

Personalized Recommendation of Related Content Based on Automatic Metadata Extraction

Andreas Nauerz¹, Fedor Bakalov², Birgitta König-Ries², Martin Welsch¹

¹IBM Research and Development
Schönaicher Str. 220, 71032 Böblingen, Germany
{andreas.nauerz|martin.welsch}@de.ibm.com

²University of Jena, Institute of Computer Science
Ernst-Abbe-Platz 1-4, 07743 Jena, Germany
{fedor.bakalov|koenig}@informatik.uni-jena.de

Abstract

In order to efficiently use information, users often need access to additional background information. This additional information might be stored at various places, such as news websites, company directories, geographic information systems, etc. Oftentimes, in order to access these different pieces of information, the user has to launch new browser windows and direct them to appropriate resources. In our today's Web 2.0, the problem of accessing background information becomes even more prominent: Due to the large number of different users contributing, Web 2.0 sites grow quickly and, most often, in a more uncoordinated way regarding, e.g., structure and vocabulary used, than centrally controlled sites. In such an environment, finding relevant information can become a tedious task.

In this paper, we propose a framework allowing for automated, user-specific annotation of content in order to enable provisioning of related information. Making use of unstructured data analysis services like UIMA or Calais, we are able to identify certain types of entities like locations, persons, etc. These entities are wrapped into semantic tags that con-

tain machine-readable information about the entity type. The entity types are associated with applications able to provide background information or related content. A location, e.g., could be associated with Google Maps, whereas a person could be associated with the company's employee directory. However, it strongly depends on the individual user's interests and experience which additional information he deems relevant. We therefore tailor the information provided based on the User Model, which reflects the user's interests and expertise. This allows providing the user with in-place, in-context background information on those entities he is likely to be interested in as well as with recommendations to related content for those entities. It also relieves users from the tedious task of manually collecting relevant additional information.

Our main concepts have been prototypically embedded within IBM's WebSphere Portal.

1 Introduction

Consider a manager using the company portal to read a news bulletin about rumors of a planned merger of her company's competitor with another company and the consequences this has for the stock market. In order to make the right decisions based on this piece of information, she needs additional information: background information on this specific

⁰Copyright © 2008 Andreas Nauerz, Fedor Bakalov, Birgitta König-Ries, Martin Welsch, IBM Deutschland Entwicklung GmbH, and University of Jena. Permission to copy is hereby granted provided the original copyright notice is reproduced in copies made.

competitor, the company it might merge with, and a summary of the stock value over the last few months. Maybe she would like to talk to someone from her research department who is familiar with the technologies that will belong to one company if the merger comes through. Today, most likely, all this information is available online. In order to access it, the manager will open new browser windows, will enter the appropriate terms in a search engine (either a global one or one that searches the company's intranet) and will eventually locate this information.

Now, imagine a different scenario: When the manager moves her mouse over the name of the competitor, a new portlet containing relevant information about the competitor pops up, when moving to a mention of the technology, a new window with the related wikipedia page opens. In a separate window a list of experts from her department together with contact details and additional resources related to the news item are displayed. Clearly, this would be a lot more efficient than the first approach. It might even happen that related information, that the manager was not aware of, is brought to her attention: in the popup a new project running by one of the company's research centers could be mentioned, that might be of interest in this context.

In this paper, we present our approach to making this happen. What is needed to realize this vision? First of all, we need a component that automatically analyzes the content, identifies terms that describe certain types of entities (e.g., persons, locations, companies, industry terms, etc.). Our solution uses the Unstructured Information Management Architecture (UIMA)¹ and the Calais² web service to achieve this. Second, we need to link entity types to applications that can provide background information. For instance, the location entity type could result in a call to Google Maps etc. Third, we need to make sure to present the user only with the background information he is interested in. Users will *not* be interested in background on things that are beyond their sphere of interest or that they know everything about, already.

¹<http://www.research.ibm.com/UIMA/>

²<http://www.opencalais.com/>

The remainder of this paper is organized as follows. Section 2 provides a short overview of the previous research that has been done in the area of annotation, personalization, and user modeling. Section 3 provides background information on the main concepts used in this paper as well as outlines the system architecture of our basic recommender system that we implemented in our previous work and describes its drawbacks. Section 4 provides information about the improved system architecture. Finally, Section 5 discusses limitations and future extensions of our system.

2 Related Work

The recommender system described in this paper is related to works in the areas of semantic annotation and tagging, personalization, and user modeling.

Early research work on content annotation pursued a *bottom-up* approach to enrich content of web pages with accompanying semantic annotations. The proposed systems helped to complete two tasks: first, to create ontologies and second, to annotate web pages with ontology derived semantic annotations. But it turned out that leaving the annotation work up to human beings is a tedious task [8].

Today a number of tools for producing semantic annotations exist (Protege [16], On-toAnnotate [19], Annotea [11], CREAM [9], etc.). Authoring tools help to extract metadata from contents of web pages, but, most often, force authors to first create the content and then annotate it in an additional annotation step. Obviously these systems assume that content is created once and rarely updated. This assumption is not true in today's Web 2.0 world where a lot of users continuously contribute content.

To automate the task of annotating content a lot of research has been done in the field of information extraction whose goal is to automatically extract structured information from unstructured machine-readable documents. Although the approach to perform the extraction is often differing, most papers in this area regard information extraction as a proper way to automatically extract semantic annotations

from web content. Thus, such systems (e.g. the World Wide Wrapper Factory [18]) allow for the construction of wrappers by explicit definition of HTML or XML queries. They try and extract detailed structural data out of the pages and require significant training before they can be productive.

Other techniques are based on machine learning: in [21] Yang describes a machine learning approach to extract semantic annotations from web content. Another system applying large-scale text analytics to extract semantic annotations is SemTag [7]. Similarly Li et al. [23] propose a machine learning based automatic annotation approach that allows for annotating web pages at a sentence level.

