ezDL: An Interactive Search and Evaluation System

Thomas Beckers
Information Engineering
University of Duisburg-Essen
Duisburg, Germany
thomas.beckers@uni-due.de

Sebastian Dungs
Information Engineering
University of Duisburg-Essen
Duisburg, Germany
sebastian.dungs@uni-due.de

Norbert Fuhr
Information Engineering
University of Duisburg-Essen
Duisburg, Germany
norbert.fuhr@uni-due.de

Matthias Jordan
Information Engineering
University of Duisburg-Essen
Duisburg, Germany
matthias.jordan@uni-due.de

Sascha Kriewel
Information Engineering
University of Duisburg-Essen
Duisburg, Germany
sascha.kriewel@uni-due.de

ABSTRACT

The open-source system *ezDL* is presented. It is an interactive search tool, a development platform for interactive IR systems, and an evaluation system. *ezDL* can be used as a meta-search system for heterogeneous sources or digital libraries, allows organizing and filtering of merged results, offers support for search sessions as well as a personal library for storing different document types. The *ezDL* framework is easy to extend and is based on a service-oriented architecture. In addition, support for performing user studies and eye tracking is provided. *ezDL* has been used as a system in several funded research projects.

Categories and Subject Descriptors

H.3.4 [**Information Storage and Retrieval**]: Systems and Software

General Terms

Human Factors, Experimentation

Keywords

interactive search system, framework

1. INTRODUCTION

In this paper we present *ezDL*, an open-source[1] software for building highly interactive search user interfaces with strategic support. It builds on the ideas developed and implemented within the Daffodil project from 2000 to 2009 [11, 16, 13], but uses more modern software technologies and interface design methods.

The *ezDL* framework can be characterized by three main purposes. It is foremost *i*) a working interactive tool for searching a heterogeneous collection of digital libraries. In addition to that, it is *ii*) a flexible software platform providing a solid base for writing customized applications as well as *iii*) a system that can be used for many different types of user evaluations.

Today many systems covering one or more aspects of *ezDL* exist but to the best of our knowledge the concept of unifying them into one single framework is unique. In the following paragraphs similar systems related to the different aspects of *ezDL* are presented.

Interactive Search Tools

Querium [9] is an interactive search system featuring a concept that focuses on complex recall oriented searches. It aims at preserving the context of searches and allows relevance feedback to generate alternative result sets. At the moment, the system is limited to two data sources.

Numerous tools exist that focus on storing and managing a personal library or citations. A popular example is Mendeley[2] which also offers different front ends and collaborative features. Other citation tools include CiteULike[3] and Connotea[4]

CoSearch [1] is a collaborative web search tool. It offers a user interface that can simultaneously take input from different users sharing a single machine. Mobile devices can be used to contribute to a collaborative search session. Data is acquired by using a popular web search engine.

Development Platforms

SpidersRUs Digital Library Toolkit [8] is a search engine development tool. The developers strove for a balance between easiness of use and customizability. The toolkit also features

[1] *ezDL* is licensed under GPL v3. Other licenses can be used on request. The main web site for developers and further information can be found here: `http://ezdl.de/developers`

SIGIR 2012 Workshop on Open Source Information Retrieval (OSIR 2012)
Portland, Oregon, USA

[2] `http://www.mendeley.com/`
[3] `http://www.citeulike.org/`
[4] `http://www.connotea.org/`

a GUI for the process of search engine creation. Results presentation follows common standards of popular web search engines. Support for complex search sessions, e.g. a tray or citation management tool are not included.

Evaluation Systems

The Lemur project[5] includes a query log tool bar that can be used to capture usage data. It can collect queries as well as user interaction such as mouse activity and is available as open source.

Bierig et al [7] presented an evaluation and logging framework for user-centered and task-based experiments in interactive information retrieval that focuses on "multidimensional logging to obtain rich behavioural data" of searchers.

2. CONCEPTS

As a re-implementation of the Daffodil project [11], *ezDL* builds on many of the same concepts and principles as Daffodil. Like Daffodil it is a "search system for digital libraries aiming at strategic support during the information search process" [16]. Its primary target group is not that of casual users using a search system for short ad-hoc queries. Instead the software aims to support searchers during complex information tasks by addressing all the steps in the *Digital Library Life Cycle*, as well as integrating search models originally proposed by Marcia Bates [2, 3, 4].

The *Digital Library Life Cycle* divides the information workflow into five phases [18], beginning with the *discovery* of information resources, which in *ezDL* is supported through the Library Choice view. This is followed by the *retrieval* phase of information search, the *collating* of found information using the personal library and tagging, *interpreting* the information, and finally the *re-presenting* phase where new information is generated. In all phases, different so-called tactics or stratagems can be employed by searchers or information workers, which we try to support through *ezDL*.

The notion of tactics and stratagems as higher-level search activities was introduced by Bates [2, 3, 4]. Based on search tactics used by librarians and expert searchers, Bates describes basic *moves*, as well as higher-level *tactics*, *stratagems*, and finally *strategies* that build on lower-level activities.

ezDL already offers direct support for some of those higher-level activities, e.g. through the use of sharing functionalities to support collaborative idea generation, through term suggestions of synonyms or spelling variants, extraction of common results terms, or through icons in the result items that allow easy monitoring of performed activities.

