Grant Agreement Number: 257528

KHRESMOI

www.khresmoi.eu

Prototype of a first search system for intensive tests

Deliverable number	*D8.3*
Dissemination level	*Public*
Delivery data	*29 August 2012*
Status	*Final*
Authors	*Allan Hanbury, William Belle, Nolan Lawson, Ljiljana Dolamic, Natalia Pletneva, Matthias Samwald, Célia Boyer*

This project is supported by the European Commission under the Information and Communication Technologies (ICT) Theme of the 7th Framework Programme for Research and Technological Development.

Executive Summary

This deliverable presents an overview of the current state of the prototypes developed in the Khresmoi project for the medical professional and general public use case, and provides links to execute these prototypes. Two complementary Khresmoi prototypes have been developed:

Khresmoi Integration Platform: This system is built on the Khresmoi integration platform developed in Work Package 6, the experimental platform for the integration of cutting edge technologies developed in Khresmoi. It is the principal integrated outcome aimed at the medical professional and general public user groups. The principal services used during the interactive search phase in this prototype are:

- ezDL, developed in WP3, which provides the comprehensive search user interface
- Mimir, developed in WP1, which provides the text search capabilities over annotated text
- GIFT, developed in WP2, which provides content-based 2D image search
- OWLIM, developed in WP5, which provides query disambiguation and suggestion services
- Multilingual tools (based on MOSES), developed in WP4, which provide query and document translation services
- Spell-checking tools developed in WP8, which provide suggestions of correct spelling

It aims to fulfil the user requirements presented in Deliverable 8.2 [2] as well as provide new features that end users cannot imagine possible yet.

Khresmoi Classic Search System: This search engine is built on commonly used open source software, such as Lucene. This system is an upgrade of an existing search system at the Health on the Net Foundation (HON) to a modular structure and to the latest open source technologies. It represents an approach widely adopted for creating domain-specific search engines with intermediate development effort. The main purpose of this search engine is to provide basic search engine components to the Khresmoi integration platform, but also to serve as a feasibility study of a Khresmoi exploitation model — the adoption of Khresmoi components into existing search engines.

Finally, a baseline search system using a popular search customisation tool (Google Custom Search) was created. It demonstrates the most basic approach to creating a domain-specific search engine, and illustrates what is possible with minimal effort using current off-the-shelf-tools. It is mainly aimed at providing support in preparing the evaluations.

For each system, the extent to which the system satisfies the user requirements outlined in Khresmoi Deliverable 8.2 is described.

Table of Contents

1 Introduction **4**
1.1 Khresmoi Integration Platform 4
1.2 Khresmoi Classic Search System 5
1.3 Google Custom Search Baseline System 6
1.4 Comparison of the three systems 6

2 Khresmoi Integration Platform **6**
2.1 Description 6
2.2 Prototype access 9
2.3 Use case requirement satisfaction 9

3 Khresmoi Classic Search System **11**
3.1 Description 11
3.1.1 Third-Party Libraries 12
3.1.2 Crawler 13
3.1.3 Document content extraction and classification 14
3.2 Search Engine 17
3.3 Prototype access 20
3.4 Use case requirement satisfaction 20

4 Google Custom Search Baseline System **23**
4.1 Description 23
4.2 Prototype access 23
4.3 Use case requirement satisfaction 24

5 Conclusion **27**

6 References **28**

Abbreviations

API	Application Programming Interface
AJAX	Asynchronous JavaScript And XML
CSE	Custom Search Engine
HTTP	Hyper Text Transfer Protocol
JSON	JavaScript Object Notation
JVM	Java Virtual Machine
REST	Representational State Transfer
TF/IDF	Term Frequency/Inverse Document Frequency
UI	User Interface
URL	Uniform resource locator
XML	Extensible Markup Language

1 Introduction

This deliverable presents an overview of the current state of the prototypes developed in the Khresmoi project for the medical professional and general public use case, and provides links to execute these prototypes. Two complementary Khresmoi prototypes have been developed:

Khresmoi Integration Platform: This system is built on the Khresmoi integration platform developed in Work Package 6, the experimental platform for the integration of cutting edge technologies developed in Khresmoi. It is the principal integrated outcome aimed at the medical professional and general public user groups. It aims to fulfil the user requirements presented in Deliverable 8.2 [2] as well as provide new features that end users cannot imagine possible yet.

Khresmoi Classic Search System: This search engine is built on commonly used open source software, such as Lucene. This system is an upgrade of an existing search system at the Health on the Net Foundation (HON) to a modular structure and to the latest open source technologies. It represents an approach widely adopted for creating domain-specific search engines with intermediate development effort. The main purpose of this search engine is to provide basic search engine components to the Khresmoi integration platform, but also to serve as a feasibility study of a Khresmoi exploitation model — the adoption of Khresmoi components into existing search engines.

Finally, a baseline search system using a popular search customisation tool (Google Custom Search) was created. The two prototypes and the baseline search system, as well as their purposes within the Khresmoi project are summarised in the remainder of this section, and then treated in detail in Sections 2 to 4.

1.1 Khresmoi Integration Platform

The Khresmoi integration platform (subsequently referred to as *integration platform*) comprises an integration of components developed by Khresmoi Work Packages 1 to 5 (described to a large extent in the following deliverables [10, 18, 9, 22]) within the integration framework developed in Work Package 6 [12, 25, 19]. The Khresmoi integration platform serves the following purposes:

- It is the main Khresmoi demonstration system.
- It is the central platform for the full integration of all Khresmoi components in a scalable way.
- It is a test platform for new developments and technologies.

1.2 Khresmoi Classic Search System

The Khresmoi Classic Search System (subsequently referred to as *classic prototype*) represents the state-of-the-art approach to search engine development. It was designed using the classic approach to building search engines with Lucene [3], and was based on a modular approach.

It is based on the search engine service that HON has been providing to its users for over 10 years, for which it has built up a large user base. Users can access a search engine from the HON web site, from numerous HONcode-certified web sites (webmasters may voluntarily add a search box to their web site along with the HONcode seal) and from a browser toolbar which has over 1000 downloads per week. Once this prototype is finalized and stable, it will replace the current HON search engines and will serve the same users. Further on, it is planned that additional technologies developed in Khresmoi will be incorporated over time.

The starting point for this prototype was WRAPIN [1] (an EU-funded project during 2001–2003) and is an enhancement of WRAPIN technologies. All previous HON experience in the development of health content search engines such as WRAPIN, HON Question-Answering[1] (developed in the framework of the EU-funded project PIPS during 2004–2008) and others have been taken into account in the development of this prototype.

The classic search engine serves the following purposes:

- It demonstrates a conventional approach to building a domain-specific search engine using commonly available tools and requiring an intermediate amount of effort.
- It provides a platform to test a Khresmoi exploitation model. It is expected that many providers of search systems will not be prepared to make excessive sudden changes to their search systems. An exploitation model considered in Khresmoi is that individual components developed in Khresmoi can be adopted and integrated into existing search systems in a step-by-step way. The classic prototype provides a platform to test the feasibility of this exploitation model through step-by-step adoption of Khresmoi technologies, and to collect experience in the implementation of this exploitation model.
- Due to HON's established user base, it is expected that usage statistics (e.g. query logs and click logs) from the classic prototype may be leveraged to inform development of the integration platform.
- It provides the crawled documents to be indexed by the integration platform.
- Some of its components are integrated in the integration platform. For instance, the spellchecker is already integrated, and the documents collected by the crawler are shared. It hence provides the basic search components to the integration platform.
- It serves as a source of results for pooling in the global empirical evaluation.
- Some of the capabilities serve as a basis for further development and benchmarking, e.g. the disease classification and thematic classification.

[1] http://services.hon.ch/cgi-bin/QA10/qa.pl

1.3 Google Custom Search Baseline System

The Google Custom Search Engine (CSE) system[2] is at present the simplest approach to constructing a search engine over a limited set of sites. It represents a commonly used approach requiring a minimal amount of development effort. Google Custom Search allows domain-specific search engines to be rapidly built by limiting the Google search to a specified list of websites. This baseline search system serves the following purposes:

- It demonstrates the most basic approach to creating a domain-specific search engine, and illustrates what is possible with minimal effort using current off-the-shelf-tools.
- It serves as a search engine to explore the medical websites to be used in Khresmoi during the preparation of the evaluation tasks (which is started before the indexing by the full system is complete).
- It serves as a source of results for pooling in the global empirical evaluation.

