

# Semantic Search with Omnifind in Wikipedia

Jürgen Umbrich

January 13, 2008

## Executive Summary

### Interactions during and after the Project

In this project we aim to enable a powerful access to the wikipedia knowledge base. We used software from the industry partner IBM (Enterprise document management system OmniFind and the text-analysis engine UIMA) to achieve our goal. This system combined with techniques from the research field of the semantic web allows to query the wikipedia knowledge base in a new and powerful way. With the text analysis engine UIMA we extract information about entities, like persons, locations, events, dates, companies , etc... and indexed the wikipedia articles and the extra extracted information with OmniFind. The query interface of OmniFinds provides a, so called , semantic search functionality. With this search interface we are able to query now for entities and relations between entities. This allows now to ask structured query like "*Give me all scientist born in Stuttgart and are related somehow to the university of Stanford*" or "*Give me all capitals located at the rhine*". Aside that, we developed also an End-User Query Interface to use the power and richness of OmniFind and its query interface in a usable way without knowledge about query syntax.

The outcome of this project is a way how we can apply structured queries to the wikipedia knowledge base. During the project time we established a lot of collaboration with the industry (IBM) and with our research institutes (DERI, AIFB). Furthermore we participated to the open-source UIMA developer group. Here a list of the research benefits of this projects.

### Contribution to UIMA Component Repository <sup>1</sup>

UIMA Component Repository is an open-source community of UIMA developer with the goal to establish a information and resource exchange platform for UIMA users and developers. We contribute to this community be publishing some of our annotators.

### Research collaboration with the main developers of UIMA: IBM Böblingen

During the project we established a knowledge exchange with the development team of IBM UIMA, located in Böblingen, Germany. A very valuable contact was the main developer Thilo Götz. The discussions with him and his advices pushed the development of the annotators and was very helpful to understand OmniFind and background tasks.

---

<sup>1</sup><http://uima.lti.cs.cmu.edu/>

### **On-going Research at AIFB**

In the field of relation extraction and learning the research will be continued at the AIFB institute at the University of Karlsruhe. The supervisor of this project showed a lot of interest for the topic of how to learn new relations and especially relations with a semantic, like persons are connected to cities by "living in".

### **On-going Research at DERI**

During a knowledge exchange with researchers of the Digital Enterprise Research Institute in September we discussed some new approaches for new end-user semantic query interfaces and also about some ongoing research for relation pattern extraction.

### **On-Going Research: Master-Thesis**

One outcome of this project work is that there is a offer for a master-thesis at the DERI Institute at the University of Galway. The topic of this master-thesis is related to the area of the semantic web, till now the exact topic is not yet clarified. Anyways, all the contacts from this project and the learned knowledge can benefit for the master thesis project.

### **Open source-Code: [www.ontoware.org](http://www.ontoware.org)**

All the developed code of this project is available at the ontoware repository of the AIFB. The repository is accessible via svn.

# Contents

<b>1</b>	<b>INTRODUCTION</b>	<b>5</b>
<b>2</b>	<b>ARCHITECTURE</b>	<b>6</b>
2.1	IBM Enterprise Search Application OmniFind . . . . .	6
2.2	UIMA: TEXT ANALYSIS ENGINE . . . . .	7
2.3	LOCAL DUMP OF THE ENGLISH WIKIPEDIA . . . . .	7
<b>3</b>	<b>ANNOTATORS</b>	<b>8</b>
<b>4</b>	<b>STRUCTURED QUERIES</b>	<b>10</b>
4.1	OMNIFIND QUERY-INTERFACE . . . . .	10
4.2	SEMANTIC QUERIES . . . . .	10
<b>5</b>	<b>END-USER QUERY INTERFACE</b>	<b>11</b>
5.1	SIMPLE QUERY INTERFACE . . . . .	11
5.2	ADVANCED QUERY INTERFACE . . . . .	13
<b>6</b>	<b>DISCUSSION</b>	<b>14</b>
<b>7</b>	<b>SUMMARY</b>	<b>15</b>
	<b>List of Figures</b>	<b>16</b>

# 1 INTRODUCTION

Wikipedia<sup>2</sup> is a multilingual, web-based, free content encyclopedia project and one of the biggest open-source library online at the Internet. More than 75.000 contributors have published more than 9.000.000 articles in over 250 languages. The knowledge of the Wikipedia explored and used for a variety of reasons, e.g. for the private use or for research tasks at school or university.