During query formulation, *ezDL* provides term suggestions to the user (e.g. synonyms and related terms). These are an example for the concept of proactive system support. Bates describes "five levels of system involvement (SI) in searching" [4]. The proactive support of *ezDL* belongs to the third level, where a search system (through monitoring of user activities) can react to the search situation without prompting by the user. Users are informed of improvement options for their current move. Jansen and Pooch [12] demonstrated that proactive software agents assisting users during their search can result in improved performance of users. The effectiveness of such suggestions has also been shown for the Daffodil system [20].

[5] `http://www.lemurproject.org/`

Tran [21] implemented a prototype of a support tool for the pearl growing stratagem. The tool shows citation relationships between documents in a graph and allows the user to follow these relationships and keep track of the search progress using document annotations. Figure 1 shows a screenshot of a pearl growing session with some documents marked as relevant. It is planned to include this tool in *ezDL* in the near future.

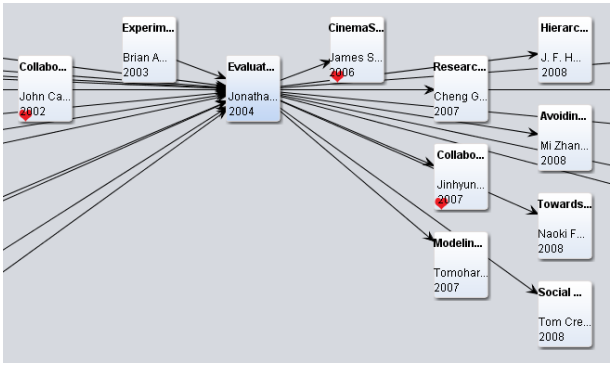

Figure 1: A close-up of the pearl growing tool

Proactive support of higher-level activities, such as suggestion of tactics and stratagems for improvable search situations [14, 15] or suggestion of search strategies with scaffolding support, is currently planned and will likely be available for *ezDL* within the next nine months.

3. ARCHITECTURE

ezDL is a continuation of the Daffodil project and therefore shares its main ideas: meta-search in digital libraries and strategic support for users. Its overall architecture likewise has inherited many features from Daffodil. Figure 2 provides a high-level overview of the system.

The system architecture makes extensive use of separation of concerns to keep interdependencies to a minimum and make the system more stable. This is true on the system level where a clear separation exists between clients and backend, but also within the backend itself, where individual "agent" processes handle specific parts of the functionality, and even within these agents. The desktop client, too, is separated into multiple independent components called "tools". *ezDL* is completely written in Java using common frameworks and libraries.

3.1 The Backend

The backend provides a large part of the core functionality of *ezDL*: the meta-search facility, user authorization, a knowledge base about collected documents, as well as wrappers and services that connect to external services. Functionality that provides collaboration support and allows storing of documents and queries in a personal library is also located here.

The right part of Figure 2 shows the structure of the backend. The components of the backend are agents: independent processes that provide a specific functionality to the system. Agents use a common communication bus for transferring messages between each other.

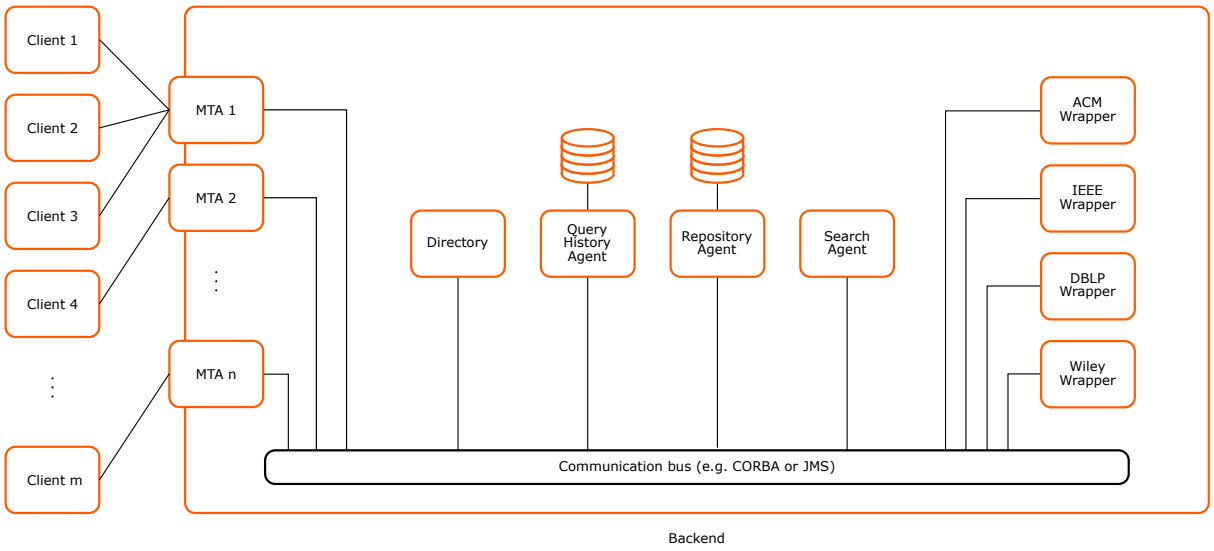

Figure 2: Overall architecture of *ezDL*

The Service-based Agent Infrastructure

Since every kind of functionality is taken care of by different agents, the crash of one agent generally only disrupts this particular functionality. For example, if the search agent crashes, detail requests and the personal library are still working. Also, it is possible to run multiple agents of each kind for load balancing and as a fail-safe mechanism.