1.4 Comparison of the three systems

In the next three sections, descriptions of the three search systems are presented, along with links to access the systems. For each system, a short analysis on the extent to which the user requirements presented in [2] are met is also presented.

As the classic prototype is described only in this deliverable, whereas the details on the integration platform have been extensively covered in many deliverables, the description of the classic prototype goes into more technical detail.

2 Khresmoi Integration Platform

2.1 Description

The Khresmoi Integration Platform is built on the Services Orchestration Cloud developed in WP6 and described in D6.4.1 [19]. The diagram from D6.4.1 in Figure 1 summarises the architecture. The principal services used during the interactive search phase in the prototype targeted at the general public and physicians are:

- ezDL, developed in WP3, which provides the comprehensive search user interface
- Mimir, developed in WP1, which provides the text search capabilities over annotated text
- GIFT, developed in WP2, which provides content-based 2D image search

[2] http://www.google.com/cse/

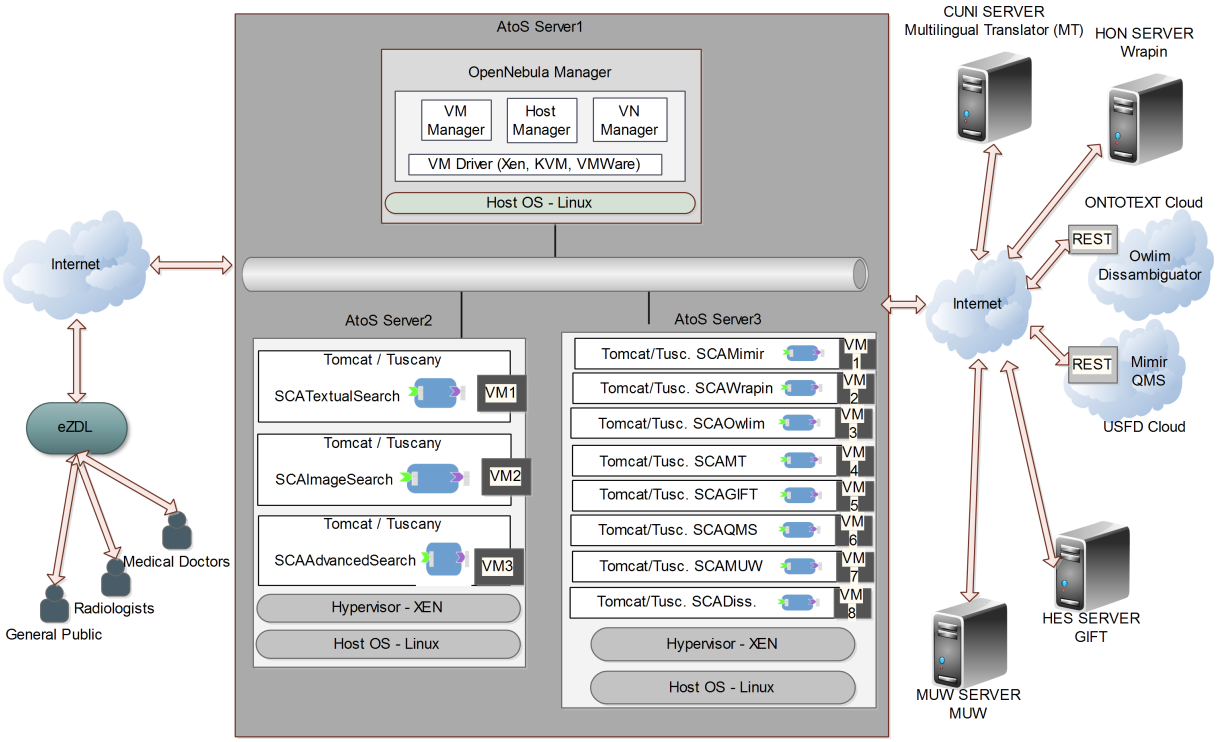

Figure 1: Khresmoi architecture (from D6.4.1)

- OWLIM, developed in WP5, which provides query disambiguation and suggestion services
- Multilingual tools (based on MOSES), developed in WP4, which provide query and document translation services
- Spell-checking tools (from the classic prototype), developed in WP8, which provide suggestions of correct spelling

Crawling, pre-processing and indexing of the data are not yet fully integrated into the architecture, due to the batch nature of these processes. At present, crawling of the data to index are done by HON and HES-SO in WP8 and WP9. Annotation of the text data is done by GATE (WP1) interacting with OWLIM (WP5), while indexing of the text is done by Mimir (WP1). Indexing of the image data is done by GIFT (WP2). The prototype currently indexes all of the HON-certified websites, as well as the list of resources targeted at physicians created in WP8 [2].

The prototype is accessed through ezDL. ezDL consists of a back end containing the core technologies, which is accessed using front ends (user interfaces). Two such user interfaces have been developed. The Java Swing-based interface, shown in Figure 2, is the most comprehensive and provides the capabilities to include the most comprehensive interface and search features. The comprehensive interface is annotated with its most important features in Figure 2. These are:

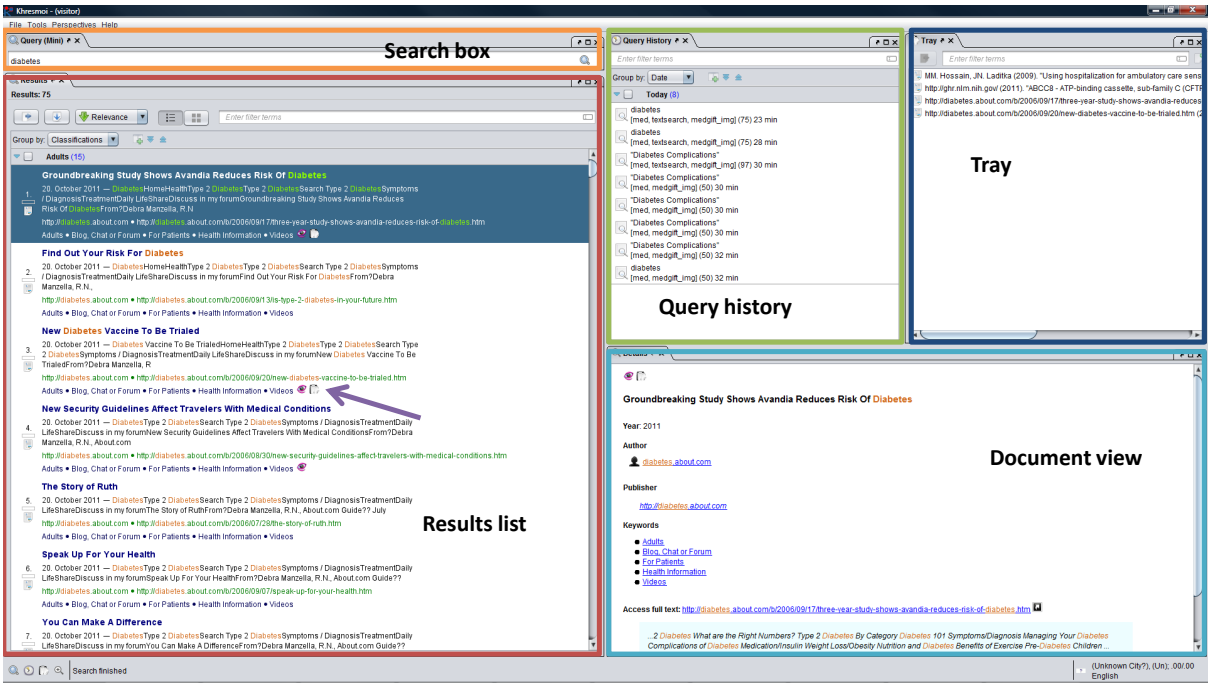

Figure 2: Khresmoi Java Swing-based interface

Search box: The simple search box is shown. Advanced search is also available.

Result list: A summary of each document is given. Controls at the top allow the results to be reordered or grouped by a number of criteria and filtered by terms. The icons indicated by the arrow show when a document has been viewed (eye icon) and moved to the tray (clipboard icon).

Tray: Documents can be dropped here to be stored for future use. Logged in users have access to personal libraries, allowing more flexibility in organisation, including the capability to add tags to results.

Query history: Lists details on all queries entered (including date and time) and allows queries to be repeated.

Document view: All data available on the document selected in the results list are shown, including a link to the full document.