Because, the Wikipedia is an encyclopedia sometimes the users do not know the exact keyword for their search query. But probably they know to which categories their search results are belonging or/and how their search results are related to other categories or to other things. For these type of search queries, searching for a particular category or for relations between search terms the Wikipedia query interface offer only a keyword-based query interface. These kind of interfaces are well known from the big search engines like Google<sup>3</sup> or YahooSearch<sup>4</sup>. While widely used keyword-based search interfaces pose a lot of problems, which can influence significantly the quality of the search results. One of the most common problems is ambiguousness of words. Words can have more than one meaning and without any context information the search engine can not differentiate among them. Furthermore it is not possible to search for relations between words and for their meaning.

We would like to introduce a more specific user scenario to demonstrate the lack of the keyword search and to show the power of semantic search.

*As research for a student paper the student wants to know all scientist which are born in Stuttgart.*

A normal keyword query for the current search engine could be "*all scientists born in Stuttgart*". But the search engine do not know that scientists are person's, Stuttgart is a city and the relation between the person and cities is here "born in". The general way to process this keyword query is to lookup for the occurrence of the single word tokens in the documents. So in the result set all documents, which are containing the word "all", "scientist", "born", "in" and "Stuttgart" and all the relation between the word or their meaning.

The Wikipedia already provides information about the category of an article and information about relations to other article Wikipedia editors can add articles to available categories or create new categories for them and they can also refer to other Wikipedia categories. But these category and link information are mainly for browsing and navigating through the encyclopedia and not for describing the semantics of and between articles.

The goal of the project was to provide a new way of accessing and exploring the knowledge in the Wikipedia. To achieve this goal we extract implicit knowledge in the Wikipedia articles and make them explicit usable. We used IBM's text analysis engine UIMA to develop new annotators, which are extracting parts of the hidden information in the Wikipedia. To store the annotated document and make them searchable we used IBM's enterprise document man-

---

<sup>2</sup><http://www.wikipedia.org/>

<sup>3</sup><http://www.google.com/>

<sup>4</sup><http://search.yahoo.com/>

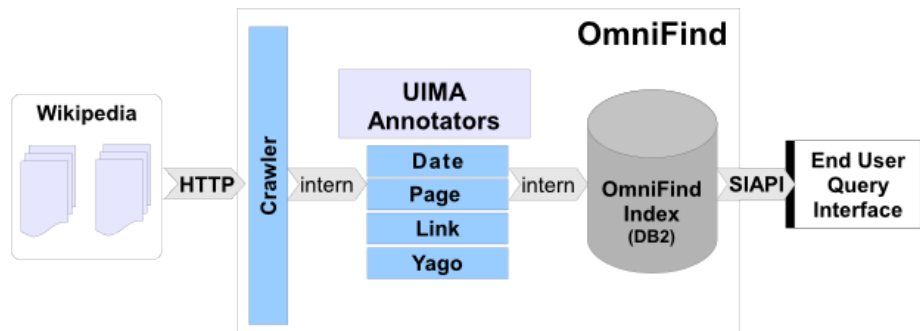


Figure 1: Overview of the architecture

agement system OmniFind, which also provides a semantic query syntax. The input data are for the document management system is served from a local dump of the English Wikipedia. Finally we developed a End-User query interface which is by far easier to use than the IBM OmniFind query syntax.

The further part of this report is structured as follows. Section 2 describes our architecture, followed by an overview of our annotators in Section 3. After learning about structured queries in Section 4 we show in Section 5 an implementation of a user-interface, that makes the query syntax of OmniFind more usable for end-user. Next, in Section 6 we are pointing out some problems we had to deal with during the project. At the end we will give a summarization in Section 7

## 2 ARCHITECTURE

In figure 1 we show an overview of our architecture. The core of our system is the UIMA text analysis engine. To discover the semantic of word and the relation between entities we developed a set of annotators. UIMA and its annotator are embedded in the Enterprise Content Management System OmniFind. We used the OmniFind crawler module to gather the input data from a local SQL data dump of the English wikipedia. To enable access to the index data we implemented a query interface in Java Swing, that use the OmniFind search interface API (SI-API). This end-user query interface is described in detail in section 5.

In the following section we provide more details about the individual components.

### 2.1 IBM Enterprise Search Application OmniFind

IBM OmniFind is the main enterprise search platform of IBM. It comes in several packages adapted to different business needs. An enterprise search system provides extensive capabilities for searching any number of structured and un-

structured data sources with a single query. Fast query response times and a consolidated, ranked result set that is based on extensive text analysis enable you to not just locate documents of interest, but extract meaning from document content.