Agents are subdivided into the main agent behaviour (registering with the Directory, sending and receiving messages, managing resources) and components that deal with specific requests. These components—the request handlers—are independent and process requests concurrently.

Beginning on the left, an MTA (Message Transfer Agent) is an agent that provides clients with a connection point to the backend. MTAs are responsible for authenticating users and translating requests from clients into messages to certain agents. E.g., if a client requests a search for a given query, the query from the client is translated into a message to the Search Agent. This mechanism creates a clear separation between the client view of the system and the internal workings: the client doesn't have to know how many agents are serving search queries and new search agents could be instantiated as the system load demands. Currently there is only one MTA implementation, which uses a binary protocol over a TCP connection, but it is possible to provide other protocols—e.g. SOAP—by using separate MTA implementations.

The Directory is a special agent that keeps a list of agents and the services they provide. Upon start, each agent registers with the Directory and announces the services it provides.

The connection to remote (or local) search services (e.g., digital libraries or information retrieval systems) is managed by wrapper agents—in Figure 2 the four agents on the right hand side. They translate the internal query representation of *ezDL* into one that the remote service can parse and translates the response of the remote service into an appropriate document representations to be handled by *ezDL*.

Example: Running Search Queries

If a client requests a search, it sends a request to the MTA with a query in *ezDL* notation and a list of remote services that the query should be run on. The request is handled by the MTA which forwards it to the Search agent. The Search agent asks the Directory for the name of agents that provide a connection to the remote services requested by the client. After receiving that list, the Search agent forwards the query to each of these agents. The agents then translate the query into something that the remote service understands and sends the answer of the remote service back to the search agent. The search agent collects all answers from all the remote services, merges duplicates and reranks them. Reranking is either performed by using the original RSVs or by using standard Lucene[6] functionality. The answer set is then sent back to the MTA that requested the search. The MTA sends the answer to the client. The search agent also forwards the collected documents to the repository agent which is responsible for serving requests for details on documents (e.g., if the user wants to see the full text).

3.2 The Frontend

There are multiple frontends for *ezDL*: among them the basic desktop client and a web client. Specialized frontends exist for various applications (see Use Cases). Clients for iOS and Android tablets are currently being developed. This subsection details the architecture of the desktop client, since this is the main client for *ezDL*.

Tools and Perspectives

A *tool* comprises a set of logically connected functionalities. Each tool has one or more *tool views*, interactive display components that can be placed somewhere on the desktop. A configuration of available tools and the specific layout of their tool views on the desktop is called a *perspective*. Users can modify existing predefined perspectives as well as create custom perspectives. The desktop client already has many