In [10], Handschuh et al. describe a framework for metadata creation when web pages are generated from a database and the database owners are cooperatively participating in the Semantic Web. In order to create metadata, the framework combines the presentation layer with the data description layer.

Recently, a few novel approaches have been proposed to address the problem of personalized annotation. In [5], Chirita et al. describe a method that automatically generates personalized annotations of website content based on the information stored on the user's desktop. In other work, Ankolekar et al. [3] proposed an architectural solution of website personalization that takes place at the client side and allows personal information about the people to be stored locally.

An important component in personalized recommender systems is the user model. A lot of research has been done in the area of user modeling. A considerable amount of user modeling approaches concentrate on application of semantic technologies for representation and acquisition of user interests and preferences. Pereira and Tettamanzi [17] describe an automated method for acquisition of user models. According to their approach, the initial user models are constructed using the data explicitly provided by the users at the time of registration. The models are then being constantly updated based on the data collected from the users' interaction with the system. Another user modeling approach is presented by Achananuparp et al. in [1]. The authors

describe a method for building semantically enhanced user models based on the analysis of clickstreams and web usage logs.

Our system is based on our previous work on automatic metadata extraction. In [14], we presented a recommender system that identifies information pieces of certain types in text fragments and automatically wraps them into semantic tags, which we then connect to other (external) information sources in order to recommend background information and related content.

However, our previous recommender system did not take into account the user interests and preferences. This led us to a problem of producing large number of irrelevant recommendations. The system highlighted every tagged fragment and linked it to external resources without consideration of relevance of this information to the user. To rectify this problem, we introduced a user model that represents the knowledge about user interests. We use this model to select only those tagged fragments that contain information that is of interest for the user.

3 Background

In this section, we give a brief overview of our previous work that the approach proposed in this paper is based upon. We will first describe the basic concepts underlying that earlier system, then we will introduce the system architecture and, finally, we will identify the weaknesses of the previous approach. The extensions proposed in this paper are aimed at overcoming these weaknesses.

3.1 Concepts

In the following subsections we will first describe an automatic method for extracting semantic annotations from text (or markup, respectively). We need these annotations as the basis for recommendations. We then describe the different concepts for recommending (external) information sources that provide users with background information or related content. Finally, we describe the mechanisms for tailoring Web content to the needs of individual users.

3.1.1 Annotation

Before we continue we want to provide a definition for the terms annotation and semantic tagging: we define **annotation** as the general process of adding meta information about an artifact (e.g. an already semantically tagged information piece inside a text). We define **semantic tagging** as the process of adding meta information by injecting (embedding) tags to XHTML fragments. Thus, semantic tagging is entirely performed at the markup level.

We distinguish three types of annotation. First, *automatic annotation*, where the system analyzes markup to find occurrences of identifiable information pieces of certain types. Second, semi-automatic annotation, where the user can tell the system the type of a piece of information which the system could not unambiguously identify so that it can be wrapped into a semantic tag. Third, manual tagging, where the process of annotation is entirely left up to the user.

In our recommender system, we are mainly interested in automatic annotation. Here, the system processes content of pages and portlets using unstructured information analysis methods in order to automatically extract entities of certain types, such as persons, locations, etc. The system then enriches the extracted entities by wrapping them into semantic tags. Afterwards these semantic tags will be, depending on the information type, associated with application logic that allows for an advanced interaction with the information piece.

For example, if a portlet is comprised of text in which the name of a person *Alice*, the location *New York* or the abbreviation *WWW* occurs, our system analyzes the entire markup, identifies the information pieces just mentioned and wraps them into semantic tags. The additional logic associated to information pieces of type person allows to lookup more detailed information about the person.

Within our solution a click on the semantic tag wrapping a person's name results in a pop-up appearing which provides more detailed information about the person which is retrieved from his profile stored in the company's employee directory. Similarly, a click in the semantic tag wrapping a location results in a pop-

up appearing which provides details about this location by displaying it on a Google Map.

3.1.2 Recommending Background Information

Initially the World Wide Web was basically a read-only medium. Authors put their static documents onto web servers which users read via their browsers. It was up to these authors to decide to which (external) sources to link in order to provide additional background information. However, in our today's Web 2.0 world content is created by entire user communities. Social networking sites, blogs, and wikis facilitate collaboration and sharing between users. Users do not only retrieve information anymore, they contribute content.

Due to the large amount of different users contributing Web 2.0 sites grow quickly and, most often, in a more uncoordinated way than centrally controlled sites. Different users use different terms to describe the same things. Some terms might be well-understood by most users, some might not. Web 2.0 communities are often heterogenous with widely differing user expertise.

Thus looking up terms is needed more frequently and becomes a tedious task. But when reading web sites, users want background information at their fingertips. If they do not understand what an abbreviation or a term stands for, who a certain person actually is, or, where a certain city is actually located, they want to be able to retrieve this information as easily and quickly as possible. They do not want to fire up a search engine to search for another site from which they could probably get the information they want, but rather be provided with that information directly, in-place.

In our Web 2.0 world a lot of this information is available via RSS/Atom feeds, via web services or simply as part of "traditional" web pages. What is missing is an environment that identifies information pieces users would typically want to lookup and means to access additional (external) information sources that can provide users with the necessary background information. We want to provide such an environment which unobtrusively enriches the information pieces this way.

3.1.3 Recommending Related Content

Analyzing occurrences of semantically tagged information pieces also allows us to recommend related content. For instance, if the term *WebSphere Portal* is identified in a news portlet and hence semantically tagged as a product name, our system would provide users with background information about WebSphere Portal probably by linking to the product site.

However, within a Portal system, the same term might occur at many other places, e.g. in a wiki portlet where users have posted some best practices, tips and tricks when working with this product, in a blog where users have commented on the product, and so forth. We track all occurrences and recommend an appropriate subset of them as related content as soon as the user interacts with one single occurrence. The order in which recommendations are listed depends on how often people have interacted with a certain occurrence so far.