OmniFind scales up to millions of documents and thousands of users. It includes pre-built integrations for indexing data and content from file shares, databases, collaboration tools, content management systems, blogs, wikis and forums. Furthermore it delivers semantic search for improved findability through native support for the Unstructured Information Management Architecture and it is built on an open architecture for text analytics (UIMA) and semantic search.

## 2.2 UIMA: TEXT ANALYSIS ENGINE

UIMA stands for the Unstructured Information Management Architecture. Unstructured Information Management applications are software systems that analyze large volumes of unstructured information in order to discover knowledge that is relevant to an end user. UIMA is a framework and SDK for developing such applications. An example UIM application might ingest plain text and identify entities, such as persons, places, organizations; or relations, such as works-for or located-at. UIMA enables such an application to be decomposed into components.

It is an open, industrial-strength, scalable and extensible platform for creating, integrating and deploying unstructured information management solutions from combinations of semantic analysis and search components.

Although UIMA originated at IBM, it has now moved on to be an Open Source project which is currently incubating at the Apache Software Foundation<sup>5</sup>.

UIMA's goal is to provide a common foundation for industry and academia to collaborate and accelerate the world-wide development of technologies critical for discovering the vital knowledge present in the fastest growing sources of information today.

IBM has empowered its products and services with UIMA creating a channel for third-party vendors to deploy their text and multi-modal analytics in larger integrated solutions.

## 2.3 LOCAL DUMP OF THE ENGLISH WIKIPEDIA

We chose the data set of the English Wikipedia for our test-corpus. The English Wikipedia contains of around 1.600.000 documents. Furthermore the knowledge is not restricted to a particular domain, the Wikipedia data set contains articles about a various of different domains and topics. Our motivation to use this data set was, that Wikipedia articles provide already a lot of meta-information like the Wikipedia-categories or links to other Wikipedia articles. Overall, the Wikipedia corpus is a great knowledge base and contains a lot of information

---

<sup>5</sup><http://incubator.apache.org/uima>

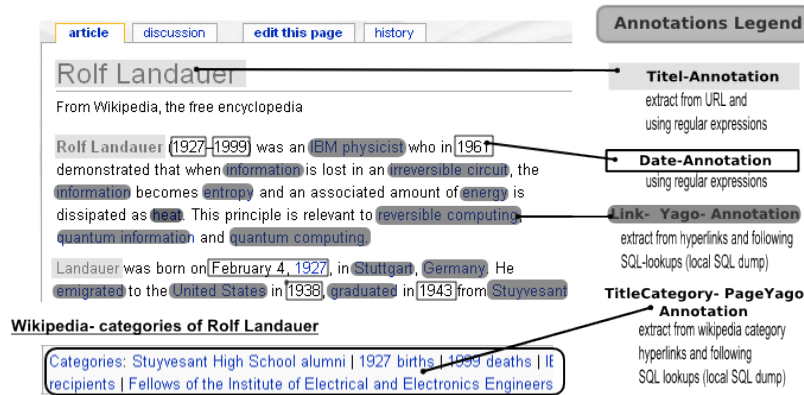


Figure 2: Annotations of a Wikipedia Article

about "real world" objects and their relation between each others, like peoples, countries, cities, religions, companies, etc..., and their relations.

### 3 ANNOTATORS

A annotator can be seen as system that anchors extra information to input sources. For example, language annotators can identify the language of a document, or the meaning of words in a grammatical context(subject, verb, adjective, ..). As described in the previous section, OmniFind integrates UIMA for this annotation tasks. We used a compatible UIMA Developer Version(UIMA SDK 1.4.2) to implement and integrate annotators. This version of UIMA provides, from the scratch, some very useful annotators like a token, sentence and paragraph annotators. Anyways, these annotators are not solving our goals, so we developed some extra annotators. In this section we will give a more detailed overview about our annotators and their functionality. Furthermore, to get an impression of our different annotations and what kind of information we extract from the Wikipedia articles see Figure 2.

- **DATE-ANNOTATOR**

The Date-Annotator is a very simple and basic annotator. We identify year, month and day information from date-format strings in the wikipedia article and annotate the original document by this date information. This annotator enables us to ask queries for particular dates, years or month. As in Figure 2 shown we identify various date-string formats by using a set of regular expressions.