While these comprehensive features are extremely useful for expert searchers, this interface has the disadvantage of running externally to a web browser, and hence involves an additional hurdle to getting started for novice users. For this reason, a lighter web-based interface to ezDL has been developed, shown in Figure 3.

Both interfaces can be reorganised into different *perspectives* to satisfy the requirements of different users or user groups. As an example, the web interface in Figure 3 is configured with a simplified interface perspective showing only the search box, result list and document view components.

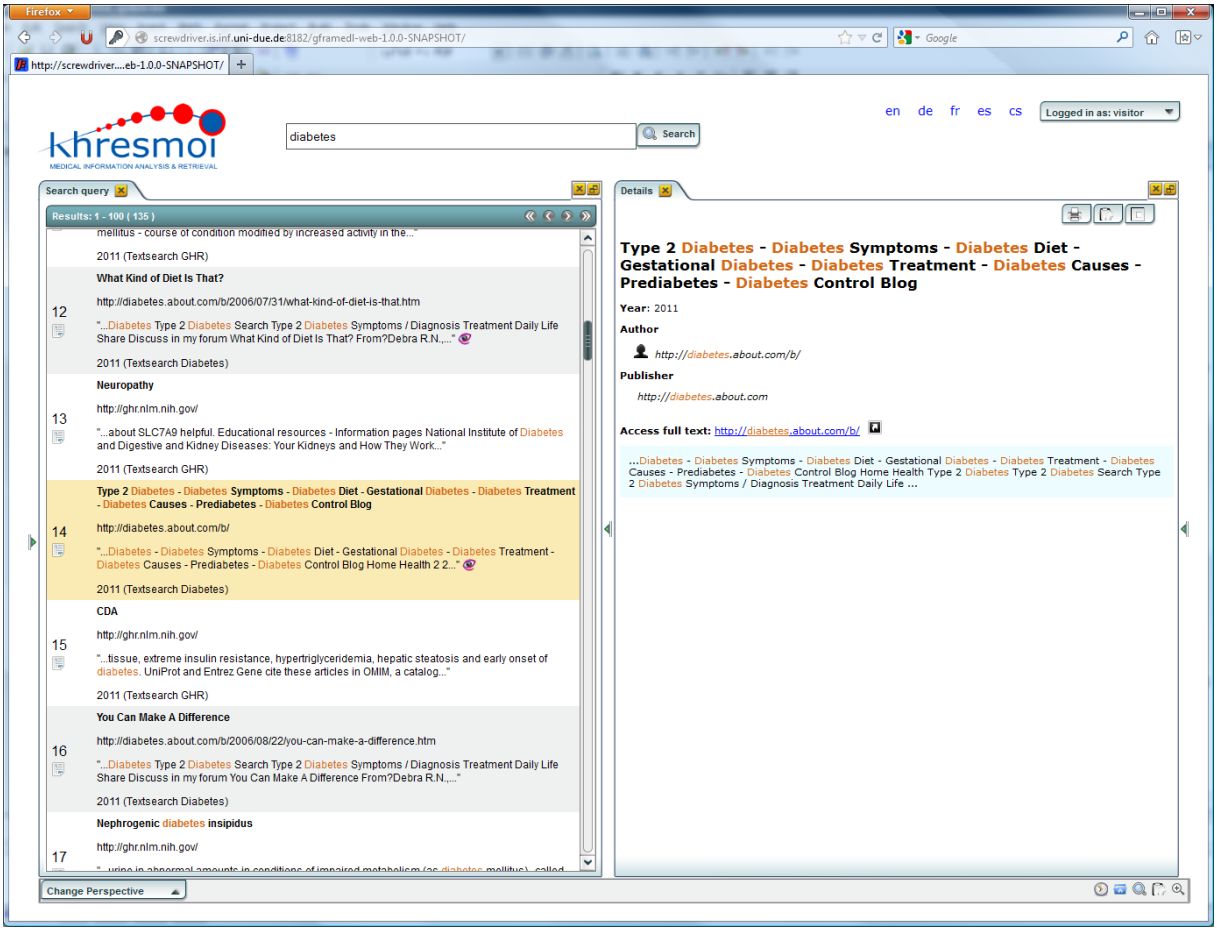

Figure 3: Khresmoi web-based interface

2.2 Prototype access

The Swing interface can be accessed here: http://khresmoi.is.inf.uni-due.de/.

The web interface can be accessed here:
http://screwdriver.is.inf.uni-due.de:8182/gframedl-web-1.0.0-SNAPSHOT/.

2.3 Use case requirement satisfaction

In Deliverable 8.2 [2], Section 5.3, a list of use case requirements for the physician and general public use cases is given. Table 1 describes which of these requirements are satisfied by the current prototype.

Use case	Current state
Resources	The prototype indexes the crawl of HON-certified websites for the general public, and a crawl of physician-targeted resources created within the Khresmoi project. The requirements show that different groups of end users tend to want to access different resources (primary resources, secondary resources, drug information, forums, etc.). Mimir has the capability to associate document categories with each document indexed, so the end users can choose which type of document to view. Some of these categories are assigned manually (by HON staff), but automated document classification into these categories has also been developed.
Classification/Filtering	Due to the ability to associate document categories with documents, it is possible to implement filtering and classification by all criteria listed in the requirements. The following have been implemented: physician/general public resources, type of resource, type of content, country of origin, date range, target general public group and media type. Others will be implemented in the remaining project time. Of particular interest is the development of automated classification algorithms to estimate these categories.
Ranking	Ranking is currently possible by document relevance and date of publication. Others, in particular those that require a social search aspect, will be implemented in the remainder of the project.
Interface layout	Through the use of perspectives, the ezDL interface can be adapted to the requirements of the end users or end user groups. A more comprehensive interface can be used for expert users, as shown in Figure 2, or this can be simplified to for example the standard search engine components of query box, results list and document details display for less experienced users.
Link description	At present the link description in the result list contains the document title, date, type of media, page name, URL, snippet and list of tags applied to the document (from which the target audience can be deduced). Social search ratings will be added once this aspect has been implemented.

Translation tools	Query translation and document translation are available. The remaining requirements are to be implemented in the remainder of the project.
Query completion/support	The following are implemented: suggested query completion/-expansion, search history, spelling correction, and term definition below the search bar. It remains to be evaluated if the integration of patient data is possible within the scope of the project, due to privacy concerns.
Collaborative aspects	Collaborative search aspects are currently under development, and are not yet in the prototype.
Tools	Image search is implemented. The personal library and tray tools meet the requirements of a self-made compendium. Result preview is also implemented. Result sharing is currently being implemented as part of the social search aspects. Both 3D body visualization and accessibility options are planned to be integrated.
Suggested links	Suggested links are not yet implemented, but discussions on how to include them have started. It will most likely be best to use both semantic information and query log information in determing the suggested links.

Table 1: Khresmoi Integration Platform use case requirements satisfaction

3 Khresmoi Classic Search System

3.1 Description

This section describes the framework of the classic prototype components, so that their particular setup and their integration with various interfaces and underlying data sources may be made clear.

The search engine relies on the following pipeline, illustrated in Figure 4:

1. Web crawling
2. Indexing
3. Searching

Pages are retrieved by the web crawler, which is an automated spider that follows every internal link on a given site. The contents of each page are then analyzed to determine how it should be indexed. For example, tokens are extracted from the titles, page content, headings (e.g. `h1`, `h2`, `h3`), or special fields called meta tags. The data extracted from web pages is stored in a NoSQL database (CouchDB) and an index (Solr). Finally, the classic prototype web application (built on Grails) displays the search results to the user by querying the index.

