Semantic Enrichment of Social Media Resources for Adaptation

Oliver Schimratzki, Fedor Bakalov, Adrian Knoth, and Birgitta König-Ries

Friedrich Schiller University of Jena, Germany
oliver.schimratzki@gmx.de,
{fedor.bakalov | adrian.knoth | birgitta.koenig-ries}@uni-jena.de

Abstract. With more and more dynamic content available on the web, we need systems that aggregate and filter information from different sources to provide us with only the information we are really interested in. In this paper, we present one such system, the CompleXys portal, aimed at users interested in complexity or subtopics thereof. It accesses a large variety of different information sources, among them calendars, news sites and blogs, semantically annotates and categorizes the retrieved content and displays only relevant content to the user.

1 Introduction

The amount of dynamic content available on the web is rapidly growing. It becomes more and more difficult for users to keep track of all relevant information - in particular since it becomes more and more overwhelming to manually separate relevant from irrelevant content. Even if a user has identified a variety of news sites and blogs that often contain information she is interested in, those sites will also contain lots of information the user is *not* interested in. Purely syntactic filtering based, e.g., on keywords, as offered by today's tools, offers only a partial solution. What is really needed is semantic filtering, i.e., filtering based on some "understanding" of the content. This will allow for higher precision, i.e., fewer irrelevant articles displayed, and higher recall, i.e. less relevant articles deleted, and will thus increase user confidence in using the tools.

In this paper, we present the CompleXys portal, an information site that will provide users with personalized access to information related to the topic of complexity. CompleXys harvests information from a large variety of sites, ranging from event calendars to blogs and news sites. It semantically annotates the retrieved content. These annotations are then used to categorize the retrieved items and to decide whether they are sufficiently related to complexity or should be discarded. In the future, CompleXys will use the categorization for a more fine-grained personalization, displaying the most relevant items most prominently and providing recommendations to the user.

In the remainder of this paper, after a brief discussion of related work in Section 2 we take a closer look at CompleXys and the underlying technologies: Section 3 provides an overview of the CompleXys architecture. We will then focus

on the most interesting part of this architecture, namely the semantic content annotator which will be presented in Section 4. Finally, Section 5 contains a summary and an outlook on our future work.

2 Related Work

In this paper we describe an architectural solution and an approach to providing a personalized access to the variety of resources residing on the Web and in intranets. To achieve this, we combine the approaches and technology from three areas of research, namely content aggregation, semantic content annotation, and content-based recommender systems.

Content aggregation, though a relatively new field, has already achieved the state of maturity. There exists a number of industry standards and commercial applications of content aggregators. Really Simple Syndication (RSS)[1] and Atom[2] formats have been successfully used by a large number of Web and desktop application for aggregating various types of content, including but not limited to calendar information, news, blog entries, and podcasts. The iCalendar[3] format is used by many applications for aggregating appointments and events from multiple calendar systems. Personal Web portals like *i*Google[4] and My Yahoo![5] allow their users to place different types of content harvested through RSS and Atom feeds on their personal pages. Portals like Technorati[6] aggregate information on more or less specific topics. RSS filtering tools like Feed Rinse[7] allow the user to define keyword based filters on RSS feeds to get rid of irrelevant items. These tools work, however, on a purely syntactic level.

The field of **semantic content annotation** mainly deals with the challenges related to availability of well-formed metadata for the unstructured text resources, which is essential for achieving high recall and precision of information retrieval. A number of approaches to semantic content annotation have been reported in the literature. For example, GATE [4] provides a framework and graphical development environment for implementing natural language processing (NLP) tasks. The framework empowers developers to implement such components as tokenizers, sentence splitters, part-of-speech taggers, gazetteers, semantic taggers, and the components for identifying relationships among the entities in the text. A number of NLP systems leverage GATE and its components for semantic tagging of content; these include but not limited to the KIM platform [8], MUSE [7], and Ont-O-Mat [6].