[6] `http://lucene.apache.org/core/`

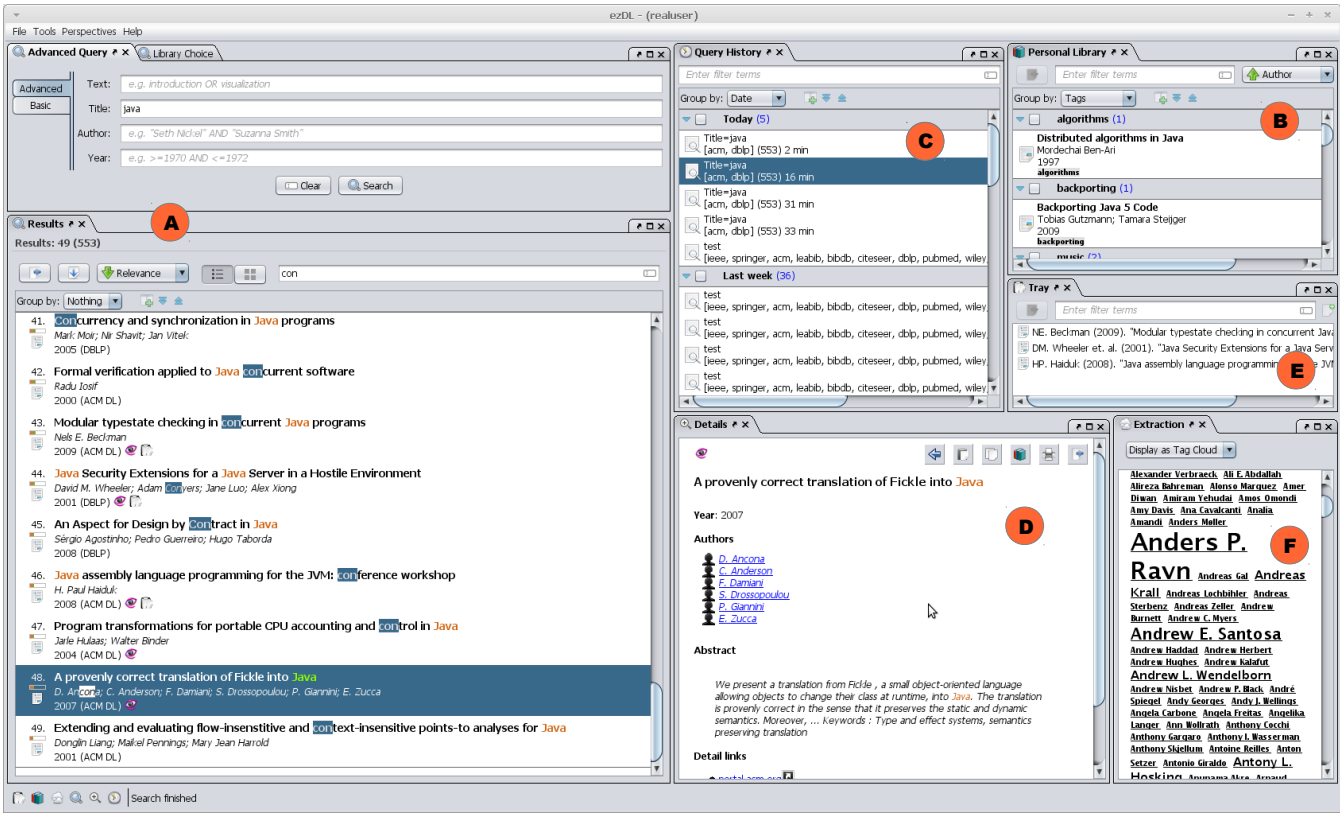

Figure 4: The Desktop client during a search session

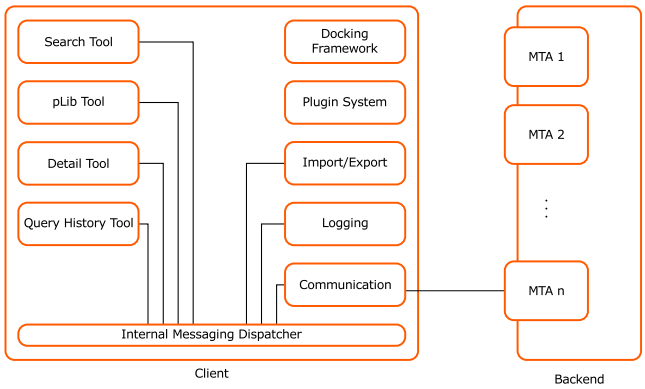

Figure 3: Architecture of the desktop client

built-in tools and functionalities and can be easily extended (see Figure 4):

- The Search Tool (A) offers a variety of query forms for different purposes and views to present the results in list or grid form, as well as a Library Choice view for selecting information sources. Results can be sorted or grouped by different criteria, filtered, and exported. An extraction function (F) can be used to extract frequent terms, authors, or other features from the result and visualize them in form of a list, a bar chart or a term cloud. Grouping criteria or extraction strategies are encapsulated and new ones can easily be added, as can be new renderers for result surrogates.
- The Personal Library (B) allows to store documents or queries persistently for authenticated users. Within their personal collection, users can filter, group and sort (e.g. by date of addition), organize the documents with personal tags, and share them with other users. Additional documents can be imported into the personal library as long as their metadata is available in BibTeX format.
- The Search History (C) lists past queries for re-use and allows grouping by date and filtering.
- The Detail View (D) shows additional details on individual documents, such as thumbnails or short summaries where available, or additional metadata not included in the surrogate that is shown in the result list. A detail link can be provided to retrieve the fulltext.
- A Tray (E) can be used to temporarily collect relevant documents within a search session.

Communication with the Backend

Like the backend, the desktop client uses a messaging infrastructure for communication between otherwise independent components. In Figure 3 a diagram of the components is shown. On the left, four of the available tools can be seen with their connection to the internal communication infrastructure (search, personal library, details, and query

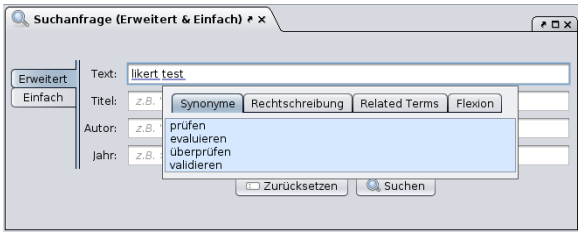

Figure 5: The search form with suggestions

history). On the right hand side, a few subsystems are presented, one of which is the external communication facility that connects the client to the backend.

As an example, if the user enters a query in the search tool and presses the "search" button, an internal message is sent to the communication facility, which transmits the query to the backend. When the answer is received, the communication facility routes the message back to the search tool.

Since from the client's point of the view the backend is hidden behind the MTA, further details are omitted in the backend part of Figure 3.