This can even be taken one step further. We also allow users to annotate already semantically tagged information pieces. This way we can recommend related content not only by having identified "exactly matching" occurrences of semantically tagged information pieces, but also by having identified similarly annotated, but differently semantically tagged, information pieces. In [15], we have described an algorithm that allows us to determine relatedness between even differently annotated resources which allows us to provide even more and accurate recommendations to related content.

The power of the concepts just described is based on the fact that we make use of the entire communities' collective intelligence in addition to the fully automatic semantic annotation mechanism. The entire community cannot only contribute content anymore, but also describe relations between information pieces being available. As collective intelligence always outperforms single users' [20], we can assume that the community will always be able to categorize content better than even the best author could.

Another general assumption is that annotating expresses interest in a resource. Hence, we can assume that information pieces being anno-

tated more often by a user are of higher importance to him. Even more important: since annotating is a collaborative process we can also assume that resources being annotated more often by all users are of higher importance to the entire community. Thus, analyzing users' annotation behavior allows us to better understand their interests and preferences and allows us to recommend "more interesting" content easier.

3.1.4 Web Personalization

Today we experience not only the massive growth of information on the Internet and companies' intranets, but also the emergence of new forms of information. Users find the information they need in various systems, such as news portals, online shops, geo-information systems, blogs, wikis, and many others. All these systems appear to be potential sources for background information and related content, hence can be used for providing recommendations on arbitrary number of extracted entities.

On the other hand, the availability of huge amounts of data can become a serious threat to the usability. Large number of recommendations might distract the user from the main content and cause the user's frustration. Therefore, we need a mechanism being able to elicit the user interests and provide only the content that is relevant to these interests. Such mechanism is defined as personalization.

In the area of Web Information Systems, personalization is a process of adapting Web content to the needs of individual users or a group of users based on the knowledge about the user interests. The goal of personalization is to provide users with the information they need without asking them explicitly for it. Personalization is a broad area and includes recommender systems, adaptation, and customization [13].

In our particular case, we aim to use personalization techniques in order to provide recommendations to background information and related content that is relevant to the user's information needs. Our method uses the knowledge about users to select the pieces of information among the automatically extracted entities that match the user interests. These selected pieces are further linked to external resources

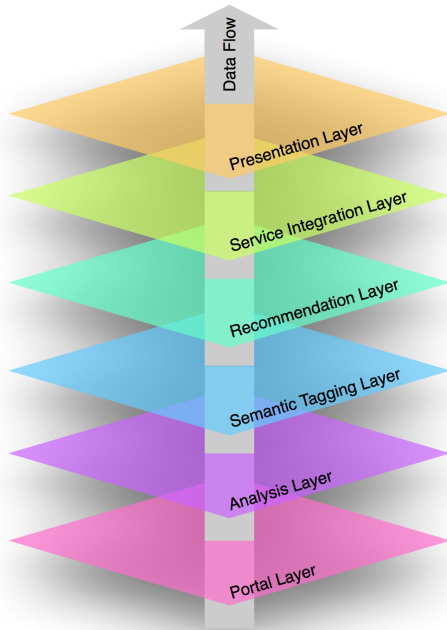


Figure 1: Layered architecture

in order to issue the personalized recommendations.

The key component of personalization is the user model that describes who the users are, what interests they have, and how they behave. The user models vary in terms of information they store, so do the approaches and techniques used to construct these models. In our recommender system, we use a hybrid approach to collect knowledge about users.

3.2 Basic Recommender System

In this section, we describe the architecture of our earlier system. This system is able to automatically annotate Portal resources and to use these annotations to provide background information and recommend related content.

The architecture of our basic recommender system is six-layered (cp. Fig. 1): content (in the form of markup) is delivered by the Portal or, to be more precise, by portlets residing on pages part of the Portal. The content is then analyzed by the analysis engines part of our *Analysis Layer*. Here, so called *Annotators* ex-

tract information pieces like people, locations etc. from the markup received. At the *Semantic Tagging Layer* the results of the analysis are converted into proper markup. This means that in this step we construct markup which we can wrap around identified information pieces; in other words, in this step we perform the actual semantic tagging. The *Recommendation Layer* determines other occurrences of the same semantic tags and occurrences of similarly user-annotated information pieces. At the service integration layer we determine the (external) services to which we could connect for each single semantic tag to provide users with additional information. For example, for a semantic tag corresponding to a person we could connect to the company’s employee directory, or to Google Maps to visualize the work location of the person. At the *Presentation Layer* we associate (client-side) application logic to the semantic tags that allows for the actual interaction (i.e. for the invocation of (external) services etc.).

We illustrate the details of our architecture in the following sections.

Portal Layer. Portals provide users with a central point of access to information. They aggregate information and services from various sources.

Portals are comprised of pages which are comprised of portlets. Portlets are applications that provide specific pieces of content. They are managed by a portlet container, that processes requests and generates dynamic content. A portlet is a pluggable user interface component. The content that is generated by a portlet is called a fragment. In contrast to a servlet that generates complete documents, the fragments generated by a portlet are pieces of markup (e.g. HTML, XHTML). In its turn these fragments can be aggregated with other fragments to form a complete document. Usually several portlets are aggregated to a portal page that represents, for instance, a complete HTML document.

Filters make common aspects available to portlets. Instead of implementing a certain functionality within each portlet, a filter implements the aspect centrally. Different filters can be combined via a filter chain.