Availability of machine-processable metadata of content is one of the most essential requirements for the **content-based recommender systems**. These

[1] http://web.resource.org/rss/1.0/
[2] http://tools.ietf.org/html/rfc4287
[3] http://tools.ietf.org/html/rfc5545
[4] http://www.google.com/ig
[5] http://my.yahoo.com
[6] http://technorati.com/
[7] http://http://www.feedrinse.com/

systems recommend relevant content to the user based on the semantic description of available resources and the user's personal preferences. The relevant content is selected by analyzing the content metadata and the user's profile and identifying the items that match the user's individual interests. A number of systems leveraging this approach have been proposed. CHIP [10], for instance, is capable of recommending the user artworks from multiple museum collections. For recommendation, the system leverages the semantic description of artworks and the user's personal interests in the domain of cultural heritage, which the system identifies based on the user's explicit ratings of artworks and semantic relations among the art topics. Other examples of the systems leveraging similar recommendation approach are the Personal Reader Framework [3] and Personal Learning Assistant [5].

3 Overall Architecture

The CompleXys portal aggregates a multitude of different sources from the Internet, categorises the retrieved content, applies semantic annotation and finally presents the filtered and personalised results to the user. Table 1 shows a schematic overview of CompleXys' architecture and its data flow in a left to right manner, which basically implements the Input-Processing-Output model.

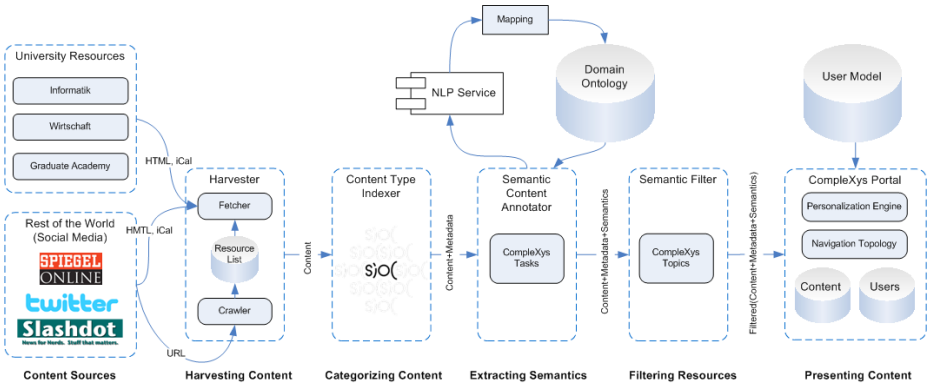

Fig. 1. Overview of the CompleXys portal. Resources are fetched and stored in crawler database, then semantically annotated and finally presented to the user, if they match his personal preferences and interests.

On the input site, the harvester retrieves arbitrary content and stores a mangled version in the crawler database. Since the particular source of each entry is known, this step also provides content type indexing for free.

The crawler database is fully generic and hence supports any kind of input source. Figure 1 shows the underlying schema. The DBMS will increment the unique key *id* for every newly retrieved entry. All later processing steps make

use of this key: querying the crawler database for new content simply means querying for all *id*s higher than the last known or processed *id*.[8]

Column	Type	Modifiers
id	bigint	UNIQUE
source	character varying(255)	
content	text	
internal_id	character varying(255)	UNIQUE not null

Table 1. Database layout of the crawler database. This database contains already fetched items from a potentially large variety of sources. *id* is supposed to be monotonically increasing, while *internal_id* holds a suitable hashsum (e.g. MD5) of the cached resource. The *source* field specifies the origin of the item stored in *content*.

The content itself is blindly stored as text (BLOB), semantic parsing is delayed to subsequent stages in the processing pipeline. The *source* column contains the SIOC content type, it serves as a type indicator to the processing modules.

The crawler is idempotent, that is, it can be run several times without storing already known content again. This property is achieved by *internal_id*, another column set to be a unique key. For each retrieved entry, the crawler calculates a suitable MD5 hash and stores both, content and its derived hash into the database. If this entry has been already fetched, the DBMS will prevent inserting a duplicate MD5 hash into *internal_id* and consequently avoid storing known content again. Obviously, finding an appropriate way for calculating the MD5 hash is crucial. CompleXys currently has built-in support for two different source types: it can directly retrieve calendar events from SQL databases and arbitrary HTML input from the web via RSS, but more resource types can be added via crawler plugins. Generating a suitable hash representing the content usually differs among sources and is hence individually implemented in each such crawler plugin.

