MuZeeker:
a domain specific Wikipedia-based search engine

Søren Halling, Magnús Sigurðsson, Jakob Eg Larsen,
Søren Knudsen, and Lars Kai Hansen

Technical University of Denmark
Department of Informatics and Mathmatical Modeling
Richard Petersens Plads, Building 321
DK-2800 Kgs. Lyngby, Denmark.
{sch, mks, sk}@muzeeker.com, {jel, lkh}@imm.dtu.dk

Abstract. We describe MuZeeker, a search engine with domain knowledge based on Wikipedia. MuZeeker enables the user to refine a search in multiple steps by means of category selection. In the present version we focus on multimedia search related to music and we present two prototype search applications (web-based and mobile). A category based filtering approach enables the user to refine a search through relevance feedback by category selection instead of typing additional text, which was found to be an advantage in the mobile MuZeeker application. We report from a usability evaluation using the think aloud protocol, in which N=10 participants performed tasks using respectively MuZeeker and a customized Google search engine. The experiment gave initial indications that participants were capable of solving tasks slightly better using MuZeeker, while the "inexperienced" MuZeeker users performed slightly slower than experienced Google users. 75% of the participants reported a subjective preference for MuZeeker.

Keywords: search engine, music, user interfaces, usability, mobile.

1 Introduction

The rapid growth and massive quantities of multimedia data available to the Internet user underlines the importance and complexity of information retrieval. The amount and diversity of data and distributed database structures challenges the way search engines rank their results and often several million results are presented to a single query. Furthermore, the constraints on mobile devices in terms of limited display size and resolution along with limited text entry facilities makes information search and retrieval an even bigger challenge. In order to improve the user experience when using search engines on mobile devices, the task is to optimize the utilization of screen real-estate and limit the amount of explicit text input required by the end-user.

We introduce the Zeeker Search Framework [1] which uses Wikipedia [2] to generate categorization of the individual search results, thus enabling search refining by a category selection instead of typing a new query. The information architecture of

Wikipedia forms a very useful knowledge base for machine learning and natural language processing thus helping the Zeeker Search Framework's contextual understanding and ability to categorize search results. A better understanding of the semantics of user queries and web content by categorization facilitates presentation in mobile devices and usability. We hypothesize that Wikipedia can offer the necessary categorization in many domains including the domain of interest here, namely music.

2 Search engine overview

The Zeeker Search Framework [1] (ZSF) is aimed to be flexible, and to offer "real-time" specialization through Wikipedia. In many domains the broad group of contributors warrants swift updates on reaction to news and domain events. The framework is developed primarily with search result categorization, topic specialization and result presentation in mind. The ZSF structure is divided into three separate layers as shown in Fig. 1, namely a data layer, an xml web service and a presentation layer. The layers enhance the search engine's flexibility, making it easier to improve and expand across platforms and technologies.

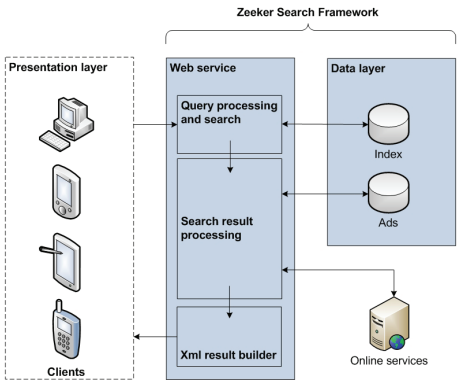

Fig. 1. Overview of the Zeeker three-tier system architecture: presentation, web service and data layer. A standard web service interface links the presentation and framework.

The data layer contains the index, categories, associated adverts and any additional meta-information such as query suggestions, document types etc. Customized indexes can be created using any subset of articles from the Wikipedia knowledge base. This is achieved by indexing articles related only to a specific category along with its sub-categories and articles e.g. Music, Birds, Films, Wine, History, etc.