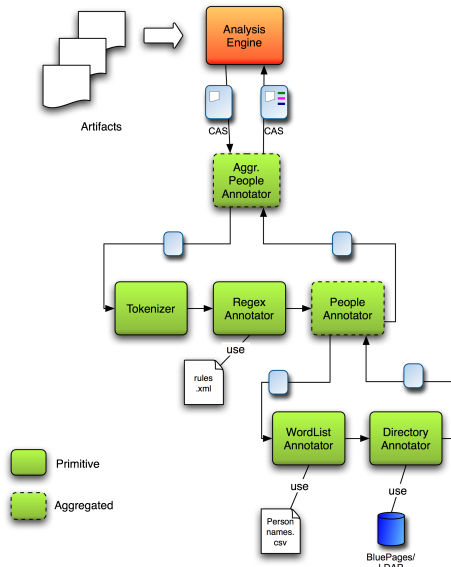


Figure 2: UIMA Analysis Engine

We use filters to change the response (i.e. to “enrich” the original markup by injecting semantic tags) before it reaches the client. This means that instead of simply transmitting the markup from the Portal to the client we pass the markup to the analysis and its subsequent layers to enrich it before we deliver it to the client.

Analysis Layer. For analyzing unstructured information such as text residing inside portlets our original system relied on the Unstructured Information Management Architecture (UIMA). UIMA is a software architecture for developing and deploying unstructured information management (UIM) applications. Typically such UIM applications analyze large volumes of unstructured information in order to structure the relevant knowledge to the end user.

Typical UIM applications make use of a variety of technologies including statistical and rule-based natural language processing (NLP), information retrieval, machine learning, ontologies, and automated reasoning. UIM application may also include structured data sources like databases or wordlists to help resolve the semantics of the unstructured source. UIMA gives developers the ability to create

component-based UIM applications.

Our UIMA analysis engine is constructed from components called *Annotators*. The *Annotators* implement the actual analysis logic. They analyze an artifact (e.g. a text document) and create metadata about that artifact. An *Analysis Engine (AE)* may contain a single *Annotator (Primitive AE)*, or it may be a composition of others and therefore contain multiple *Annotators (Aggregated AE)*.

Fig. 2 shows an *Aggregated AE* that consists of three *Annotators*: *Tokenizer*, *Regex Annotator*, and *People Annotator*, where the latter is also an *Aggregated AE* containing a *Wordlist Annotator* and a *Directory Annotator*.

The *Tokenizer* splits the input into *Token Annotations* using a simple whitespace segmentation. The result of tokenization is a set of tokens and sentence annotations. For example, the text “The author of this paper is Andreas Nauerz” is tokenized into eight tokens and annotated as a sentence.

The *Regex Annotator* retrieves the tokens from the *Token Annotator’s Annotation Index* and compares each token against a regular expression that matches potentially a person name and creates a *PotentialPersonName Annotation*. In our sample the words *Andreas* and *Nauerz* both start with an uppercase character, which might be a good indicator to represent a name within an English text. After that the first aggregated *Wordlist Annotator* that belongs to the *People Annotator* compares the *PotentialPersonName’s* first name with a wordlist that contains common first names. Because *Andreas* is a common first name the wordlist includes it.

The next *Annotation* comes from the *Regex Annotator* too, but it does not occur in the wordlist because *Nauerz* is not a common first name. Nevertheless, due to the minimal distance between these *Annotations* the *Wordlist Annotator* decides to create a *PersonName Annotation* where it sets the so called features *firstName* and *lastName* to *Andreas* and *Nauerz* respectively.

The final *Directory Annotator* queries the employee directory to prove if a person with this name exists. For stored entries of people it creates a *Person Annotation* instance that is additionally extended with other information

from the directory (e.g. birth of date, phone number, office).

The result of an *Annotator's* work are so called *Feature Structures*, which are data structures that have a type and a set of (attribute, value) pairs. An annotation is a particular type of *Feature Structure* that is attached to a region of the artifact being analyzed (e.g. a span of text in a document).

All *Feature Structures*, including annotations, are represented in the UIMA *Common Analysis Structure (CAS)*. The *CAS* is the central data structure through which all UIMA components communicate. In fig. 2 we have illustrated how one *CAS* is passed from one *Annotator* to another. Each *Annotator* can make contributions to the *CAS' Annotation Index*.

The *Annotators* of an *Aggregated AE* are arranged in a *Flow*. In the example above the *AE* uses the simplest possible *Flow*, a *Fixed Flow* that is included in the UIMA framework. The purpose of a *Flow* is to decide, which *Annotator* is included in a analysis. Because a *Flow* has access to the *CAS*, very sophisticated *Flows* can be implemented by *Annotator* developers.

Semantic Tagging Layer. At this level the results of the annotation must be converted into proper markup. This layer needs to know about the annotation objects created by the analysis layer. The decision, if and how annotation objects are rendered to semantic tags is made here. This means that this layer is responsible for generating valid HTML fragments representing semantic tags which can later be injected into the original text (or markup, respectively).

Recommendation Layer. The *Recommendation Layer* constructs, for each semantically tagged information piece, a list comprised of references to the other "exactly matching" semantically tagged information pieces. This means that when a user for example clicks the information piece *Andreas Nauertz* (a person name obviously), he will be provided with a list of all other places where this information piece occurs in the Portal system.

The layer also constructs a similar list comprised of references to "similarly" annotated information pieces. This means that when a user

is for example clicking the previous introduced person *Alice*, he will be provided with a list of all places where related persons (in the sample given *Bob* and *Charly*) occur in the Portal system.

Service Integration Layer. The *Service Integration Layer* scans the markup that has meanwhile been enriched with semantic tags. It determines the so called *External Service Connectors* that are available for each type of semantic tag and generates the necessary invocation code. The available services are determined via the *Service Registry* that maps each information type to its available services. A service connector is a piece of application logic that allows to connect to an external service like e.g. Google Maps.

Presentation Layer. The *Presentation Layer* scans the markup that has meanwhile been enriched with semantic tags which additionally have been assigned the information which recommended content and which *External Service Connectors* are available, too. At this layer we add (client-side) application logic for each semantic tag allowing for the actual invocation of the external services or for accessing related content.