The SQL crawler connects to specified source databases in the University's network and harvests information about upcoming events, potentially of interest to the user. Since CompleXys strictly adhere to UTF-8 character encoding throughout the whole processing, the crawler is responsible to convert any source specific encoding, e.g., from Latin1 to UTF-8. This way, subsequent processing modules do not have to take care for different character encodings.

The retrieved SQL calendar events are normalised into a standardised template as shown in Figure 2.

The crawler finally calculates the appropriate MD5 hash for this event by concatenating the source prefix (a constant arbitrary string), the event's primary key in the foreign database and the provided last-update timestamp. This way, the MD5 hash of the concatenation is different for each event from every

[8] SELECT * FROM crawler WHERE id > already_seen

```
def fill_template(params)
    "<sioc:Item rdf:about=\"#{params[:source]}-#{params[:their_id]}\">\n" +
    "\t<vevent:dtstart>#{params[:date]}</vevent:dtstart>\n" +
    "\t<dcterms:creator>#{params[:speaker]}</dcterms:creator>\n" +
    "\t<vevent:location>#{params[:affi]}</vevent:location>\n" +
    "\t<dcterms:title>#{params[:title]}</dcterms:title>\n" +
    "\t<dcterms:abstract><![CDATA[#{params[:abstract]}]]>\n\t</dcterms:abstract>" +
    "\t<vevent:url>#{params[:url]}</vevent:url>" +
    "\t<vevent:dtend>#{params[:endtime]}</vevent:dtend >\n" +
    "\t<dcterms:modified>#{params[:lastupdate]}</dcterms:modified>\n" +
    "</sioc:Item>"
end
```

Fig. 2. Standardised ruby template for calendar events. All occurrences of *params* are substituted by values retrieved from a SQL based event management system.

source database. Even more, updates to already retrieved events have a different timestamp, and consequently, a new MD5 hash together with this updated content will end up in the crawler database. Whenever the CompleXys portal encounters multiple entries for the same source URI in its crawler database, younger rows are updates to already known events.

In addition to SQL calendar events, the harvester has a HTTP crawler for arbitrary HTML content. URLs are extracted from RSS feeds specified in a static configuration file (see Figure 3).

Whenever possible, the crawler tries to use the print version of a document to remove navigation menus, advertisements and other unrelated noise. If the source already provides a more structured representation, e.g., iCal format, it will be used instead. Likewise the SQL crawler, the HTTP crawler wraps retrieved content into a SIOC[9] schema as depicted in Figure 4, generates a suitable MD5 hash and tries to store the result in the crawler database. Again, this insert will fail if the content is already known.

At this stage, all entries in the crawler database are simply unstructured raw text. Unless already provided by the source, there is no semantic information available, yet. However, semantic annotation is required to decide if a given content item is of interest to the user. The next section will explain in detail how this is done.

Once semantic annotation has been provided, relevant items are displayed to the user of the CompleXys portal categorized in appropriate domains. We are currently working on integrating our approach to personalization into CompleXys. This will allow to adapt the information provided to individual user needs: Only information relevant to a specific user (and not to complexity in general) will be provided, the most important information will be displayed most prominently, related information (and possibly related users) will be recommended etc. Underlying this adaptation is a user interest model realized as

[9] http://sioc-project.org/ontology

```
<?xml version="1.0" encoding="UTF-8"?>
<rss-channels>
   <channel>
      <url>http://scienceblogs.com/sample/technology.xml</url>
      <source>scienceblogs.com</source>
      <type>blogs</type>
      <name>ScienceBlogs - Technology</name>
      <category>Technology</category>
   </channel>
   <channel>
      <url>http://scienceblogs.com/sample/medicine.xml</url>
      <source>scienceblogs.com</source>
      <type>blogs</type>
      <name>ScienceBlogs - Medicine</name>
      <category>Medicine</category>
   </channel>
       <channel>
       <url>http://www.wired.com/wiredscience/feed/</url>
       <source>wired.com</source>
       <type>blogs</type>
       <name>Wired Science</name>
       <category>Science</category>
   </channel>
</rss-channels>
```