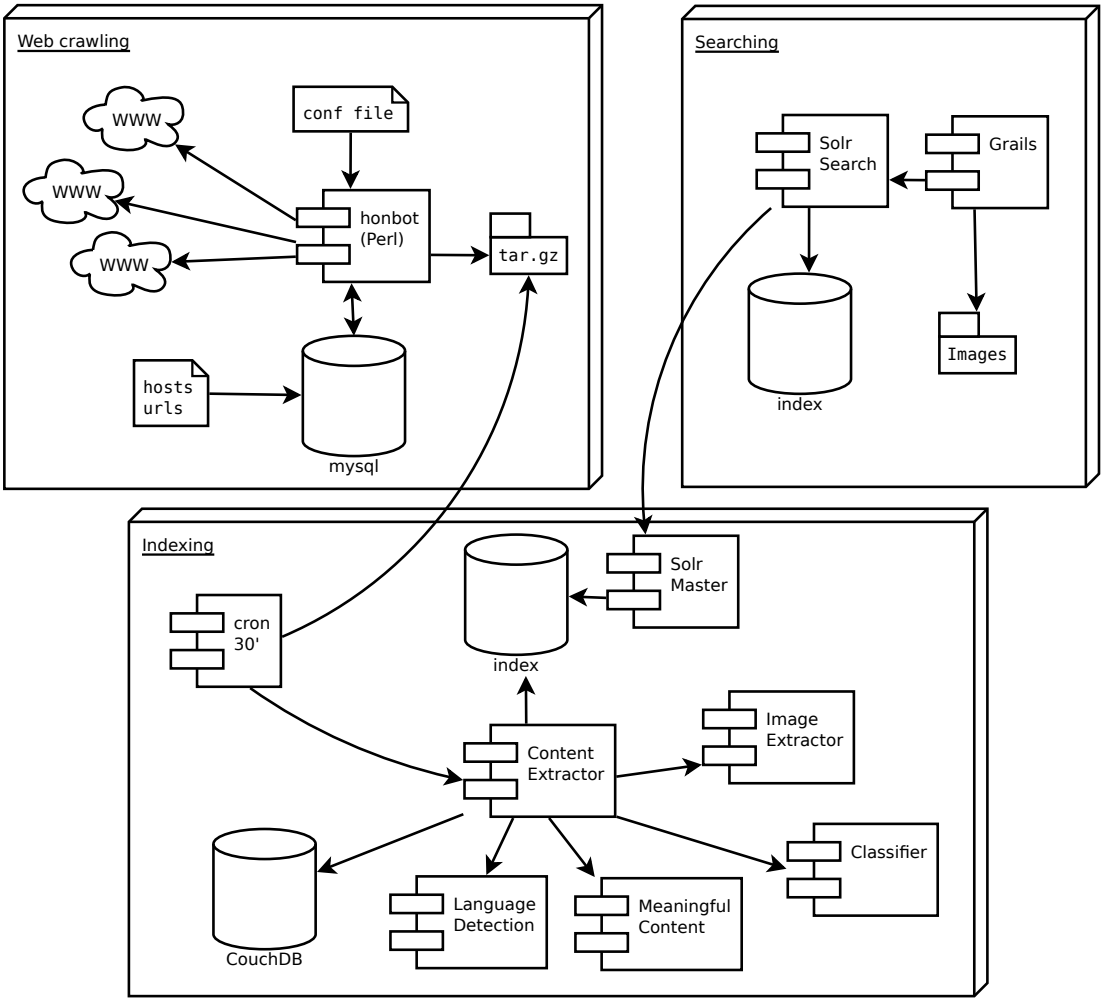

Figure 4: Khresmoi classic prototype overview

3.1.1 Third-Party Libraries

The classic prototype pipeline makes use of several third-party libraries and frameworks, which are detailed below:

Apache CouchDB: CouchDB [5] is an open-source database that focuses on user-friendliness and on being "a database that completely embraces the web." It is a NoSQL key-value store

that uses JSON to store documents, JavaScript as its query language, and REST over HTTP as its API.

Apache Solr/Lucene: Solr/Lucene is the longstanding most popular open-source search platform [3]. Its major features include full-text search based on TF/IDF, hit highlighting, faceted search, dynamic clustering, database integration, rich document (e.g., Word, PDF) handling, and geospatial search. A mature Apache project, it has been in active development for over a decade and powers many widely-used applications, such as Twitter and the Eclipse IDE [6].

Solr/Lucene is written in Java and runs as a standalone search server within a servlet container such as Tomcat [4]. Its REST-like HTTP/XML and JSON APIs may be easily integrated with web applications written in any programming language. Previously, Solr and Lucene were separate Apache projects, with Lucene acting as the pure Java backend and Solr as the servlet frontend [20].

Grails: Grails [26] is an open-source web application framework that focuses on developer productivity by applying modern software engineering principles like "convention over configuration." Grails naturally complements standard Java web application development, since it is built on Spring and based on Groovy, which is a dynamic, JVM-compiled scripting language.

AJAX Solr: AJAX Solr [11] is a JavaScript library for creating user interfaces to Apache Solr. It is a JavaScript framework, but requires an AJAX implementation to communicate with Solr. As such, it may be easily integrated with any major AJAX framework, such as jQuery, MooTools, Prototype, or Dojo.

jQuery and jQuery UI: jQuery [13] is a fast and concise JavaScript library that simplifies HTML document traversal, event handling, animating, and AJAX interactions. jQuery UI [14] provides abstractions for low-level interaction and animation, advanced effects and high-level, themeable widgets. It is built on top of the jQuery JavaScript library.

3.1.2 Crawler

This module implements a configurable web traversal engine, also known as a robot or spider. Given an initial page URL, the crawler will fetch the contents of that page, extract all inward-pointing links on the page, then eventually perform the same operation on those links. Features of the crawler module include:

- Follows the Robot Exclusion Protocol (i.e. robots.txt) [17].
- Supports the meta element proposed extensions to the Protocol [16].
- Implements many of the *Guidelines for Robot Writers* [15].

- Builds on standard Perl 5 modules for WWW, HTTP, HTML, etc.

The web crawler deployed for this prototype focuses on the list of HONcode-certified websites. Crawling began on April 20, 2012, and, as of this writing, the index contains over 4,300,000 pages and over 16,000 images. This content is made freely available to Khresmoi partners in raw[3], CouchDB[4], and Solr [5] formats.

3.1.3 Document content extraction and classification

The Content Extractor package extracts the textual content from a web page and applies other extraction services described below.

Meaningful Content: This package applies textual analysis metrics on HTML documents to extract the most meaningful parts, excluding uninformative text like navigation panes and advertisements. Much of this work was previously discussed in [18], Section 3.3.6.

Relevancy score: The page relevancy score extractor has been implemented with the goal of determining whether or not a given document is relevant from the point of view of a medical- or health-based search engine. Several different benchmarks were tested for this purpose using different information extracted from the documents and different classification algorithms (see [18], Section 3.3.6 for detailed discussion).

Empirical testing has shown that the best results are obtained by using the Naïve Bayes algorithm based on presence of terms from health-oriented documents versus a general corpus. The score calculation retained was not shown to be robust enough to be used as a threshold for keeping or disregarding documents. However, it is used for boosting a document's score at query time.

Keyword Classifier: Pursuant to the requirements described in [24], Section 4.1, the goal of this task was to classify web pages within various health related topics. Based on user requirements HON has established the list of topics with the intention of covering both medical (e.g. *Cancer*, *Cardiovascular diseases*) and more general thematics (e.g. *Alcohol*, *Tobacco*), while also taking into consideration various user groups (e.g. *Women*, *Young people*). The list of topics is given below:

- Alcohol
- Babies and Children
- Biological and chemical risks
- Cancer
- Cardiovascular diseases
- Carers
- Consumer safety
- Drugs

[3] http://khresmoi.honservices.org/~khresmoi/crawled_pages/
[4] http://khresmoi.honservices.org:5984/_utils
[5] http://khresmoi.honservices.org/hon-search/select?q=*:*

- Elderly
- Environmental health
- Food safety
- HIV/AIDS
- Influenza
- Insurance
- Long-term care
- Medicines and treatment
- Men
- Mental health
- Nutrition
- Other infectious diseases
- Other non-comm. diseases
- Patient safety
- People with disabilities
- Physical risks
- Prevention and Promotion
- Rare diseases
- Research
- Road safety
- Sex
- Social environment
- Sports and Leisure
- Tobacco
- Travel
- Vaccinations
- Women
- Young people

We had initially considered using example-based supervised learning as a solution for this task. It has been determined that adapting such tools would not be a good solution for various reasons. First off, such an approach requires a large training/test set of documents, which is not at our disposal. Secondly, even if such a set were to be created, it might make the system efficient for one language (notably English), but would not be applicable for other languages.

Thus, we opted for keyword-based bootstrapping as a method of performing this task. Taking as its source publicly-available glossaries from trusted sources (HONcode-certified web sites) related to the given topic, a list of key phrases per topic was created. The lengths of the keys span from 1 to 5 words. The list of phrases created in this way was than manually examined in order to remove the candidate terms that were too general and thus poor representatives of the given topic. For example, we can take the key phrase *Orphan drugs* as being a part of the glossary for the topic *Rare diseases*, while none of the separate components of this phrase would be considered as a good indicator of this topic.