3.3 Limitations

Our system described above provides basic recommender functionalities allowing users to access related content and background information on the entities that were extracted automatically by the UIMA analysis engines. However, after an initial evaluation of the system, we have identified a number of limitations and drawbacks:

1. **Irrelevant recommendations.** The system provides additional information on every object that was tagged by the analysis engine without respect to the relevance of the tagged object to the user. The system does not take into account user interests and preferences. Therefore, it leads to a large number of irrelevant recommendations that harm the overall usability of the system.

2. **Hardcoded binding of information types to sources of related content.** In our basic recommender system, the information types are mapped to the sources of additional information by the developer. This makes it impossible for the administrators of the systems to specify new sources of related content and background information.
3. **Huge amount of work required to develop analysis engines.** Development of unstructured information analysis engines, in general, is a technologically complicated process, which, in most cases, requires a large corpus of annotated documents for training the models and extensive vocabularies of terms and concepts in a certain domain as well as sophisticated regular expression patterns and machine learning algorithms. Therefore, development of an analysis engine for a new information type may become unacceptably expensive.

4 Architectural Extensions

Considering the limitations and drawbacks of our basic recommender system, we propose a number of architectural extensions and improvements. After a brief overview, we will describe each of them in detail in this section.

1. **Generation of user-specific recommendations.** The recommender system must take into account user interests and preferences in order to provide recommendations tailored to the needs of individual users. In order to achieve this, the system obviously needs some knowledge about the user. Therefore, we have added a user model to our architecture. This user model contains static and dynamic information about the user, in particular about her interests and areas of expertise.
2. **Mechanism for flexible mapping of information types to information sources.** We also need a flexible mechanism to specify the rules that govern what sources should be queried for additional information given a certain concept that the user is interested in. For this purpose, we

have introduced a personalization model, which allows multidimensional representation of factors that influence the decision on what information to deliver.

3. **Harnessing external unstructured information analysis engines.** In order to reduce the amount of work required for development of unstructured information analysis engines, the system must be able to use external engines and services for automatic tagging. As an example, we provide an integration with the Calais service and show how this (currently freely available) service can be used instead of our custom built UIMA engine.

In order to implement the above-mentioned features, we have made a number of extensions. Fig.3 shows the system architecture of our extended recommender system. We introduced a *Domain Model* that defines top-level and domain-specific concepts and relations among them as well as a *Task Model* that defines common and domain-specific information-gathering actions. The layered architecture of the old system (fig.1) was extended by a *Personalization Layer* that resides between the *Analysis Layer* and the *Semantic Tagging Layer*. The layer includes a *User Model* that contains knowledge about user features and a *Personalization Model* that defines the rules governing what information should be delivered to the user when she encounters a certain object on a Portal page. Finally, the UIMA analysis engines, located on the *Analysis Layer*, were supplemented by an external unstructured text analysis service, in this case *Calais*. The proposed architectural improvements are described in more detail in the following sections.

4.1 Domain Model

The key element in our system that both the user model and the concepts-actions model refer to is the *Domain Model*. A domain model is a structure that defines concepts and relations among them in a given domain, e.g., finance, medicine, biology, etc [2]. We have chosen the finance domain for our proof-of-concept implementation. Therefore, in our domain model

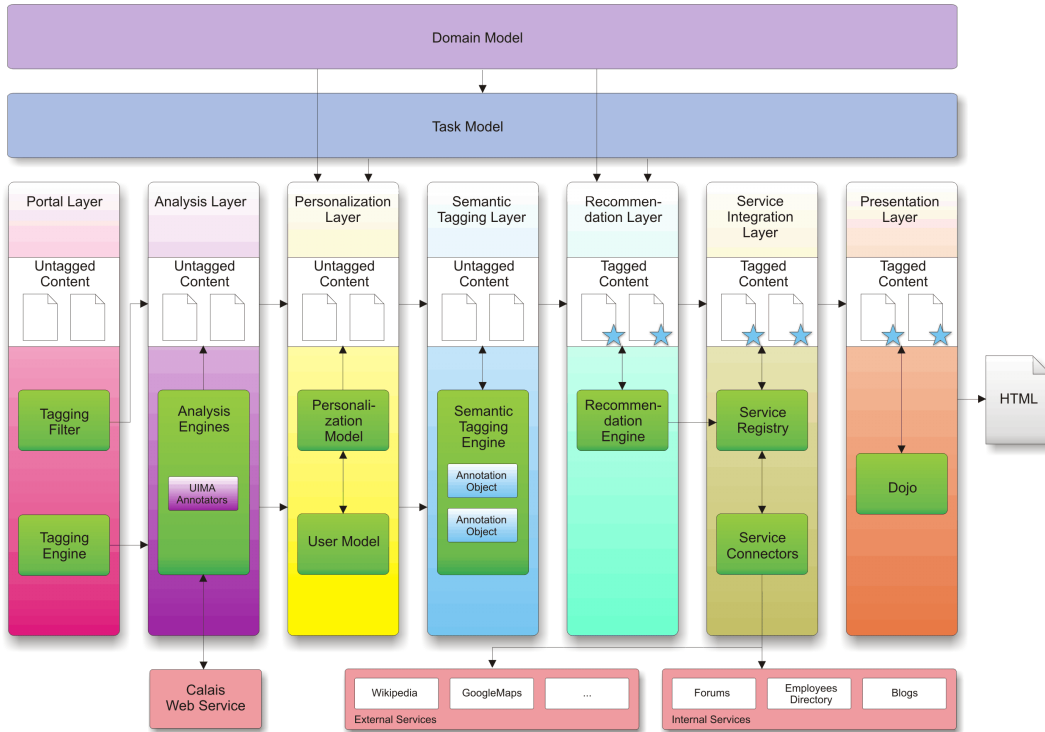


Figure 3: Extended System Architecture

we define the concepts that the users from financial realm may work with, such concepts as stock, bank, account, etc.