Fig. 3. Example settings file for CompleXys' HTTP crawler.

```
public String wrapNewsItem(NewsItem newsItem){
  String wrappedNewsItem =
    "<sioc:Post rdf:about=\"" + newsItem.link + "\">\n" +
       "\t<dcterms:title>" + newsItem.title + "</dcterms:title>\n" +
       "\t<dcterms:created>" + newsItem.pubDate + "</dcterms:created>\n" +
       "\t<sioc:topic rdfs:label=\"" + newsItem.category + "\"/>\n" +
       "\t<sioc:content>\n" + "<![CDATA[" + newsItem.content +
       "]]>\n\t</content>\n" +
    "</sioc:Post>";
  return wrappedNewsItem;
  }
```

Fig. 4. Wrapper code for encapsulating newsfeed items into SIOC and DublinCore.

an overlay over the domain model, that collects user interests based on the interactions of the user with the system and also allows the user manual adaptations [1, 2].

4 Semantic Content Annotators

The Semantic Content Annotators pursue the purpose of extracting semantic data from incoming text documents and of annotating this data back to the resources. Furthermore, they are meant to decide whether a given resource is relevant for the topic of complexity and to categorize it by means of corresponding topical concepts.

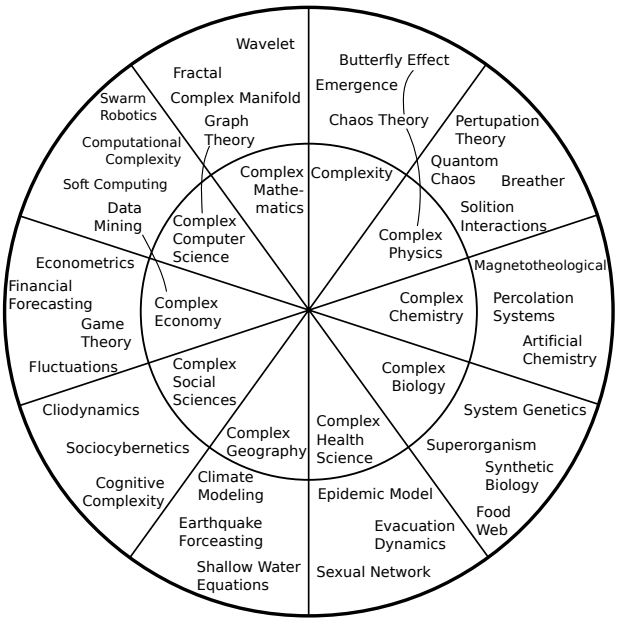

Fig. 5. The CompleXys taxonomy

Both, the annotation and the categorization tasks rely on an ontology, that represents the domain knowledge space of complexity. It is implemented as a SKOS[10] taxonomy and shallowly organized within two hierarchical levels - ten main categories and 297 appendant terms. Furthermore some terms are interconnected by the relation type *related*, to express either topical closeness between two terms or an ambiguity of belonging, when a term could be assigned to more than one main category. Figure 5 shows an excerpt of the model as taxonomy

[10] http://www.w3.org/TR/skos-reference/

circle[11]. The main categories are displayed in the inner circle, while the outer circle contains examples of their appendant terms. The connections between some of the terms are exemplarily for the use of the *related* relationships.

Figure 6 visualizes the architectural composition of the Semantic Content Annotator. It is structured as a parallel working pipeline, utilizing the standard java concurrency package[12] for its implementation. The current pipeline consists of five components, which are called CompleXys Tasks. The Crawled Content Reader and the Content Writer take care of an internally valid data structure and of persistency tasks. In contrast, the inner Complexys Tasks are the actual processing units. They analyze the resources, extract the semantic data and finally annotate it back to the text. The analysis is based on existing NLP services from various contexts. These services are called by using intermediate GATE modules, so that the Complexys Tasks need not care about the technical details of the annotation, but just have to adjust the modules according to their needs and evaluate the solutions.