Besides deep categorization and relatively high trust article contexts [3], Wikipedia has a number of additional useful attributes. All articles have a quality measure, such as "stub", "normal" and "featured" that indicates how Wikipedia rates the context of the article. Every article also has associated discussion pages and logs of what has changed, who changed it and when. These article attributes provide information, e.g., about the article quality. Long discussion pages, many major and minor edits, the profile of the user doing the editing, amount of links to other Wikipedia articles,

amount of external links, how many images are included and which images, templates used etc. can all be used as quality indicators for a specific article. These measures can even be used as filters to remove bad articles when trying to generate accurate contextual information.

During the indexing process, part of the additional contextual information extracted from Wikipedia, is stored along with a traditional full inverted index [4] of the articles content. The contextual information acquired during indexing enables the search engine to relate individual search results with information from external multimedia resources such as YouTube videos. The XML web service layer provides the search and information retrieval service. Using the knowledge gained from Wikipedia, the ZSF is able to classify a given user submitted query to a set of ranked categories along with a set of ranked search results. In contrast to other datamining and clustering based search engines the Wikipedia categories are user generated and constantly under review. The returned categories can then be used in subsequent queries to narrow the search domain. We consider the filtering capability of ZSF one of the main advantages relative to conventional search engines. The web service returns search results, categories, adverts and contextual information retrieved from external resources in XML format, facilitating presentation of the results in a desired form. Currently the web service layer supports searches based on the latin or cyrillic character set and can easily be extended to support other character sets as well.

As mentioned above, many subsets of Wikipedia can be used to create an index and thereby creating domain specific search engines. As a proof-of-concept, we chose to build a music search index with web- and mobile-based presentation layers ontop of the ZSF called MuZeeker. MuZeeker uses the contextual information from the search results to relate individual search results to YouTube videos and could be extended to relate to resources such as Last.fm (Audioscrobbler), lyrics databases, etc.

The encyclopedic knowledge available in Wikipedia has been used in other contexts besides ZSF such as [5] where Wikipedia data is used for semantic disambiguation and recognition of named entities. Semantic information based on Wikipedia has also been used in search engines such as Powerset [6] and Koru [7]. Powerset uses its semantic understanding to find answers to user's questions within the articles while Koru uses its semantic understanding to expand queries and guide the user interactively to the best results. Other search engines using Wikipedia data include Wikiwix [8] and Exalead [9]. These two engines associate categories with each search result but do not provide category searches at the same level as ZSF.

3 User interfaces

Two prototype user interfaces (the presentation layer) have been developed for the search engine. The user interface for MuZeeker is designed with an appearance similar to web search engines such as Google. The user can key in the query terms in an input field, which can also contain a set of parameters to refine the query. The syntax of the query is similar to that of typical web search engines, that is, it includes a number of search terms and supports a basic set of operators: AND, OR, NOT, EXACT (match), and ORDER. These operators can be used along with other

operators, thus giving the user a flexible syntax to build the queries needed to find the most relevant information. A basic search query consists of a number of search terms, which would match a text where all the search terms are available, as by default, the search engine uses a boolean AND query of all the query terms. Category filters can be added using a "-cat" parameter to the search query and several categories can be included.

The primary difference between MuZeeker and conventional search engines is, that MuZeeker returns two sets of results for any search query. One is a set of articles matching the search query, and the other is a set of matching Wikipedia *categories*. The set of matching categories enables the user to refine his/her search, a form of relevance feedback. The user can either add categories by including the "-cat" parameter mentioned above, or by selecting one of the matching categories returned by the search engine. Selecting a category is equivalent to manually adding the category to the search query and submitting it. Users repeat this process until a satisfying result has been obtained. A result will include a snippet from the original Wikipedia article (the first couple of lines), a link to the original Wikipedia article and may include other resources as well.

The web-based user interface for MuZeeker is designed similarly to conventional search engines, while allowing for the additional features present. For current design see [10]. A ranked list of search results is linked to additional information in the web interface. In addition to the Wikipedia search results, additional information about each search result is included when available. This information is primarily the document type of the search results, e.g. musical artist, television series, mountain, software etc. When the information is available, a link to a video search on YouTube is provided. In MuZeeker the links could be to a music video related to an article describing a particular song.