- **PAGETITLE- AND PAGECATEGORY- ANNOTATOR**

The PageTitle- and PageCategory- Annotator extracts the title string and the Wikipedia categories. The page-categories are parsed out of the cat-

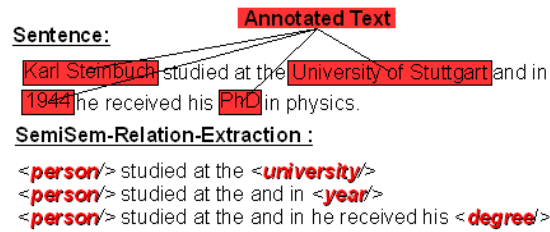


Figure 3: Example: Relation-Extraction

egory Hyperlinks of the site and the page title is extracted out about the page-URL. Furthermore we parse the document for other occurrences of the title string to obtain more information of the page.

- **LINKTITLE AND LINKCATEGORY ANNOTATOR**

Most of the Wikipediaauthors are interlinking their article with other Wikipedia pages. We use these hyperlinks and annotate them with the Wikipedia-categories of their target pages. To get the categories of one of these link we connect to the local Wikipedia data base and execute a SQL-Query.

- **YAGOCCLASS ANNOTATOR**

The YagoClass-Annotator use the data from the YAGO ontology<sup>6</sup> to annotate Wikipedia-categories by more abstract categories. This ontology is a huge semantic knowledge base, containing the unification of Wikipdia and Wordnet<sup>7</sup> and knows around 14 million facts about entities(e.g. person, city, organization). For each wikipdia category the data set includes a hierarchy of abstract categories, e.g. for the `american_tennis_player` Wikipedia category we can find a yago category hierarchy `player < person < causal_agent`. The task of the YagoClass-Annotator is to use Link-and PageCategory-Annotations to identify the Yago-category-hierachy for each title and link.

- **RELATION-EXTRACTION ANNOTATOR**

The Relation-Extraction annotator is using annotations from he previous annotators to extract possible relations between objects. The core idea is to use sentence annotations from the UIMA framework to discover relations between YagoCategory annotations. These YagoCategory annotation can be seen as objects, e.g persons, cities, events, locations etc.... We assume that they must be somehow connected or related when minimum two of these objects are in one sentence. In Figure 3 we show an example sentence and the extracted semi-semantic relation patterns.

<sup>6</sup><http://www.mpi-inf.mpg.de/~suchanek/downloads/yago/>

<sup>7</sup><http://wordnet.princeton.edu/>

For humans it should be easy to identify entities and their relations between them, (e.g. a *PERSON* is related to a *UNIVERSITY*). The outcome of the Relation-Extraction annotator is a new knowledge base containing objects and possible information about how they could be connected. In further research these semi-semantic relation patterns can be transformed into semantic relation patterns, through identifying the relation between two objects, e.g. *entity:[PERSON] relation:STUDIED\_AT entity:[UNIVERSITY]*.

## 4 STRUCTURED QUERIES

### 4.1 OMNIFIND QUERY-INTERFACE

OmniFind's search interface supports the same standard query operators as most common search engines, like free text and word phrase search as well as operators like AND-, OR-, NOT- and WILDCARD operators. In addition, it is possible to ask semantic queries, with the OmniFind semantic query syntax as described in the following.

### 4.2 SEMANTIC QUERIES

Using the semantic query syntax of OmniFind allows the user a wide variety of query functionalities<sup>8</sup>. After specifying the mapping of the annotations to the OmniFind index, the semantic query syntax allows to ask also for annotations. These enables the user the possibility to structure their queries with the extra information from the annotators. This enables search for more specified concepts, like searching for person's names. Furthermore we can search for relationships between object like "*the person and the city must occur in the same sentence*" or more specific like "*a persons lives in the city versus a person died in a city*". The semantic search syntax provides support for two new kinds of search syntax, XML fragments and a subset of XPath. The search queries can also contain both keywords and semantic search terms.

Our sample query translated in the OmniFind semantic query syntax with XML fragments would look like this

Query1:

```
@xmlf2::'<pageyago><@category>physicist</@category></pageyago>
<title><@name>Albert</@name></title>'
```

Query3:

```
@xmlf2::'<pageyago><@category>person</@category></pageyago>
<p>
<title></title>
"born in"
<yago>
<@category>
```

---

<sup>8</sup><http://www.ibm.com/developerworks/db2/library/techarticle/dm-0508lang/>

```
    city
    </@category>
    Stuttgart
  </yago>
</p>'
```