Query Processing and Proactive Support

The queries that users enter are expressed in a grammar specific to *ezDL* that is quite flexible and allows simple queries like `term1 term2 term3` as well as more complicated ones like `Title=term1 AND (term2 NEAR/2 term3)`. Internally, the query is represented as a tree structure that can also keep images as comparison values so *ezDL* can be used to specify image search queries.[7]

During query formulation, the user's interaction is observed by the system. If the system notices a break in the user's typing, the query is processed by modules of the proactive support subsystem that can either ask the backend for suggestions or calculate them directly in the frontend. The suggestions can replace query terms e.g. by spelling corrections, insert new terms (e.g., for synonym expansion), or tag terms with concepts from an ontology. The ontology items become part of the query so that a query can contain both plain text terms and ontology terms. When suggestions are found for a term, the term is marked by an underline in the query text field and a popup list is shown that presents the suggestions (see Figure 5).

3.3 Extending and Customizing ezDL

Each agent and the desktop client are extensible using a plugin system. Plugins are registered at a central component that can later be asked to return plugin objects of a specific type. As an example, it is possible to add a new proactive suggestion module to the system that implements a new way of retrieving suggestions. Also the popup list that shows the suggestions can be replaced by an alternative. Further uses of the plugin system are export and import modules and modules that extract information from the result list.

Adding a new service is usually done by implementing a new agent. There is an abstract class that takes care of most issues but the actual functionality. This is usually implemented using specialized classes (request handlers) for which there are abstract implementations, too. Thus, developers can concentrate on the business logic.

Connecting to a new collection for searching (a digital library, a local IR system, a BibTeX file, etc.) is accomplished by implementing a wrapper agent. These are agents specialized in translating between *ezDL* and a remote system. Remote systems can be those that provide a stable API like SOAP or SQL but also those that only have a web site and a search form. *ezDL* has built-in support for most common fields (e.g. title, author, publication year, abstract) and data types (e.g. text, numbers, images). There are abstract wrappers available to quickly connect to a Solr[8] server. If required, web pages can be scraped using an elaborate tool kit that is configured by an XML file. Because of this, even digital libraries without a proper API can be connected. Sometimes, digital libraries change their web page layout, breaking scripts that parse their HTML. Configuring the page scraping using an XML file makes automatic repairs of the configuration possible. See [10] for an example implementation based on Daffodil, The approach outlined in this work uses repeated queries to infer the template elements of the web pages and step-wise generalisation to find the location of known information on the page.

There is also a library of code for translating the *ezDL* query representation into other languages.

Agents—and, thus, wrapper agents—announce themselves to the Directory agent when started. The client can ask the backend for a list of known wrapper agents, so there is no need to change any code or configuration outside of the agent. This also enables developers to store the code and put it under version control independent from the main *ezDL* code.

Often, services in the back end are used in the client in an individual tool. One example for this is the search facility, which consists of the search tool in the Swing client and the search agent in the back end. Writing a new tool for the Swing client can be done by implementing an OSGi plugin. The tool code itself is fairly simple since there is an abstract implementation for the glue code. The remaining task is implementing a Swing GUI and communicating with the back end by firing events and listening for an answer.

4. EVALUATING SEARCH SYSTEMS

To support user-centred evaluations, *ezDL* has a builtin evaluation mode that addresses many of the major challenges inherent in setting up evaluation tasks and tracking user activity during the experiments. The following is a brief overview of those functionalities within *ezDL* directly designed to support evaluations.

Logging user actions

For evaluations with actual users all user actions performed with the system should be logged for later inspection and analysis. *ezDL* has a built-in logging facility that stores all the interaction data of the user in a relational database (currently *mysql* is used). A *log session* comprises all *log events* that a user or the system has triggered. A log event has *i*) a unique name identifying this type of event, *ii*) timestamps from the frontend and the backend, *iii*) a sequence number

[7]This mechanism is used in the Khresmoi project (see Section 5) to allow general physician and radiologists to search for medical images.

[8]`http://lucene.apache.org/solr/`

to ensure the correct order, and *iv*) parameters as multiple key/value pairs. For example, when a user performs a search for `information retrieval` in the DBLP and ACM digital library the corresponding log event may look like this:

```
event:
  name: "search"
  clientTimestamp: 1/4/2012 15:26:32,1234
  timestamp: 1/4/2012 15:26:32,3456
  sequenceNumber: 10
  parameters:
      query: "information retrieval"
      sources: dblp, acm
```

The logging facility takes care about allocating activities to sessions and users. If it is required to log some previously unlogged action, this can be simply integrated by sending a corresponding logging message to the backend.

Tracking AOIs

Gaze tracking is a method for user-centred evaluation that has recently gained popularity within the IR field [5, 7]. A challenge for logging highly interactive systems with changing interfaces and moving components is keeping track of their position, so that gaze points or fixation data of users can be aggregated across so-called *Areas of Interest* (AOIs, see Figure 6). This feature has been integrated into *ezDL* with the help of the *AOILog* framework [6].

Fixed layout on screen

The layout of the desktop can be locked to keep UI-related variance low. With a fixed layout it is no longer possible for a test subject to open additional tool views or change the layout of the desktop.

Loading predefined perspectives

Predefined perspectives can be loaded immediately after the system has been started. This allows the evaluator to create custom perspectives that can be used for an evaluation without selecting them manually.