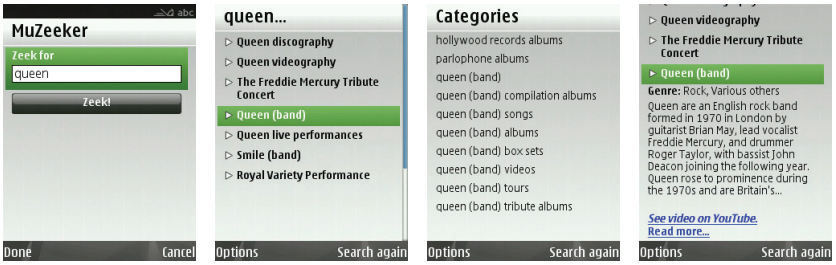

Fig. 2. Four screenshots from the MuZeeker mobile application prototype: a) main search interface, b) ranked list of search results, c) ranked list of categories and d) expanded result

The user interface for the mobile MuZeeker search application is adapted for the S60 mobile platform, see Fig. 2. The mobile application is implemented for Nokia N95 and N95 8GB mobile phones using the Web-Runtime (WRT) environment [11]. Due to the limitations of the platform and devices a number of optimizations of the user interface has been made. Compared to the web-based user interface discussed above some features have be left out. The core functionality included are the search facility, the search results, the category selection and the display of a small snippet of the articles. Links to the original Wikipedia article and YouTube video are included in

the present prototype. Selecting a link will launch external applications on the mobile device in order to display the content (mobile webbrowser or Real Player). The constraints of the display allows only about ten lines of text on the screen. This makes it difficult to include both the list of search results and the list of categories on the same screen and still allow easy user navigation. Therefore, the search results and categories have been divided into two screens, as illustrated in Fig. 2. By default the ranked list of search results are shown to the user, but shifting to the ranked list of categories is done by pressing one button. The number of search results and categories displayed is ten items due to the screen limitations to avoid scrolling a list, which would likely be perceived as cumbersome. The intention of the approach is that text entry should be limited to the extent possible. Thus, a typical scenario is that the user types in some search terms using the main search interface (Fig. 2.a). By using the results and categories the user can refine the search by adding a category to the search query by *selecting* the category from the list (Fig. 2.c). The advantage is that little typing (text input) is required in order to refine the search, as it can be done using the four-way scroll key. Nevertheless, the mobile application does support the full query syntax and operators as described above, but it would typically be too cumbersome to use, due to the text entry limitations on standard mobile phones.

4 User experience study

A usability evaluation of the web-based MuZeeker interface was carried out with the primary goal to compare and contrast to a well-established search engine, here chosen to be Google. This was implemented so that participants solved typical tasks using the two search engines. Testing and comparing MuZeeker with the general search engine is challenged since MuZeeker is domain specific. Therefore, a Custom Google Search Engine [12] was created to make the results comparable. The Custom Google Search Engine was designed to only search within the en.wikipedia.org domain excluding sites having the character ':' in the url and included a default search term – music – which was invisible to the test participants. This means that in the test, Google's functionality was the same as the standard Google search engine, but a set of predefined parameters were automatically added in each search. Furthermore, to assure that it was the search engines that were tested (and not Wikipedia), test participants were instructed not to use links pointing out of the search engine domain, i.e., solving the tasks using only the text snippets.

The experiment was conducted using the think aloud protocol as in [13] with N=10 participants in total, nine male and one female. The first two tests were considered pilot tests which led to protocol adjustments. The average age was 26.6 years with a standard deviation of 2.7 years. All except one had experience in designing or programming software and all had, or were currently studying at MSc or PhD level. All ten reported Google as their preferred web search engine. The tasks were designed without prior knowledge of how difficult they would be to solve in the two search engines. However, the tasks were designed to increase in difficulty. Some tasks were closed tasks such as "Who left The Supremes in 1970" while others were open and thereby yielded a broader test. The open tests have been left out of the comparisons