In our recommender system, the domain model is represented as an ontology, which defines concepts as ontological classes by specifying their properties and relations to other classes. E.g., an event *Acquisition*, denoting the fact of acquiring one company by another one, can be described as a subconcept of *FinancialTransaction* concept and described by such attributes as *date*, *acquiree*, *acquirer*, and *transationAmount*.

The domain model also contains instances of concepts. E.g., for concept *Company*, the model contains such instances as "Microsoft", "IBM", "Google", etc. Class instances might be required to represent specific user interests in the user model. For example, our manager is reading a news article about an acquisition of *Company A* by *Company B*. The manager has been dealing with the former company for many years, hence she possesses expert knowledge on it. However, she has never heard any-

thing about the latter company, therefore, she may need background information on it. In order to model such situation in the user model, we need to have instances of *Company A* and *Company B* explicitly defined in the domain model.

4.2 Task Model

In addition to the definition of concepts and relationships between them, we need a structure that defines domain tasks, the actions that can be performed by users in a certain domain. We specify such actions in a *Task Model*. Here, we concentrate on the information-gathering activities that could be taken by the user on a certain object contained in the text she is reading. The model itself is represented as an ontology. Each task is defined as an ontological concept and described by the set of input and output concepts. E.g., for the task *get-CompanyAddress*, we specify *Company* as an input concept and *Address* as an output concept. The input and output concepts are de-

rived from the domain model.

As we have chosen finance domain for our proof-of-concept implementation, the task model in our system defines most of the information-gathering actions performed by employees of financial organizations, e.g.: *getStockQuotes*, *getCurrencyExchangeRates*, *getMarketStatistics*, etc.

The information-gathering activities are used in the *Personalization Model* to define the information that should be provided to the user when a certain object is present in the document.

4.3 User Model

Many recommender engines need information about the user to generate relevant recommendations. In order to provide these engines with the necessary input, we construct a *User Model* reflecting various user features. Our user model consists of two parts: static and dynamic. The static part represents the user’s demographic characteristics, such as date of birth, gender, mother tongue, etc. The dynamic part of the model represents the user interests and expertise.

The dynamic part is constructed as an overlay model [4]: The interests and expertise of an individual user are represented as references to the concepts defined in the domain model. Also we indicate the values that show the degree of the user’s interest and expertise in these concepts. To specify these values, we use set of linguistic variables: $\{not\ interested, partially\ interested, interested\}$ and $\{novice, medium, expert\}$. We associate these variables with fuzzy sets, which are defined by the corresponding membership functions [22, 12].

The membership functions are based on the concept frequency value, which denotes importance of a certain concept for the user. To compute this value we need to know what concepts and how many times the user has encountered in the past. For this purpose, we analyze content of the pages accessed by the user and extract certain named entities using the Calais Web service and UIMA framework. These two analysis engines extract such entities as industry term, technology, company, location, etc. Occurrences of those entities are recorded in

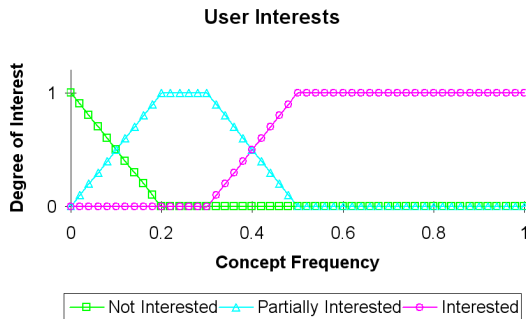


Figure 4: Membership functions for user interests

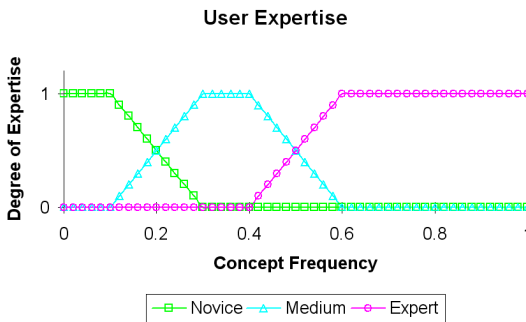


Figure 5: Membership functions for user expertise

the user log, which allows us to estimate importance of a concept by computing the concept frequency value:

$$CF_{i,j} = \frac{c_{i,j}}{\sum_k c_{k,j}} \quad (1)$$

where c is the number of occurrences of the concept $_i$ for the user $_j$, and the denominator is the total number of occurrences of all concepts registered for the user $_j$. The computed concept frequency is a real number that can take values from 0 to 1. We use this value to define the membership functions that represent the degree of user interest; μ_{ni} , μ_{pi} , μ_i show the degree to which the user is *not interested*, *partially interested*, and *interested* in the given concept (Figure 4). In the same way we define the membership functions to represent the degree of user expertise; μ_n , μ_m , μ_e show the degree of the user being *novice*, *medium*, and *expert* in a given concept (Figure 5).

4.4 Personalization Model

In the *Personalization Model*, we define the personalization rules that govern what information is provided to the user. These rules basically define which information-gathering actions should be delivered to a user with certain interests and knowledge when she encounters a certain concept in the text she is reading. The personalization rules are specified in the Event-Condition-Actions[6] form as shown in Listing 1, where *event* denotes a situation when the user encounters a certain concept in the document she is reading, *condition* is combination of user features and context descriptions, and *actions* are the information gathering-actions that should be provided to the user when the event occurs.

```

on
  (event)
if
  (condition)
then
  (actions)

```

Listing 1: Personalization Rule Formula

In order to combine different user features and context descriptors, we represent them as dimensions. In its turn, the actions are specified at the intersections of these dimensions. For instance, in order to represent a personalization rule for an event when a student of a business school, interested in banking, with no knowledge in this field, is reading a news article that contains a bank in the text, we need to create three dimensions: *User Interests*, *User Expertise*, and *Document Concept* and plot values "Banking", "Novice", and "Bank" respectively. At the intersection of these values, we specify what information should be delivered to the user, which in this case, could be the website of the bank, an encyclopedia article about the bank, and news related to this bank (Figure 6).