## 5 END-USER QUERY INTERFACE

To make it easier for end-users to use the search possibilities of OmniFind we developed an end-user query interface. The whole query interface is a stand-alone software, written in JAVA 1.5. We adapted this interface especially to our annotations. The main focus of the End-User Query Interface was to create a query interface, which enables to the end-user an easy way to use the semantic query syntax of OmniFind. The standard query interface, provided by OmniFind, can only process semantic queries when they are described with the OmniFind query syntax. As we showed a example of this query syntax in the previous section, for end-users this query syntax seems to be not easy to understand and to use. Our query interface offers a module to create such semantic queries in a visual way. Because it is a innovative way of searching, we assume that a lot of the end-users have problem to get familiar with our query interface. Therefor we created also a very simple interface, that is similar to the current query interfaces like the Ebay<sup>9</sup> or Amazon://www.amazon.com/query interface. Because this simple query interface is limited to create complex structured queries we provide an advanced query interface which allows to use the full functionality of the OmniFind semantic search.

### 5.1 SIMPLE QUERY INTERFACE

A screenshot of the simple user interface and how to answer our first sample query can be seen in Figure 4. People can use the keyword search field for very simple keyword queries, optional with operators like AND, OR, NOT, WILDCARDS and word phrases. In the PageTitle query field people can search exactly for page titles or word snippets in page title. Using the page category query field, people can filter out only pages of a special page category. Below this pagecategory-query-field is the ResultPagesContaining query field. In this query field people can restrict the search result to special outgoing links or can search for special categories in the linked pages. If people only use the keyword search query field they can use the same functionalities like Wikipedia provides in their own query interface.

With the simple query interface we are already able to solve our usecase query as you can seen in in Figure 4.