below, because different search objectives results in different completion times and ratios. The latter tests in each search engine were complicated tests which asked the test participants to find similarities between two persons, bands or similar. Information was gathered about participants' age, education, general use of the Internet, search engines, and music. 14 tasks were given sequentially in writing and each test participant was timed, and asked to read aloud each task. They were told that some tasks might not be solvable, which is believed to make test participants behave closer to a typical situation, in which, users are not aware whether or not a given task can be solved. In the concluding debriefing session, the test participants were asked to rate their experience of using the two different search engines. The questionnaire asked the test participants to rate each search engine in terms of the intervals: Terrible–Wonderful, Frustrating–Satisfying, Dull–Stimulating, Confusing–Clear and Rigid–Flexible, based on the Questionnaire for User Interface Satisfaction (QUIS) for subjective evaluation measures [14]. The test participants also indicated which search engine they would like to use for subsequent similar tasks and to rate the tasks in terms of the intervals: Hard – Easy, Hard to grasp – Easy to grasp and Unrealistic – Realistic.

5 Results and discussion

On average, MuZeeker performed better than the custom Google search engine (although marginally) in terms of the number of correctly solved tasks, as can be seen in Fig. 3.a.

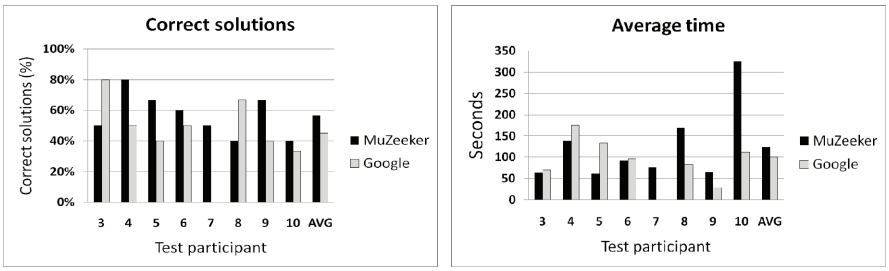

Fig. 3. a) Percentage of correct solutions for each test participant. b) Average time used for correctly solving tasks for each test participant. A time value of zero indicates no correct solutions. The open-ended tasks have been left out in both illustrations.

On average, the custom Google search engine performed better than MuZeeker (although marginally) in terms of the time spent on solving a task, as can be seen in Fig. 3.b. These results indicate that "inexperienced" MuZeeker users are slightly slower but on the other hand able to solve tasks slightly better in MuZeeker compared to the custom Google search engine. All test participants found the tasks easy to grasp/understand and they also found the tasks realistic meaning that they could see themselves doing similar tasks in search engines. However, a majority of the test participants found that the tasks were hard to solve. The overall subjective satisfaction of MuZeeker was slightly higher than Google, as can be seen in Fig. 4. A paired t-test

on the 5 subjective measures showed a significant difference between the averages ($p < 0.05$).

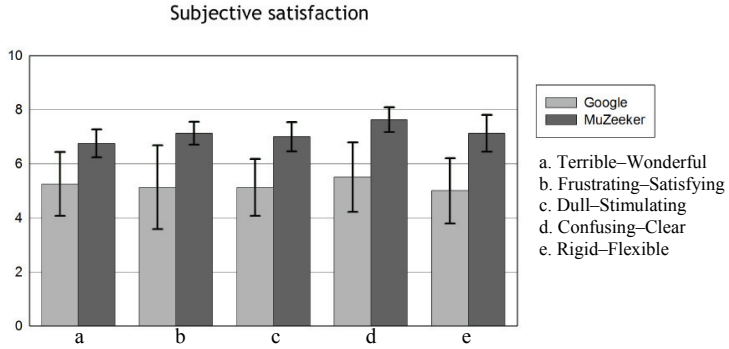

Fig. 4. Overview of the subjective satisfaction with the two alternative search engines, with an overall higher subjective satisfaction of MuZeeker compared to Google.