4.5 Service Registry

The *Service Registry* is a central database for storing information about internal and external sources for background information and related content. The registry maps each action

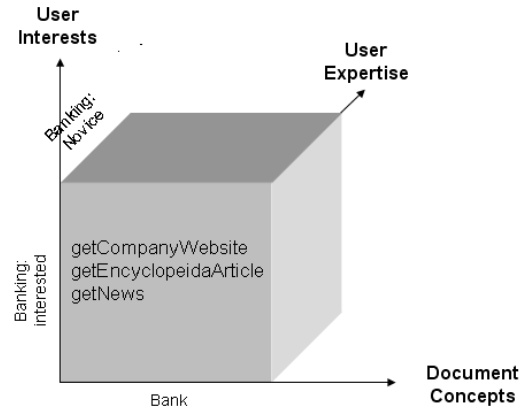


Figure 6: An example of multidimensional representation of personalization rules

defined in the *Task Model* to the service that "does" that action. E.g., for the action *getMap*, we specify the Google Maps service, which displays a given location or address on the map. Other services might be internal to the portal, e.g., there could be a service that finds related content by searching for pieces of information with the same tags as the piece of information currently considered.

Every service is provided with a WSDL description (Web Service Description Language³), which specifies the technical details required to execute the service. Invocation of the internal and external services is carried out by the *Service Connector*.

4.6 Calais Web Service Integration

In addition to the UIMA analysis engines, we propose a component that harnesses the Calais Web service for semantic tagging of text documents. Calais is a RESTful service that can receive an HTML, XML, or plain text document as an input and provide back a semantically annotated document in RDF format. The service extracts business-related entities such as company, currency, industry term, technology, etc. It also supports extraction of certain events and facts, such as business relation, bankruptcy, company investments, etc.

³<http://www.w3.org/TR/wsd1>

```

<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#" .
  xmlns:c="http://s.opencalais.com/1/pred/">.
....
  <rdf:Description rdf:about="http://d.opencalais.com/456492acc9e0">.
    <rdf:type rdf:resource="http://s.opencalais.com/type/City"/>.
    <c:name>Armonk</c:name>.
  </rdf:Description>.
  <rdf:Description .
    rdf:about="http://d.opencalais.com/f95337/Instance/2">.
    <c:subject rdf:resource="http://d.opencalais.com/456492acc9e0"/>.
    <c:detection>.
      [ earnings IBM reported its Q3 earnings report in ]Armonk[, N.Y.
        The report was presented by chief].
    </c:detection><c:offset>151</c:offset><c:length>6</c:length>.
  </rdf:Description>.
  ....
  <rdf:Description rdf:about="http://d.opencalais.com/add43142d9d1">.
    <rdf:type rdf:resource="http://s.opencalais.com/type/Company"/>.
    <c:name>IBM</c:name>.
  </rdf:Description>.
  <rdf:Description rdf:about="http://d.opencalais.com/ba4633f95337/Instance/5">.
    <c:subject rdf:resource="http://d.opencalais.com/43142d9d1"/>.
    <c:detection>.
      [quarterly earnings ]IBM[ reported its Q3 .
        earnings report in Armonk, N.Y.].
    </c:detection><c:offset>112</c:offset><c:length>3</c:length>.
  </rdf:Description>.
  ....
</rdf:RDF>.

```

Figure 7: RDF Output

For example, if we provide the Calais service with the following string as an input "IBM reported its Q3 earnings report in Armonk, N.Y.", the returned result will contain a metadata structure containing four descriptions: "IBM" as a *Company*, "IBM reported its Q3 earnings" as a *CompanyEarningsAnnouncement*, "Armonk" as a *City*, and "N.Y." as a *StateOrProvince*. The analysis result is represented in RDF format as it is shown in fig. 7.

Integration with the Calais Web service considerably alleviates the development task. In contrast to the UIMA analysis engines, the Calais services does not require manually annotated corpus of documents for training the analysis engines or regular expression patterns.

5 Conclusion and Future Work

In this paper we have presented our approach to an automated, personalized recommenda-

tion of background information and related content in portal systems. The main building blocks of our approach are: Semantic annotations of content via UIMA or other services like Calais, a concept-actions model that links semantic tags to possible actions available for this type of concept, a service registry that allows to find services that are able to perform a certain action, a user model that contains information about user interests and expertise, and a personalization engine that chooses which content to recommend which background information on based on the user interest and expertise. This approach is an extension of an earlier approach that also allowed for automatic recommendation but had several drawbacks: Recommendations were not user-specific, linking between concepts and services was static, and we needed a custom-made annotation engine to find the semantic tags. We believe that the approach proposed in this paper overcomes these shortcomings. We are currently extending the existing implementation which has been done within IBM's Websphere Portal, to accommodate the extensions. Once the implementation

is completed, we intend to run thorough user studies to compare the two approaches. Despite its shortcomings, already the earlier approach was positively evaluated by test users.

Acknowledgements and Trademarks

We would like to thank Thomas Fischer, a student at the Department of Business Informatics at the University of Jena, for his contribution to this paper.

IBM and WebSphere are trademarks of International Business Machines Corporation in the United States, other countries or both. Other company, product and service names may be trademarks or service marks of others.

About the Authors

Andreas Nauerz holds BSc. and MSc. degrees in Information Technology and Computer Science. He is currently working at the IBM Laboratories at Boeblingen, Germany and is, at the same time, PhD student acting as the head of the Minerva research group at the University of Jena. His research focus lies on the application of adaptation and recommendation technologies to Web Portals by leveraging modern Web 2.0 and Social Computing techniques. Andreas has been co-organizer, committee member and reviewer for various conferences, workshops and journals.