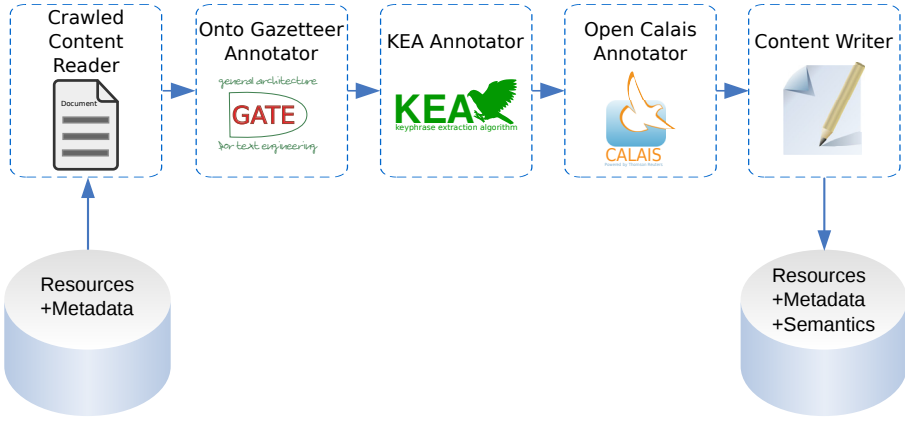

Fig. 6. The Semantic Content Annotators

The Crawled Content Reader is the first component of the pipeline and its main purpose is to gather the documents from the input data store and to prepare them for the succeeding tasks. It wraps the new resources into the internally used GATE data format, embeds them into the corresponding persistency layer and sends them into an output queue for further processing in the pipeline.

The Onto Gazetteer Annotator searches the text for keywords, that are listed in the gazetteer files and annotates found terms with the corresponding taxonomy concepts. The frequency of occurring annotations can then be used as a

[11] The complete ontology can be found at http://www.minerva-portals.de/o/complexys.rdf.

[12] http://java.sun.com/j2se/1.5.0/docs/api/java/util/concurrent/package-summary.html

simple indicator for the categorization. The central element of this component is the OntoGazetteer or semantic tagger, that is included in the information extraction system ANNIE[13]. It is not directly applicable to the SKOS CompleXys taxonomy, but can make use of a derived, rule-based version. Therefore, every main category of the domain model gets its own *.lst* gazetteer file, wherein all subordinate terms are listed one per line. A file *mappings.def* defines the mapping rules from the *.lst* files to SKOS concepts. However, the expressiveness of the gazetteer data is very limited, so the relationships can not be transformed.

The KEA Annotator also categorizes a document into the concepts of the CompleXys domain model. It is based on the Keyphrase Extraction Algorithm KEA[14], that analyzes texts in order to identify the most important words or word groups for each one. The idea of leveraging this behavior for the task of semantic data extraction is, that KEA is implicitly capable of scoring terms according to their text importance. While the OntoGazetteer is capable of answering the question "Do taxonomy terms occur in the text and how often?", KEA goes one step further and additionally tries to answer "Are these terms relevant for the text?". In order to do so, it utilizes additional factors like the relative term occurrence in a single text, compared to the occurrence in all processed texts or the SKOS *related* relationships as weight boosting functions. To ensure that the keyphrases are matchable to the domain model anyway, it simply uses the CompleXys taxonomy as a controlled vocabulary for the extraction process. To use this functionality the older KEA GATE plugin was manually adapted to the new KEA version 5.0, which allows the controlled indexing. As categorization model CompleXys is trained with the CiteULike-180 data set[15]. First evaluations indicate, that a well adjusted KEA Annotator is capable of outperforming the competing OntoGazetteer solution by means of precision.

The Open Calais Annotator utilizes the OpenCalais[16] metatagging web service to semantically annotate named entities, events and facts in the text. The so obtained data is not yet used for the domain categorization, but links the data to the wide external set of Calais' stored semantical knowledge base. To exploit these relations has great potential in further improving the categorization, but also for other features like enriching the displayed resources in the front end with additional information.

