fully indexed set of the retrieved documents and can compute annotations (e.g. class or cluster information) or re-rank the set. In the following, we briefly describe some annotation methods that have already been implemented.

Categorizing search results can be done in several ways. Without using any additional information, one can find clusters of similar documents based on their $tf \times idf$-vectors [3]. The found clusters can then be labeled, e.g. by the method proposed in [13].

Another approach studied by us is based on the use of semantic information provided by an ontology. As language itself is highly ambiguous so are the search terms. However, a user has usually only one meaning in mind. Therefore, categorizing the search results in groups of meaning (the so called *Sense Folders*) can help extracting the possibly interesting documents. The different meanings of a word are stored in an ontology. In our case, we use the WordNet ontology [29, 28], which is a general ontology. More details about the sense folder approach can be found in [16, 9].

A third approach studied by us is based on a user profile, which is constantly built while the user is browsing the web. Part of this profile are the bookmarks. If structured, they provide some hints about topics of interest of the user and how he distinguishes one from another. We currently use an approach based on Naïve Bayes classifiers to assign the bookmark categories provided by a user to the entries in a result set [21].

3.4 Visualization

The easiest way of visualizing the results is a ranked list of the result documents. Information gained from categorization can be displayed in an additional row for every document stating the assigned class. However, this is just slightly helpful as the user still has to scan all results.

Displaying a category tree with the documents on the lowest level can speed up the search process [17], because the user only needs to look at selected results as the category tree gives some clues about the content of the specific web sites.

Bringing this category tree a step further is to display them in form of a graph. Here, more information can be visualized in a single fingure, e.g. similarity of topics can be shown by mapping similar topics closer together than distinct topics (see tools like SPIRE [23], SCI-Map [22], VxInsight [7] and mappings using self organizing maps [11, 27]).

4 Conclusion

In this paper, we have given a brief overview of the CARSA system, an architecture that supports the development and evaluation of context adaptive information retrieval systems. We have motivated and described the underlying architecture that allows the integration of diverse methods for user support in information retrieval tasks in a unified framework. In order to show its usability in practice, we briefly described some interfaces and annotation methods that we have already implemented based on this architecture. However, more detailed user studies and large-scale performance experiments on the classifiers still have to be done.

Further information about CARSA is available on the web pages of the information retrieval group Magdeburg: `http://irgroup.cs.uni-magdeburg.de/`.

References

1. B. Stein, S. Meyer zu Eissen: AISEARCH: Category Formation of Web Search Results, 2003.
2. AISEARCH, http://www-ai.upb.de/aisearch/.
3. G. Salton, A. Wong, C.S. Yang: A vector space model for automatic indexing. In: Communications of the ACM 18(11), 1971.

4. E. Jaspers, R. Wijnhoven, R. Albers, J. Nesvadba, J. Lukkien, A. Sinitsyn, X. Desurmont, P. Pietarila, J. Palo, and R. Truyen: CANDELA - Storage, Analysis and Retrieval of Video Content in Distributed Systems In: Proc. of the 3rd International Workshop on Adaptive Multimedia Retrieval (AMR), 2005.
5. J.J. Jung, J.-S. Yoon, G. Jo: Collaborative Information Filtering by Using Categorized Bookmarks on the Web, In: 14th International Conference of Applications of Prolog (INAP), 2001.
6. P.K. Chan: Constructing Web User Profiles: A Non-invasive Learning Approach, In: Proc. of the International Workshop on Web Usage Analysis and User Profiling, 1999.
7. K. W. Boyack, B. N. Wylie, G. S. Davidson: Domain Visualization Using VxInsight for Science and Technology Management, Journal of the American Society for Information Science and Technologie 53(9):764-774, 2002.
8. Google API, http://www.google.com/apis/.
9. E.W. De Luca, A. Nürnberger: Improving Ontology-Based Sense Folder Classification of Document Collections with Clustering Methods, In: Proc. of the 2nd Intern. Workshop on Adaptive Multimedia Retrieval (AMR), 2004.
10. D. Abrams, R. Baecker, M. Chignell: Information Archiving with Bookmarks: Personal Web Space Construction and Organization, In: Proc. of the SIGCHI conference on Human factors in computing systems, 1998.
11. A. Nürnberger: Interactive Text Retrieval Supported by Growing Self-Organizing Maps, In: Proc. of the International Workshop on Information Retrieval, 2001.
12. W. Fontijn, J. Nesvadba, A. Sinitsyn: Integrating Media Management Towards Ambient Intelligence; In: Proc. of the 3rd International Workshop on Adaptive Multimedia Retrieval (AMR), 2005.
13. K. Lagus, S. Kaski: Keyword Selection Method for Characterizing Text Document Maps, In: Proc. of the ninth international conference on artificial neural networks (ICANN'99), Vol. 1, 1999.
14. H. Lieberman: Letizia: An Agent That Assists Web Browsing, In: Proc. of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI-95), 1995.
15. S. Chakrabarti, Y. Batterrywala: Mining themes from bookmarks, In: Proc. of the Workshop on Text Mining held at the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), 2000.
16. E.W. De Luca, A. Nürnberger: Ontology-Based Semantic Online Classification of Documents: Supporting Users in Searching the Web, In: Proc. of the European Symposium on Intelligent Technologies (EUNITE), 2004.
17. S. Dumais, E. Cutrell, H. Chen: Optimizing search by showing results in context, In: Proc. of the SIGCHI conference on Human factors in computing systems, 2001.
18. F. Tanudjaja, L. Mui: Persona: A Contextualized and Personalized Web Search, In: Proc. of the 35th Hawaii International Conference on System Sciences, 2002.
19. F. Lui, C. Yu, W. Meng: Personalized Web Search by Mapping User Queries to Categories, In: Proc. of the 11th International Conference on Information and Knowledge Management, 2002.
20. E.W. De Luca, A. Nürnberger: Supporting Information Retrieval on Mobile Devices In: Proc. of the 7th International Conference on Human Computer Interaction with Mobile Devices and Services, 2005.
21. K. Bade, A. Nürnberger: Supporting Web Search by User Specific Document Categorization: Intelligent Bookmarks In: Proc. of the Leipziger Informatik Tage (LIT), 2005.

22. H. Small: Visualizing science by citation mapping, Journal of the American Society for Information Science 50(9):799-813, 1999.
23. J. A. Wise, J. J. Thomas, K. Pennock, D. Lantrip, M. Pottier, A. Schur and V. Crow: Visualizing the non-visual: Spatial analysis and interaction with information from text documents, In: Proc. of IEEE Symposium on Information Visualization, 1995.
24. Vivisimo, http://vivisimo.com.
25. J. Budzik, K. Hammond: Watson: Anticipating and Contextualizing Information Needs, In: 62nd Annual Meeting of the American Society for Information Science, 1999.
26. T. Joachims, D. Freitag, T. Mitchell: WebWatcher: A Tour Guide for the World Wide Web, In: Proc. of the 15th International Joint Conference on Artificial Intelligence, 1997.
27. A. Nürnberger and M. Detyniecki: Weighted Self-Organizing Maps: Incorporating User Feedback, In: Proc. of 13th Int. Conf. on Artificial Neural Networks and Neural Information Processing (ICANN/ICONIP 2003), pp. 883-890, Springer-Verlag, 2003.
28. WordNet homepage, http://wordnet.princeton.edu/.
29. C. Fellbaum: WordNet, an electronical lexical database, 1998.
30. Y. Labrou, T. Finin: Yahoo! as an ontology: using Yahoo! categories to describe documents, In: Proc. of the 8th Int. Conf. on Information and Knowledge Management, 1999.