---

<sup>9</sup><http://www.ebay.com/>

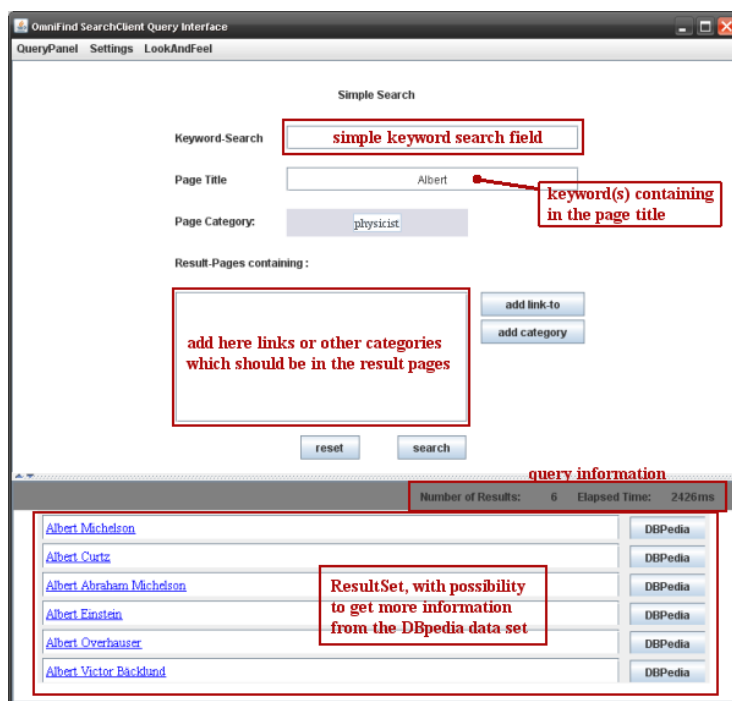


Figure 4: Simple Search Interface

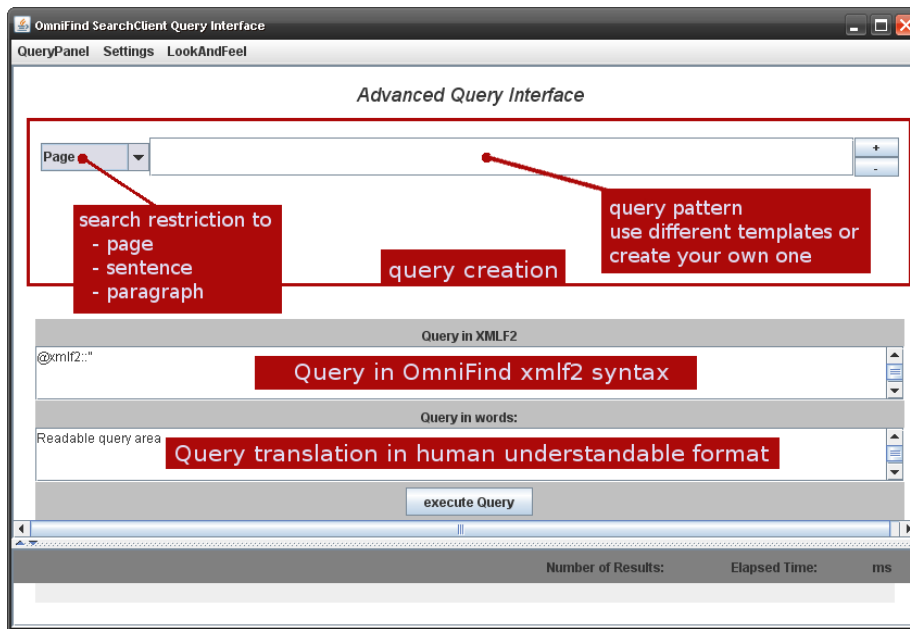


Figure 5: Advanced Query Interface

## 5.2 ADVANCED QUERY INTERFACE

For users, which are more familiar with the query interface we provide also a more advanced query interface. The query interface offers functionalities to create complex structured queries. In Figure 5 you can see the components of the advanced query interface. The core component is the query creation module. A query in this interface is a set of query patterns. For each pattern the user can restrict the search focus to one of three modes, (1) no restriction at all, lookup on the whole page text, (2) paragraph restriction, lookup for the query pattern will be only executed on the paragraphs of the pages, and (3) sentence restriction, search only for sentence containing the query pattern. A query pattern can contain

- keywords, word phrases and keyword query operators
- queries for links and their Wikipedia category
- queries for more YAGO categories
- queries for page titles and page categories( as Wikipedia categories and YAGO categories)

Furthermore, users can also see the translation of the OmniFind query syntax into a more human readable representation.

## 6 DISCUSSION

During the project time we had to deal with a lot of problems. Most of them were caused from the software and our architecture we used.

### Installation of OmniFind

In the very beginning of the project we installed OmniFind and UIMA on one of our servers. The installation of OmniFind was a challenge on its own. After nearly one month we were had a clean installation of all OmniFind components and we were able to administrate the software of IBM. Hardware and software incompatibilities lead to this time delay

### Deploying and Debugging of Annotators

One of our major problem occurred during the phase of developing the annotators. it has been pointed out to us the the documents, provided from the OmniFind developers, were for the development of the annotators unhelpful.

### Limitation of SI-API: The OmniFind Query API

OmniFind provides a query API (SI-API) for accessing the OmniFind Search Server from other computers and networks. The problem we had to deal with this API was, that:

- the use of the API was not well documented
- example code snippets were out of date
- processing the query results is limited, by ten results for each query

*MAYBE NOT TRUE- have to look for streaming results!! -> nullpointerexception of results !?!?* For the End-User Query Interface the limitation to ten result for each query is a problem. Sometime the end users want to navigate through more than only ten results.

### Unstableness of the whole Setup

As mentioned, we had big problems to install OmniFind on one of our servers. During the project we figured out another problem which was related to our server and the installation. As promised from IBM, OmniFind should be a scalable application and should be able to handle some hundreds user and millions of documents. But when we stressed the server with crawling, annotating and searching tasked, the server crashed from time to time. We were not able to figure out what leads to this system crashes, if it was our hardware setting or the architecture of OmniFind and our plugged-in annotators. **/\* SEB: Das ist ein Hardwareproblem. Die Leros stürzt auch ohne Omnifind ab. \*/**

## 7 SUMMARY

In this project we showed a way how applications from industry partners and information extraction and semantic web technologies can significantly improve the search functionalities. In detail, we used a symbiosis between a commercial application(OmniFind) and a powerful open-source software(UIMA) to enable a new way to search in Wikipedia. We transformed parts of the implicit knowledge in the Wikipedia articles into explicit knowledge, that users can now directly use this extracted knowledge in their queries. During the project time we got a lot of help from our collaboration with the research institute DERI, Galway and AIFB, Karlsruhe. Furthermore we established a knowledge exchange with the main developers of UIMA, IBM, Böblingen. Furthermore we plan to continue parts of the challenges in this project with both research institutes. Especially the learned knowledge about the development and the use of UIMA annotators and the challenges we had to deal with the relation pattern extraction is a ongoing topic in the semantic web research field.

## List of Figures

1	Overview of the architecture . . . . .	6
2	Annotations of a Wikipedia Article . . . . .	8
3	Example: Relation-Extraction . . . . .	9
4	Simple Search Interface . . . . .	12
5	Advanced Query Interface . . . . .	13