Splash screen for choosing evaluation settings

A splash screen can be enabled that is shown before starting the system. It can be used to choose and set settings for the evaluation session, e.g. a search task description or the system variant when doing a comparison of different UIs or system features.

Several user studies have been performed and experimental systems implemented using *ezDL* as a base system. The next section will present some of them in more detail.

5. USE CASES

ezDL is currently running as a live system, and is being used and extended in a number of projects of various sizes.

The live system[9] features all core functionalities that part of a more specific project. These include a simple and an advanced search function, various result manipulation options, a temporary document store, and exporting of meta information (e.g. in BibTeX format). Registered users can also use a personal library to store, annotate and share found or imported documents. Currently, nine different digital libraries are connected to the system focusing on computer

[9] `http://www.ezdl.de/`

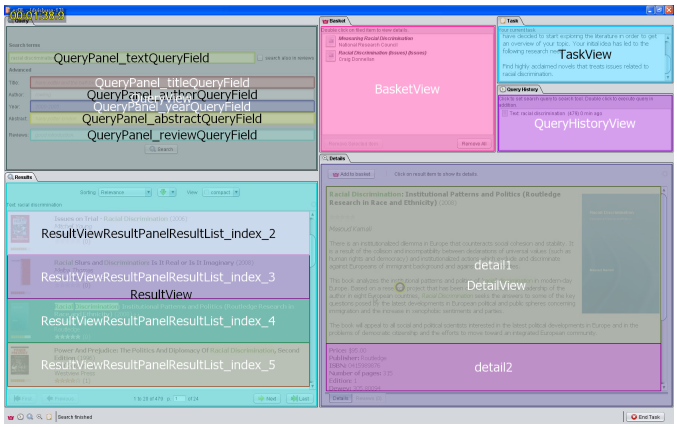

Figure 6: An *ezDL* client overlayed with AOIs for eye tracking (from the HIIR project)

science libraries, but including others like Pubmed and the Amazon catalogue. The system is publicly available, still under constant development and updated regularly.

Khresmoi[10] is a four year project funded by the EC, which aims at building a multilingual and multimodal search system for biomedical documents. The *ezDL* framework is used for all user interfaces developed within the project. These include variations of the stand-alone Swing client, such as an interface for search medical images, including 3D data. Another version of the interface will be customized to the needs of general practitioners. For casual users with health related information needs, an easy to use web based interface (see Figure 7) is under development. From a functional point of view numerous new data sources were made available. The set of searchable data types was extended to cover the specific demands of the medical domain. The system allows the user to specify an image as a query to perform a similarity search. Additional collaborative and social functions will be added to the full client in later versions.

Within the INEX 2010 Interactive Track *ezDL* was used to observe how users act during their search sessions [19]. Valuable insights on user behaviour were gained. An application for viewing the logged data and a questionnaire tool controlling the experiment flow have been implemented.

For the ongoing CAIR[11] project an advanced 2010 INEX *ezDL* version is used. To answer the question whether clustering of results can improve efficiency of searches with vague information needs *ezDL* was expanded by a clustering service and visualization [17] (see Figure 8). New data sources and a browser were added to the system. For the evaluation a task selection and a questionnaire tool were developed. Log data generated by *ezDL* can be analysed automatically.

The AOI logging framework mentioned in Section 4 was implemented as part of the HIIR (Highly Interactive IR) project[12]. The project's goal is improving interaction with the system by considering the users cognitive efforts [23, 22].