6 out of 8 test participants (75%) indicated that they would choose MuZeeker over Google if they had to carry out similar tasks in the future. The two test participants with preference for Google had a normal query length, but fewer than average queries, which could be a direct result of faster completion times, as the average query count and length is constructed from all tasks (solved or not solved). A summary of the results is shown in Table 1.

Table 1. Summary of the experimental results. The participants solved a total of 44 tasks in each search engine.

	MuZeeker	Google
Tasks solved correctly (average)	56.8%	45.5%
Average number of searches/category choices	20.75	19.25
Average completion time (correctly solved) (s)	124.0	99.9
Subjective preference	6	2

On inquiry it was found that none of the test participants were able to explain what the category filter did and how they did it, although many said that they liked its results. Many test participants were surprised at how difficult it was to obtain certainty of the correctness of the snippet information, as they were presented out of context. Still many, although aware of these uncertainties, mistakenly made incorrect inferences solely out of the snippet information. Some even did this, although it contradicted what they thought they knew about a band or artist.

6 Conclusions

We have described our Wikipedia domain specialized search engine MuZeeker. The user can refine a search in multiple steps by means of selecting categories. The domain supported by the MuZeeker search engine is music. We have carried out initial experiments using two search applications prototypes (web-based and mobile).

The experiments were carried out using the think aloud protocol with ten participants performing tasks using respectively MuZeeker and a custom Google search engine. Our findings from experiments present evidence that MuZeeker users were capable of solving tasks slightly better using MuZeeker, but the "inexperienced" MuZeeker users performed slightly slower than experienced Google users. However, further studies are necessary in order to arrive at solid conclusions. 75% of the participants reported a subjective preference for MuZeeker for solving similar tasks. The category-based filtering approach enables users to refine a search by performing selections rather than typing additional text. This is highly relevant in the mobile application where the amount of necessary text input is minimized and navigation support in refining a search is enhanced. The experiments also showed that the participants had problems explaining the category filters. In some cases the multiple steps made it unclear to the participants which query the shown search results related to. The initial experiments with the mobile application demonstrated the search approach as a promising way to make large quantities of information searchable on mobile devices.

Acknowledgments. This project was supported by The Danish Research Council for Technology and Production, through the framework project 'Intelligent Sound', www.intelligentsound.org (STVF No. 26-04-0092). We would like to thank Forum Nokia for the support of mobile devices. Finally we would also like to thank the ten participants in the usability experiment.

References

1. Magnus Sigurdsson and Søren C. Halling. Zeeker: A topic-based search engine. Technical University of Denmark, IMM-Thesis-2007-91 (2007)
2. Wikipedia – http://en.wikipedia.org
3. Jim Giles. Internet encyclopedias go head to head. Nature 438, 900-901, December (2005)
4. Justin Zobel and Alistair Moffat. Inverted files for text search engines. ACM Comput. Surv., 38(2):6, (2006)
5. Silviu Cucerzan. Large-Scale Named Entity Disambiguation Based on Wikipedia Data. Proceedings of EMNLP-CoNLL 2007, 708-716 (2007)
6. PowerSet – http://www.powerset.com
7. David N. Milne and Ian H. Witten and David M. Nichols. A knowledge-based search engine powered by wikipedia. CIKM '07: Proceedings of the sixteenth ACM Conference on information and knowledge management, 445-454 (2007)
8. Wikiwix – http://www.wikiwix.com
9. Exalead – http://www.exalead.com
10. MuZeeker search engine website – http://www.muzeeker.com
11. Widgets: Nokia Web Runtime (WRT) platform – http://www.forum.nokia.com/main/resources/technologies/browsing/widgets.html
12. Google Custom Search Engine – http://google.com/cse
13. Boren, M. T. and Ramey, J. Thinking Aloud: Reconciling Theory and Practice. IEEE Transactions on Professional Communication. 43, 3, 261-278 (2000)
14. Chin, J. P., Diehl, V. A., and Norman, K. L. Development of an instrument measuring user satisfaction of the human-computer interface. CHI '88: Proceedings of the SIGCHI conference on Human factors in computing systems, 213-218 (1988)