Classification is then performed based on the density of the glossary's key phrases within the web page content and title. In this process a permuted keyword matching on a larger window was used (e.g. a 5-word text window for 2-word key phrase).

Key phrase "$< kt1 >< kt2 >$"
Matches:
"$[\cdots] < kt1 >$ [w1] [w2] [w3] $< kt2 > [\cdots]$"
or
"$[\cdots] < kt2 >$ [w1] [w2] [w3] $< kt1 > [\cdots]$"

The advantage of this approach is twofold. Firstly, applying the classification to other languages requires only translation of the key phrases. Secondly, adding a new topic would require

only the creation of the key phrase list for it, this being less laborious compared to training/test set creation.

Disease Section Extractor: In [24], Section 4.2, plans were laid out for paragraph-level classification of various health topics. A baseline version to extract *disease sections* is included in this prototype.

Many health web sites have a consistent page structure when describing diseases — e.g. *Symptoms*, *Diagnosis* and *Treatment*. The Disease Section Extractor identifies such well-known "disease sections" across various English-language web sites using rule-based extraction methods. This information can then be used for filtering the search results in the frontend interface, e.g. for a user who is interested in diabetes, but only pages about its symptoms. In the future, such data could also be used as input for supervised learning algorithms, which would open up the possibility of classifying arbitrary text.

Currently the supported web sites are Mayo Clinic[6], WebMD[7], National Health Service[8], Everyday Health[9], UpToDate[10] and PubMed Health[11]. Identified sections are *Overview*, *Symptoms*, *Causes*, *Diagnosis*, and *Treatment*.

Image Extractor: This package extracts images found in HTML pages. Because users of search engines tend to only read the first few pages of results, the extractor focuses on precision over recall, and thus has a very strict set of rule-based criteria for extracting images.

For instance, images must contain an `alt` or `title` tag, they must adhere to certain reasonable size limitations, and they must not be externally linked. This Draconian thresholding tends to exclude low-quality images such as advertisements and icons, as well as images that are not easily searchable using text-only methods.

The XML in Figure 5 shows the structure of an extracted image in Solr. Indexed text data associated with images includes the filename (tokenised based on camel-case and other common filename conventions), `alt` tag, `title` tag, surrounding text within a small window, and surrounding text within a larger window. The larger window is only used based on term vector similarity with the filename, `alt` text and `title` text to avoid including irrelevant descriptions.

Language Detection: The Language Detection service is a simple wrapper around the language-detection [21] Java library. This is an important part of the crawler, because it determines what language we attach to a crawled document, and by extension its extracted images. Currently all official languages of the European Union are supported except Irish and Maltese.

[6] http://www.mayoclinic.com/
[7] http://www.webmd.com/
[8] http://www.nhs.uk
[9] http://www.everydayhealth.com/
[10] http://www.uptodate.com/
[11] http://www.ncbi.nlm.nih.gov/pubmedhealth/

```
<doc>
  <str name="cleanedFilename">Cancer</str>
  <long name="contentLength">130184</long>
  <str name="contentMD5">cd5c50647fe7c3c6a3f8c600916ffaa7</str>
  <date name="date">2011-09-13T20:55:44Z</date>
  <str name="docType">image</str>
  <str name="domain">prostate.net</str>
  <str name="followingText">
    Can Finasteride and Dutasteride Prevent Prostate Cancer?
  </str>
  <int name="height">325</int>
  <str name="id">
    http%3A%2F%2Fwww.prostate.net%2Fwp-content%2Fuploads%2F2011%2F09%2
        FCancer.jpg
  </str>
  <str name="imageType">JPEG</str>
  <str name="language">en</str>
  <str name="parentUrl">
    http://www.prostate.net/2011/bph/can-proscar-and-avodart-prevent-
        prostate-cancer/
  </str>
  <str name="precedingText">Editor</str>
  <str name="site">www.prostate.net</str>
  <str name="title">Dictionary Series - Health: cancer</str>
  <str name="url">
    http://www.prostate.net/wp-content/uploads/2011/09/Cancer.jpg
  </str>
  <int name="width">490</int>
</doc>
```

Figure 5: Extracted image in Solr

3.2 Search Engine

The first prototype of the front end search engine application provides the following features:

1. A list of search results ranked by pertinence to the search query. The basic attributes for each retrieved page are: the page title and a short description of the page (snippet).
2. Paging of search results, with total number of results displayed.
3. Spelling suggestion (“Did you mean . . . ?”).
4. Results filtering.
5. Autocompletion suggestions.

The interface layout is illustrated in Figure 6. Key features of the search engine are described below:

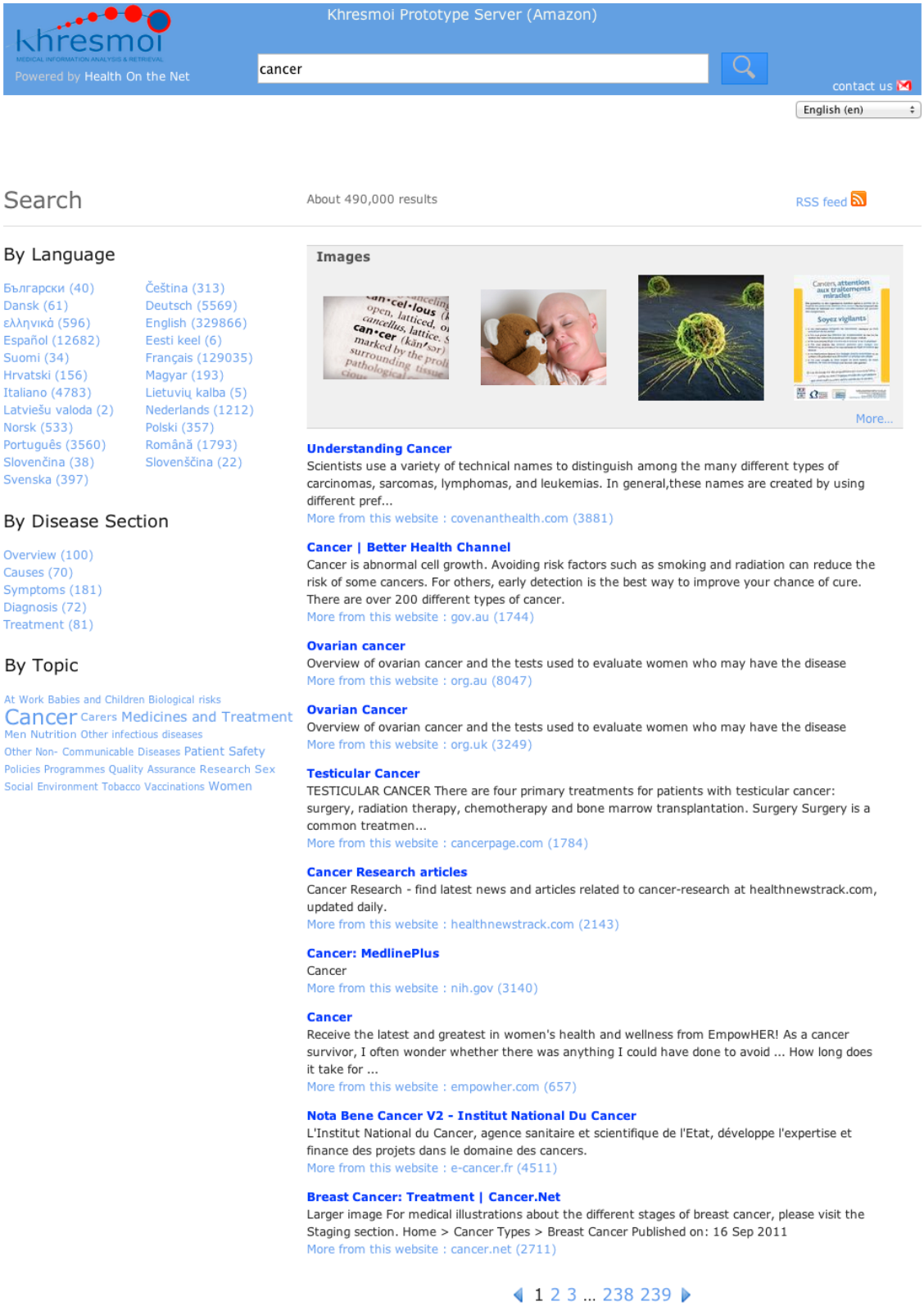

Figure 6: Classic prototype search interface layout

Language filtering: The "By Language" widget allows the user to filter the search results by the language of the page. The language of the interface itself may be changed independently.