Finally the Content Writer ensures, that every document is correctly stored in the *Semantic DB*, before the pipeline terminates. However, it also checks if a document has actually exceeded the critical threshold of Onto Gazetteer annotations or Kea annotations, that marks the relevancy for the domain of complexity. If a document fails to pass this test, it is deleted. Furthermore, the annotations of a document are counted and mapped to their corresponding main categories. The document is ultimately regarded as being a member of the most frequently occurring categories.

[13] http://gate.ac.uk/sale/tao/splitch6.html
[14] http://www.nzdl.org/Kea/index.html
[15] 11.01.2010: http://maui-indexer.googlecode.com/files/citeulike180.tar.gz
[16] http://www.opencalais.com/

5 Conclusion and Future Work

In this paper, we have described the CompleXys portal, an information system about complexity, as an example for a system that allows to automatically aggregate, semantically annotate and filter content stemming from a wide variety of sources. We believe that in times of a rapidly growing amount of content that is being dynamically created in ever increasing rates, such systems are an absolute necessity to ensure that users do not "drown in information" and at the same time do not miss relevant information. Only with such intelligent support will we be able to take advantage of this information revolution.

Up to now, the parts of CompleXys dealing with information harvesting, annotating and filtering have been implemented. A first evaluation shows that CompleXys reaches, indeed reasonable precision and recall with acceptable runtime. For more details please refer to [9]. Right now, we are working on integrating our approach to personalization into CompleXys. Once this has been done, the portal will be launched as an information site for the members of the research focus area "Analysis and Management of Complex Systems" at our university and for the general public.

References

1. F. Bakalov, B. König-Ries, A. Nauerz, and M. Welsch. A Hybrid Approach to Identifying User Interests in Web Portals. In *Proc. of the 9th Int. Conf. on Innovative Internet Community Systems*, 2009.
2. F. Bakalov, B. König-Ries, A. Nauerz, and M. Welsch. IntrospectiveViews: An interface for scrutinizing semantic user models. In *Proc. of the 18th Int. Conf. on User Modeling, Adaptation, and Personalization*, 2010.
3. R. Baumgartner, N. Henze, and M. Herzog. The Personal Publication Reader: Illustrating web data extraction, personalization and reasoning for the semantic web. In *Proc. of the 2nd European Semantic Web Conference*, 2005.
4. H. Cunningham, D. Maynard, K. Bontcheva, and V. Tablan. GATE: A framework and graphical development environment for robust nlp tools and applications. In *Proc. of the 40th Anniversary Meeting of the Association for Computational Linguistics (ACL'02)*, 2002.
5. P. Dolog, N. Henze, W. Nejdl, and M. Sintek. Personalization in distributed e-learning environments. In *Proc. of the 13th Int. World Wide Web Conf.*, 2004.
6. S. Handschuh, S. Staab, and F. Ciravegna. S-CREAM - Semi-automatic CREAtion of Metadata. In *Proc. of the 13th Int. Conf. on Knowledge Engineering and Knowledge Management*, 2002.
7. D. Maynard, V. Tablan, K. Bontcheva, H. Cunningham, and Y. Wilks. MUSE: a MUlti-Source Entity recognition system. *Computers and the Humanities*, 2003.
8. B. Popov, A. Kiryakov, D. Ognyanoff, D. Manov, and A. Kirilov. KIM a semantic platform for information extraction and retrieval. *Natural Language Engineering*, 10(3-4):375 – 392, 2004.
9. O. Schimratzki. An approach for semantic enrichment of social media resources for context dependent processing, Diploma Thesis, University of Jena, 2010. Fulltext: http://www.minerva-portals.de/publications/theses/an-approach-for-semantic-enrichment-of-social.

10. Y. Wanga, N. Stash, L. Aroyoa, P. Gorgels, L. Rutledge, and G. Schreiber. Recommendations based on semantically enriched museum collections. *Web Semantics: Science, Services and Agents on the World Wide Web*, 6(4):283–290, 2008.