6. CONCLUSION AND OUTLOOK

[10] `http://www.khresmoi.eu/`

[11] `http://www.uni-weimar.de/cms/index.php?id=17632`

[12] `http://www.is.inf.uni-due.de/projects/hiir/index.html.en`

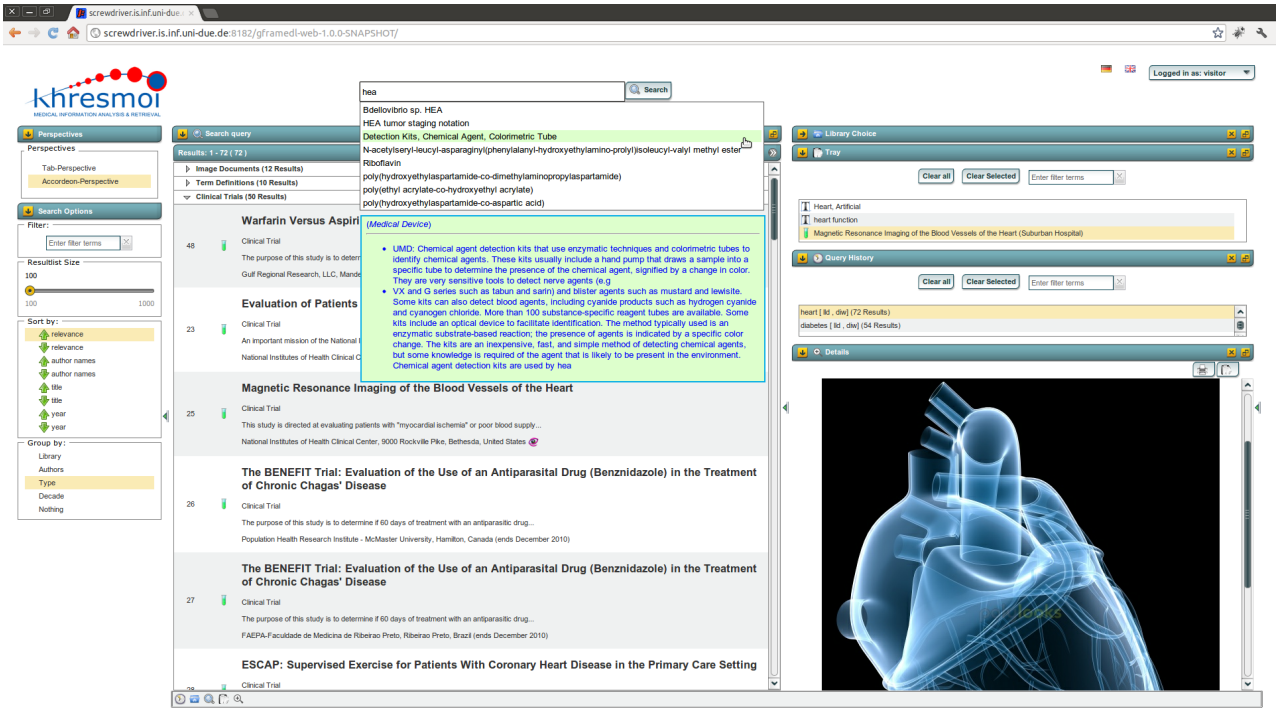

Figure 7: The web client used in the Khresmoi project

We presented *ezDL*, which is a framework system for interactive retrieval and its evaluation. Building upon state-of-the art interface technology and usability results, *ezDL* can provide an advanced user interface for many IR applications. The system can also be easily extended, at the functionality level as well as at the presentation level; thus, new concepts for the design of IR user interfaces can be integrated into *ezDL* with little effort. Furthermore, the system provides extensive support for performing user-oriented evaluations. In the same way as there are various experimental IR back-end systems, there is now an IR frontend system that allows for easy experimentation and application of interactive retrieval.

Acknowlegments

The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement №257528 (KHRESMOI). Parts of this work were supported by the German Science Foundation under grant nos. FU 205/24-1 (HIIR) and FU 205/22-1 (CAIR).

7. REFERENCES

[1] S. Amershi and M. R. Morris. Cosearch: a system for co-located collaborative web search. In *Proceedings of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, CHI '08, pages 1647–1656, New York, NY, USA, 2008. ACM.

[2] M. J. Bates. Idea tactics. *Journal of the American Society for Information Science*, 30(5):280–289, 1979.

[3] M. J. Bates. Information search tactics. *Journal of the American Society for Information Science*, 30(4):205–214, 1979.

[4] M. J. Bates. Where should the person stop and the search interface start? *Information Processing and Management*, 26(5):575–591, 1990.

[5] T. Beckers and N. Fuhr. User-oriented and eye-tracking-based evaluation of an interactive search system. In *4th Workshop on Human-Computer Interaction and Information Retrieval (HCIR 2010) @ IIiX 2010*, 2010.

[6] T. Beckers and D. Korbar. Using eye-tracking for the evaluation of interactive information retrieval. In *Proceedings of INEX 2010*, pages 236–240, 2011.

[7] R. Bierig, J. Gwizdka, and M. Cole. A User-Centered Experiment and Logging Framework for Interactive Information Retrieval. In *Understanding the user - workshop in conjuction with SIGIR'09*, 2009.

[8] M. Chau and C. H. Wong. Designing the user interface and functions of a search engine development tool. *Decision Support Systems*, 48(2):369 – 382, 2010.

[9] A. Diriye and G. Golovchinsky. Querium: a session-based collaborative search system. In *Proceedings of the 34th European conference on Advances in Information Retrieval*, ECIR'12, pages 583–584, Berlin, Heidelberg, 2012. Springer-Verlag.

[10] A. Ernst-Gerlach. Semiautomatisches Pflegen von Wrappern. Diplomarbeit, Universität Dortmund, FB Informatik, 2004.

[11] N. Fuhr, C.-P. Klas, A. Schaefer, and P. Mutschke. Daffodil: An integrated desktop for supporting high-level search activities in federated digital