Disease section filtering: The "By Disease Section" widget allows the user to filter results by the extracted disease sections. Results in this view show the text from the section itself, and are linked to the relevant anchor tag within the source page, if applicable.

Topic filtering: The "By Topic" widget allows the user to filter based on topics identified by the Keyword Classifier. It is presented in a visually-appealing "tag cloud" format, where the font size indicates the number of documents found under each topic, as shown in Figure 7.

Figure 7: Health topic filtering

Group by site: Results are grouped by domain name. Most commercial search engines, such as Google, collapse on the parent site so that only one or two entries are shown from the same domain. The classic prototype follows suit, also displaying a link that allows expanding the results to show more hits from the same site. In the expanded mode, results are collapsed based on the MD5 checksum of the document, to ensure that duplicates do not pollute the search results.

Query completion: Autocompletion, when added to an input field, enables users to quickly find and select from a pre-populated list of queries as they type. These queries are generated from the document base, so they are guaranteed to return at least one result. This feature is based on the standard Solr Suggester component [8], using token N-grams up to a Markov degree of 1 (i.e. bigrams). It is shown in Figure 8.

Spellchecker: A spelling correction is suggested by the system when the user provides misspelled search terms. This system is based on the standard Solr SpellCheck component [7]. It does not require an explicit vocabulary, but rather leverages the vocabulary coverage of the Lucene document base to generate spelling corrections. It is shown in Figure 9.

Because misspelled words may appear in the document base itself, Solr's `spellcheck.onlyMorePopular` feature is used, whereby spelling corrections are only proposed if they appear significantly more frequently than the input term. For instance, as of this writing, the query `alzheimer` returns 35,426 results, whereas `alzeimer` returns 87 results. This extreme disparity indicates that `alzeimer` is probably misspelled.

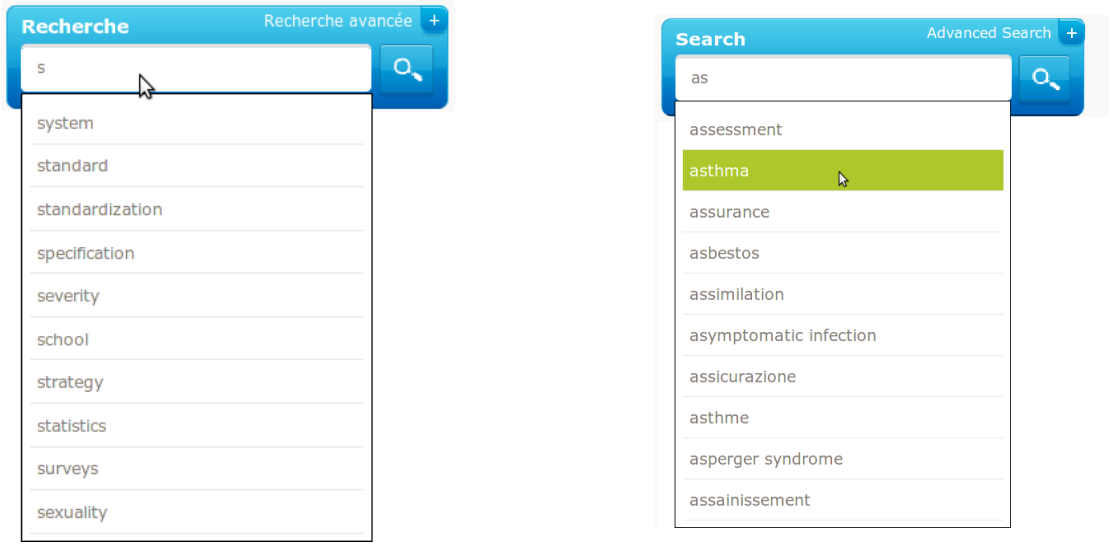

Figure 8: Query completion

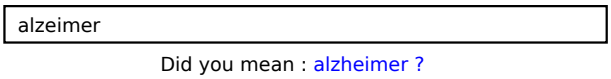

Figure 9: Spellcheck example

Document snapshot: Placing the mouse over the eye icon next to a search result gives a preview of the retrieved page, as shown in Figure 10. This snapshot is not interactive, but it provides a quick indication of what the user can expect to see on the target page.

Images: With each page of search results, if there are images related to the query term, 4 images are displayed (expandable to 12 and independently pageable). Figure 11 shows this feature with the query `cancer`.

3.3 Prototype access

The prototype is available on http://khresmoi.honservices.org/hon-search/

3.4 Use case requirement satisfaction

In [2], Section 5.3, a list of use case requirements for the physician and general public use cases is given. Table 2 describes which of these requirements are satisfied by the current prototype.

Figure 10: Document snapshot

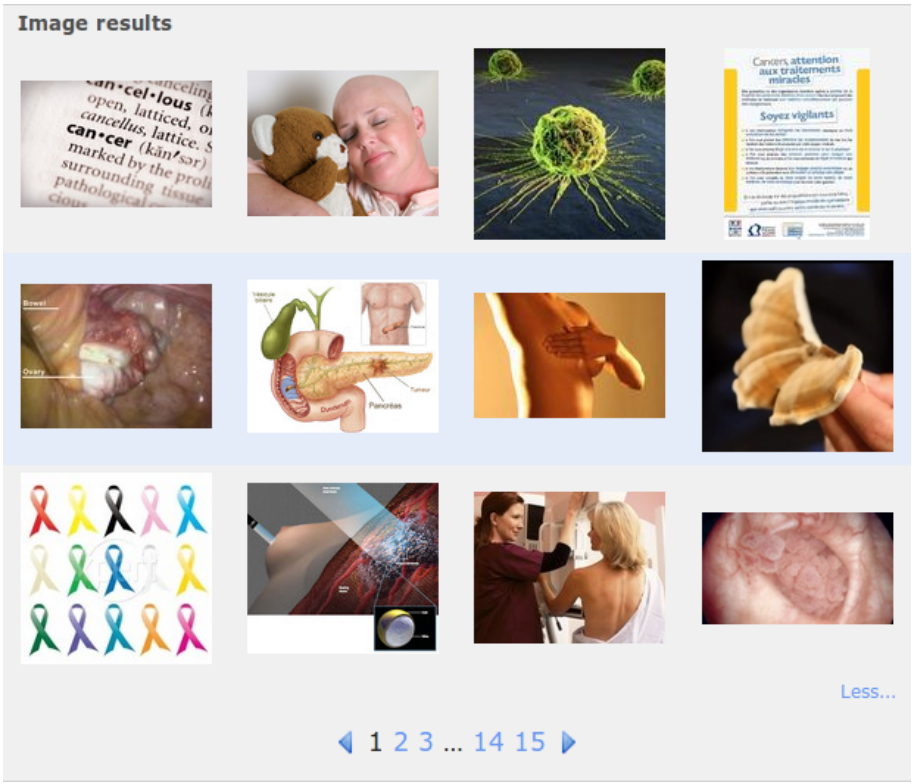

Figure 11: Image search preview

Use case	Current state
Resources	HONcode-certified web pages are indexed, among which there are pages corresponding to hospitals, information on drugs and other health products, news and forums, which primarily reflect the interests of members of the general public. The categories are assigned manually by HON staff. This collection is also indexed by the integration platform.
Classification/Filtering	The following filters have been implemented: language of result, disease section and health topic. Type of resource, type of content, date range, location, and type of audience (by age, gender and health/non-health professional) are filtering features that will be included from the integration platform.
Ranking	Documents are currently ranked by relevance to the input query and general relevance to the medical domain.
Interface layout	The classic prototype is mostly intended for the general public. Based on survey results [23], users tend to ignore advanced search capabilities, hence preference is given to a simple interface. Currently it is possible to change the language of the interface and of the search results, which also changes the topic names.
Link description	At present the link description in the result list contains the document title, snippet, parent domain, and URL.
Translation tools	Translation tools are not implemented, however as HONcode-certified web sites come in various languages, users have the possibility of filtering the results by language.
Query completion/support	Query autocompletion and spelling correction are implemented.
Collaborative aspects	Collaborative search aspects are not available.
Tools	Image search and result previews (snapshots) are both implemented.
Suggested links	Suggested links are not implemented, but it is planned to include them from the integration platform.