Fedor Bakalov received his M.Eng. in Information Management from Asian Institute of Technology in Thailand in 2007. He is currently working towards his Ph.D. at the University of Jena. His research interests include adaptive hypermedia, mashups, user modeling, and semantic web.

Birgitta König-Ries holds the Heinz-Nixdorf Endowed Chair for Practical Computer Science at the University of Jena. Her research interests include the next-generation of portals, semantic web technologies and their evaluation, and more generally the transparent integration of distributed resources.

Martin Welsch is a senior member of the IBM WebSphere Portal development team at

the IBM Laboratories in Boeblingen, Germany and Honorary Professor for Practical Computer Science at the University of Jena. His research and teaching interests include portal web technologies, pervasive computing and operating systems.

References

- [1] Palakorn Achananuparp, Hyoil Han, Olfa Nasraoui, and Roberta Johnson. Semantically enhanced user modeling. In Yookun Cho, Roger L. Wainwright, Hisham Hadad, Sung Y. Shin, and Yong Wan Koo, editors, *SAC*, pages 1335–1339. ACM, 2007.
- [2] A. Gómez-Pérez and. *Ontological Engineering*. Advanced Information and Knowledge Processing. Springer, 2003.
- [3] Anupriya Ankolekar and Denny Vrandečić. Kalpana - enabling client-side web personalization. In Erik Duval, editor, *Proceedings of HT'08 - Hypertext 2008*, Pittsburgh, Pennsylvania, JUN 2008. ACM, ACM.
- [4] Peter Brusilovsky and Eva Millán. User models for adaptive hypermedia and adaptive educational systems. In Peter Brusilovsky, Alfred Kobsa, and Wolfgang Nejdl, editors, *The Adaptive Web: Methods and Strategies of Web Personalization*, volume 4321 of *Lecture Notes in Computer Science*, chapter 1, pages 3–53. Springer, Berlin, Heidelberg, 2007.
- [5] Paul A. Chirita, Stefania Costache, Wolfgang Nejdl, and Siegfried Handschuh. P-tag: Large scale automatic generation of personalized annotation tags for the web. In Carey L. Williamson, Mary Ellen Zurko, Peter F. Patel-Schneider, and Prashant J. Shenoy, editors, *Proceedings of the 16th International Conference on World Wide Web*, pages 845–854. ACM, 2007.
- [6] Corporate Act-Net Consortium. The active database management system manifesto: a rulebase of adbms features. *SIGMOD Rec.*, 25(3):40–49, 1996.

- [7] Stephen Dill, Nadav Eiron, David Gibson, Daniel Gruhl, Ramanathan V. Guha, Anant Jhingran, Tapas Kanungo, Sridhar Rajagopalan, Andrew Tomkins, John A. Tomlin, and Jason Y. Zien. Semtag and seeker: bootstrapping the semantic web via automated semantic annotation. In *WWW*, pages 178–186, 2003.
- [8] M. Erdmann, A. Maedche, H. Schnurr, and S. Staab. From manual to semi-automatic semantic annotation: About ontology-based text annotation tools, 2000.
- [9] Siegfried Handschuh and Steffen Staab. Authoring and annotation of web pages in cream. In *WWW*, pages 462–473, 2002.
- [10] Siegfried Handschuh, Steffen Staab, and Raphael Volz. On deep annotation. In *WWW*, pages 431–438, 2003.
- [11] José Kahan, Marja-Riitta Koivunen, Eric Prud’hommeaux, and Ralph R. Swick. Annotea: an open rdf infrastructure for shared web annotations. *Computer Networks*, 39(5):589–608, 2002.
- [12] Alenka Kavcic. Fuzzy user modeling for adaptation in educational hypermedia. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 34(4):439–449, Nov. 2004.
- [13] Maurice D. Mulvenna, Sarabjot S. Anand, and Alex G. Büchner. Personalization on the net using web mining: introduction. *Commun. ACM*, 43(8):122–125, 2000.
- [14] Andreas Nauerz, Michael Juninger, and Shiwan Zhao. A recommender based on automatic metadata extraction and user-driven collaborative annotating. In *ReColl’2008: International Workshop on Recommendation and Collaboration*, 2008.
- [15] Andreas Nauerz, Stefan Pietschmann, Rene Pietzsch, and Shiwan Zhao. A framework for tag-based adaptation in web portals. In *WWW*, 2008.
- [16] Natalya Fridman Noy, Michael Sintek, Stefan Decker, Monica Crubézy, Ray W. Ferguson, and Mark A. Musen. Creating semantic web contents with protégé-2000. *IEEE Intelligent Systems*, 16(2):60–71, 2001.
- [17] Ce’lia Da Costa Pereira and Andrea Tetamanzi. An evolutionary approach to ontology-based user model acquisition. In Vito Di Gesù, Francesco Masulli, and Alfredo Petrosino, editors, *WILF*, volume 2955 of *Lecture Notes in Computer Science*, pages 25–32. Springer, 2003.
- [18] Arnaud Sahuguet and Fabien Azavant. Looking at the web through XML glasses. In *Conference on Cooperative Information Systems*, pages 148–159, 1999.
- [19] S. Staab, A. Maedche, and S. Handschuh. An annotation framework for the semantic web, 2001.
- [20] James Surowiecki. The wisdom of crowds. *American Journal of Physics*, 75(2):190–192, February 2007.
- [21] Hsin-Chang Yang. Bridging the www to the semantic web by automatic semantic tagging of web pages. In *CIT*, pages 238–242, 2005.
- [22] L. A. Zadeh. Fuzzy sets. *Information and Control*, 8(3):338–353, June 1965.
- [23] Lei Zhang and Yong Yu. Learning to generate cgs from domain specific sentences. In *ICCS*, pages 44–57, 2001.