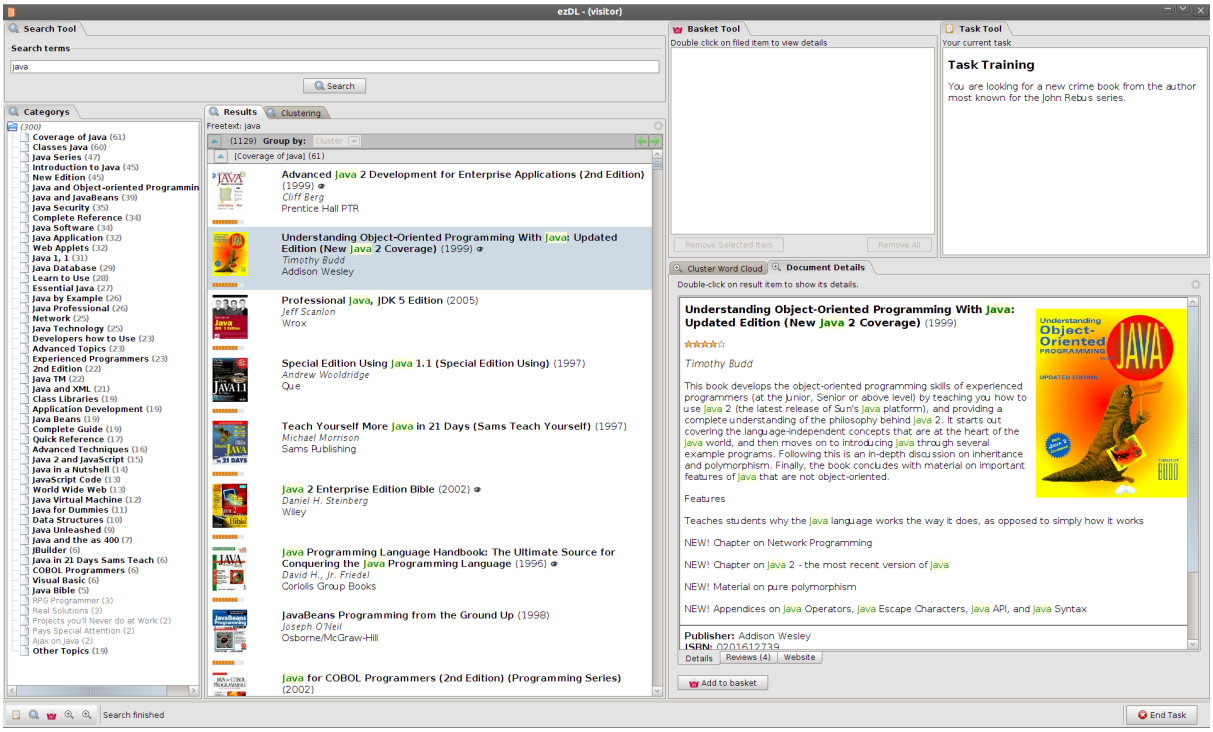

Figure 8: The *ezDL*-based system extended with clustering functionality developed within the CAIR project

libraries. In *Research and Advanced Technology for Digital Libraries. 6th European Conference, ECDL 2002*, pages 597–612, Heidelberg et al., 2002. Springer.

[12] B. J. Jansen and U. Pooch. Assisting the searcher: utilizing software agents for web search systems. *Internet Research: Electronic Networking Applications and Policy*, 14(1):19–33, 2004.

[13] C.-P. Klas, S. Kriewel, and A. Schaefer. Daffodil - Nutzerorientiertes Zugangssystem für heterogene digitale Bibliotheken. *dvs Band*, 2004.

[14] S. Kriewel. *Unterstützung beim Finden und Durchführen von Suchstrategien in Digitalen Bibliotheken*. PhD thesis, University of Duisburg-Essen, 2010.

[15] S. Kriewel and N. Fuhr. An evaluation of an adaptive search suggestion system. In *32nd European Conference on Information Retrieval Research (ECIR 2010)*, 2010.

[16] S. Kriewel, C.-P. Klas, A. Schaefer, and N. Fuhr. Daffodil - strategic support for user-oriented access to heterogeneous digital libraries. *D-Lib Magazine*, 10(6), June 2004. available at `http://www.dlib.org/dlib/june04/kriewel/06kriewel.html`.

[17] M. Lechtenfeld and N. Fuhr. Result clustering supports users with vague information needs. In *Proceedings of the 12th Dutch-Belgian Information Retrieval Workshop 2012, Ghent, Belgium*, February 2012.

[18] A. Paepcke. Digital libraries: Searching is not enough–what we learned on-site. *D-Lib Magazine*, 2(5), May 1996. `http://www.dlib.org/dlib/may96/stanford/05paepcke.html`.

[19] N. Pharo, T. Beckers, R. Nordlie, and N. Fuhr. Overview of the inex 2010 interactive track. In *Proceedings of the 9th International Workshop of the Initiative for the Evaluation of XML Retrieval (INEX 2010)*, 2011.

[20] A. Schaefer, M. Jordan, C.-P. Klas, and N. Fuhr. Active support for query formulation in virtual digital libraries: A case study with DAFFODIL. In A. Rauber, C. Christodoulakis, and A. M. Tjoa, editors, *Research and Advanced Technology for Digital Libraries. Proc. European Conference on Digital Libraries (ECDL 2005)*, Lecture Notes in Computer Science, Heidelberg et al., 2005. Springer.

[21] V. T. Tran. Entwicklung einer Unterstützung für Pearl Growing. Diplomarbeit, Universität Duisburg-Essen, 2011.

[22] V. T. Tran and N. Fuhr. Quantitative analysis of search sessions enhanced by gaze tracking with dynamic areas of interest. In *The International Conference on Theory and Practice of Digital Libraries 2012*. Springer tbp, September 2012.

[23] V. T. Tran and N. Fuhr. Using eye-tracking with dynamic areas of interest for analyzing interactive information retrieval. In *Proceedings of the 35th international ACM SIGIR conference on Research and development in Information Retrieval*. ACM tbp, August 2012.