Table 2: Khresmoi classic prototype use case requirements satisfaction

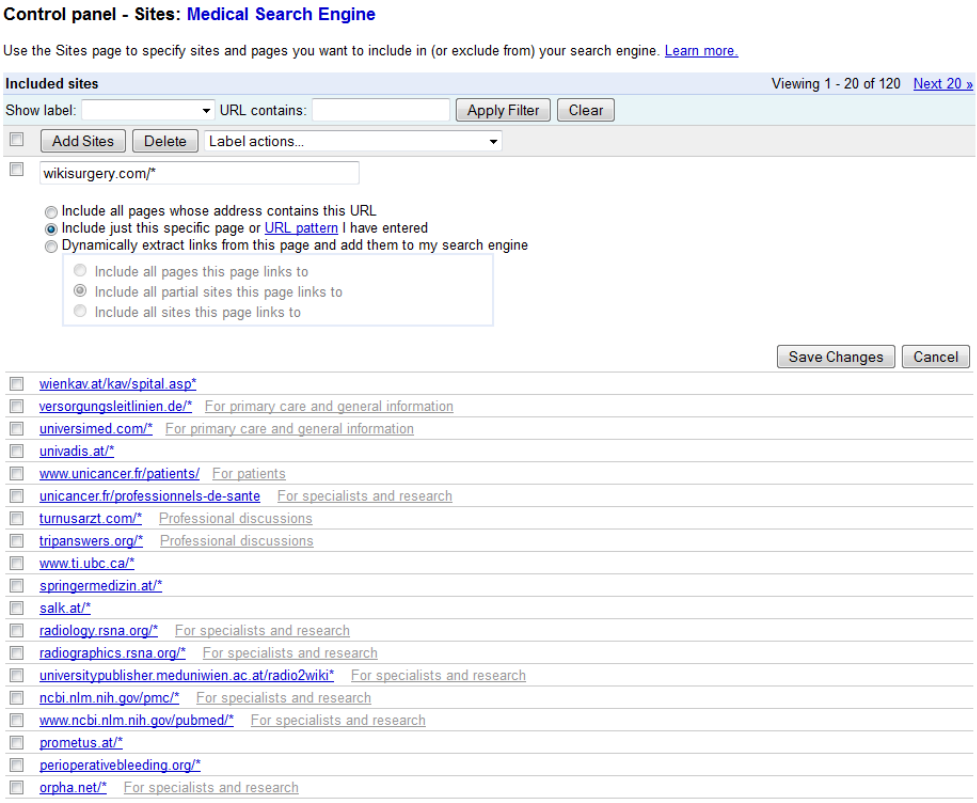

Figure 12: Google sites listing

4 Google Custom Search Baseline System

4.1 Description

Google offers a service called Google Custom Search Engine (CSE), which allows users to define their own, specialised search engines that make use of the Google index and some of Google's technologies. To compare ongoing developments in Khresmoi with the possibilities offered by Google CSE, we created a Google CSE search engine based on the curated list of medical websites compiled by the Khresmoi project (Figure 12). A screenshot of the search interface is shown in Figure 13.

4.2 Prototype access

The Google CSE engine ("Google CSE Baseline") is currently available on the web at `http://samwald.info/medical_search_engine/`.

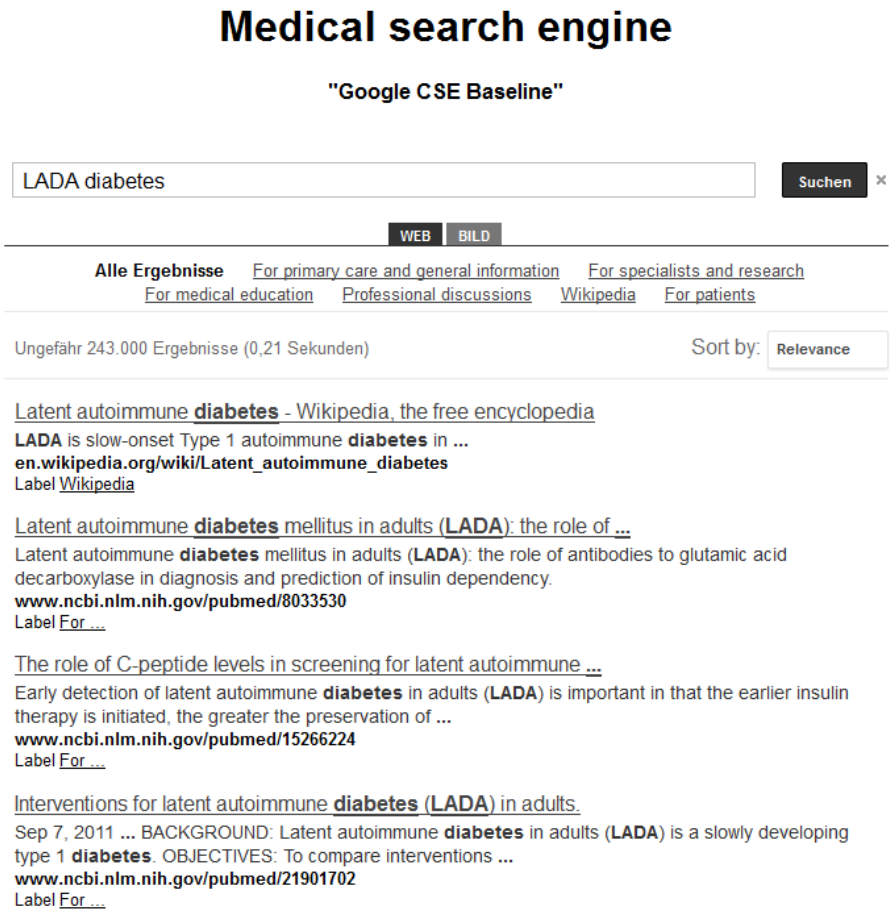

Figure 13: Google CSE overview

4.3 Use case requirement satisfaction

In [2], Section 5.3, a list of use case requirements for the physician and general public use cases is given. Table 3 below describes which of these requirements are satisfied by the current prototype.

Use case	Current state
Resources	Resources can be easily customised. The list of resources of interest to medical practitioners created in Khresmoi was used.

Classification/Filtering	Currently, Google CSE only supports very coarse classification and filtering of search results. We assigned resources to one or more of the following six major categories: “For primary care and general information”, “For specialists and research”, ”For medical education”, “Professional discussions”, “Wikipedia” and “For patients”. The user can select one of these categories by clicking a tab at the top of the search results, in order to narrow down the results. It is possible to configure which effect the selection of a certain category by the user has on search results. There are several possible effects to choose from: (i) Excluding all sites that are not within the category; (ii) Ranking sites within the category higher than sites outside the category (“boosting”); and (iii) Automatically adding words to the search query. Only the first two options were used in the Google CSE Baseline search engine. Importantly, Google CSE (as well as the main Google search engine) does not support any type of faceted browsing, or other advanced means for incrementally filtering search results. This means that the user cannot “tunnel in” to search results based on categories or tags.
Ranking	Ranking is currently possible by relevance and date of publication. Google CSE can also use structured data and embedded semantic markup for sorting, but this was not currently used by the Google CSE baseline search engine. While Google’s success is largely driven by the good relevance ranking of its search results, it was noted that the results in the Google CSE Baseline search engine were not always satisfactory. In some scenarios, Wikipedia was ranked too high, with the first 10 search results all referring to different Wikipedia articles. This was addressed by manually decreasing the rank of Wikipedia in the Google CSE, and by offering a separate “Wikipedia” category.
Link description	At present, the link description in the result list contains the document title, a search result snippet, and the category/categories assigned to the source of the search result. Google CSE allows developers to create custom renderers for search results in Javascript, which can then also generate custom renderings of the semantic markup found in the result documents. This advanced feature was not used in the current version of the Google CSE baseline search engine, as most sources do not currently provide such markup.

Translation tools	Google CSE does not offer translation tools.
Query completion/support	Google CSE allows query autocompletion based on terms automatically extracted from indexed documents, as well as custom dictionaries that can be uploaded by the Google CSE administrator. Spelling mistakes are corrected. Google CSE offers the possibility to define a list of synonyms for automated query expansion. This feature was not used in the current version of the Google CSE baseline search engine.
Collaborative aspects	Collaborative aspects and social networking are not addressed by Google CSE itself. It would be possible to place social networking features (e.g., a blog, links to Google Plus or Facebook) on the site where the Google CSE widget is hosted at. However, it currently does not seem possible to functionally integrate such features with the CSE widget. The main Google search engine interface integrates "social results" from personal contacts on Google Plus, but these features are not available in the Google CSE.
Tools	Image search is implemented. The compilation of self-made compendiums / personal bibliographies is not possible with Google CSE. Result preview is not available in Google CSE.
Suggested links	Suggested links can be realized with the "promotion" feature of Google CSE. For example, this feature makes it possible to specify a regular expression (e.g. `.*trial.*`), and a link that should be displayed whenever a user query matches the regular expression (e.g., a link to the clinical trials database `ClinicalTrials.gov`). This is demonstrated in Figure 14. The generated links are prominently displayed at the top of the search results.

Table 3: Google CSE use case requirements satisfaction

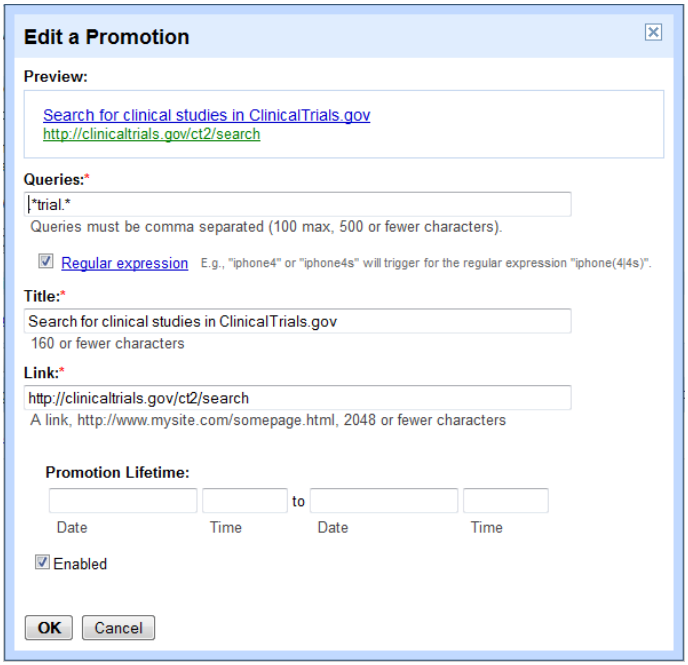

Figure 14: Google promotion

5 Conclusion

In this deliverable, two prototypes are described: the *Khresmoi Integration Platform* and the *Khresmoi Classic Search System*. Furthermore, a baseline *Google Custom Search Engine* (CSE) for medical information is described. It is further explained to what extent the three systems satisfy the user requirements outlined in [2].

The Khresmoi Integration Platform is the scalable platform in which the technology developed in Khresmoi is integrated. As such, it represents the most technologically advanced of the prototypes. The majority of the features requested in the user requirement gathering phase are already implemented, with the main exception being the social and collaborative search aspects. These were identified from the user requirement gathering as an area in which users request support. As this was not initially planned, the development of these aspects started later in the project. Beyond this, approaches for taking advantage of the powerful semantic search capabilities provided by Mimir without requiring the users to create lengthy queries in the Mimir command language will be investigated. New requirements and necessary improvements are expected from the user-centred and global empirical evaluations to be conducted in the Autumn of 2012.

The Khresmoi Classic Search System represents a domain-specific search engine based on open source software as is commonly implemented in industry. This system is an updated version of the search engine that has been developed by HON over the past 10 years to use the latest open source technology. It serves the main purposes of providing some search engine components to the Integration Platform, providing a route for the step-by-step integration of Khresmoi technologies into the HON search engine without requiring a complete change of

search engine, and serving as a model of a potential exploitation path for Khresmoi — the integration of Khresmoi components into existing search engines.

The Google CSE system was created to examine the absolute baseline in currently available customised search solutions. Google CSE can be used to provide a solid, simple search engine for selected medical resources on the web. However, it currently does not offer important features required by Khresmoi, such as multilinguality/translation support, fine-grained filtering of search results and collaborative search aspects. Furthermore, some features, such as the suggested links, are implemented by hijacking tools designed for other purposes. It is currently not clear if and when all required features will become available in Google CSE. The baseline results provided by this search system will be used in the global empirical evaluation. This system also served for initial exploration of the indexed pages for the creation of the user-centred evaluation tasks.

6 References

[1] Célia Boyer, Sarah Cruchet, Angus Roberts, and Jan Dedek. Indexed documents in the biomedical domain. Khresmoi Confidential Deliverable D4.1.1, April 2012.

[2] Célia Boyer, Manfred Gschwandtner, Allan Hanbury, Marlene Kritz, Natalia Pletneva, Matthias Samwald, and Alejandro Vargas. Use case definition including concrete data requirements. Khresmoi Public Deliverable D8.2, February 2012.

[3] Apache Software Foundation. Lucene. http://lucene.apache.org, 2011.

[4] Apache Software Foundation. Apache tomcat. http://tomcat.apache.org/, 2012.

[5] Apache Software Foundation. Couchdb. http://couchdb.apache.org/, 2012.

[6] Apache Software Foundation. Poweredby. http://wiki.apache.org/lucene-java/PoweredBy, 2012.

[7] Apache Software Foundation. Spellcheckcomponent. http://wiki.apache.org/solr/SpellCheckComponent, 2012.

[8] Apache Software Foundation. Suggester - a flexible “autocomplete” component. http://wiki.apache.org/solr/Suggester, 2012.

[9] Lorraine Goeuriot, Gareth Jones, Liadh Kelly, Sascha Kriewel, and Pavel Pecina. Report on and prototype of the translation support. Khresmoi Public Deliverable 3.1, May 2012.

[10] Mark A. Greenwood, Angus Roberts, Niraj Aswani, and Phil Gooch. Initial prototype for semantic annotation of the khresmoi literature. Khresmoi Public Deliverable D1.2, May 2012.

[11] Evolving Web Inc. Ajax solr. https://github.com/evolvingweb/ajax-solr/blob/master/ASL-LICENSE, 2012.

[12] Emmanuel Jamin, Vassil Montchev, and Konstantin Pentchev. State of the art, concepts and specification for the "early software architecture". Khresmoi Public Deliverable D6.1.1, May 2011.

[13] The jQuery Foundation. jquery: The write less, do more, javascript library. http://jquery.com, 2012.

[14] The jQuery Foundation. jquery user interface. http://jqueryui.com/, 2012.

[15] Martijn Koster. Guidelines for robot writers. http://www.robotstxt.org/guidelines.html, 1993.

[16] Martijn Koster. About the robots ⟨meta⟩ tag. http://www.robotstxt.org/meta.html, 2007.

[17] Martijn Koster. The web robots pages. http://www.robotstxt.org/, 2007.

[18] Georg Langs, Andreas Burner, Joachim Ofner, René Donner, Henning Mueller, Adrien Depeursinge, Dimitrios Markonis, Célia Boyer, Alexandre Masselot, and Nolan Lawson. Khresmoi Public Deliverable D2.2, May 2012.

[19] Ivan Martinez, Miguel Angel Tinte, Ana Juan Ferrer, and Francesco D'Andria. Khresmoi Public Deliverable D6.4.1, November 2011.

[20] Mark Miller. Lucene and solr development have merged. http://www.lucidimagination.com/blog/2010/03/26/lucene-and-solr-development-have-merged/, 2010.

[21] Shuyo Nakatani. Language detection library for java. http://code.google.com/p/language-detection/, 2011.

[22] Konstantin Pentchev and Vassil Momtchev. Large scale biomedical knowledge server. Khresmoi Confidential Deliverable D5.2, May 2012.

[23] Natalia Pletneva, Alejandro Vargas, and Célia Boyer. Requirements for the general public health search. Khresmoi Public Deliverable D8.1.1, May 2011.

[24] Angus Roberts, Niraj Aswani, Natalia Pletneva, Célia Boyer, Thomas Heitz, Kalina Bontcheva, and Mark A. Greenwood. Manual annotation guidelines and management protocol. Khresmoi Confidential Deliverable D1.1, February 2012.

[25] Iván Martínez Rodríguez and Miguel Angel Tinte García. Evaluation of the 'early software architecture' and further specification. Khresmoi Public Deliverable D6.3.2, May 2012.

[26] SpringSource. Grails. http://grails.org, 2011.