Natural Language Processing for Semantic Assistance in Web Portals

Fedor Bakalov*, Bahar Sateli†, René Witte†, Marie-Jean Meurs†,‡, Birgitta König-Ries*
*Institute for Computer Science, Friedrich Schiller University of Jena, Germany
†Semantic Software Lab, Concordia University, Montréal, QC, Canada
‡Centre for Structural and Functional Genomics, Concordia University, Montréal, QC, Canada

***Abstract*—Web portals are a major class of web-based content management systems. They can provide users with a single point of access to a multitude of content sources and applications. However, further analysis of content brokered through a portal is not supported by current portal systems, leaving it to their users to deal with information overload. We present the first work examining the integration of natural language processing into web portals to provide users with semantic assistance in analyzing and interpreting content. This integration is based on the portal standard JSR286 and open source NLP frameworks. Two application scenarios, news analysis and biocuration, highlight the feasibility and usefulness of our approach.**

I. Introduction

Web portals emerged in the late 1990s, primarily as gateways to different information resources available on the Internet or within an enterprise intranet. Nowadays, a large number of organizations use portals extensively as a single-point access to information, applications, and people. Application domains for portals vary from commercial (enterprise and marketplace portals) to non-profit applications (governmental and educational portals). However, due to the exponential growth of the amount of content available through web portals, it becomes more difficult and time-consuming for users to deal with this information. Current portal systems enable users to obtain relevant content using, for instance, keyword search or directory structures. However, a serious bottleneck remains by reading and interpreting the retrieved content. Oftentimes, portal search engines return several hundreds or even thousands of hits for a simple search query.

Natural language processing (NLP) and related semantic technologies promise to support users in analyzing, transforming, and creating knowledge from large amounts of content. However, it is an open question how exactly these technologies can be combined with existing information system infrastructure like web portals, in a way that brings measurable improvements to their users.

In this paper, we describe an architectural solution and a graphical user interface to support portal users in retrieving, transforming, and interpreting content using NLP tools. We elaborate on an extension that integrates the web portal technology with the Semantic Assistants framework [1], an extensible software architecture that allows invoking literally any NLP or text mining tool using either Web Services or Application Programming Interfaces (API). We illustrate how an average user can benefit from NLP in a web portal. For instance, we show how NLP support in a news aggregating portal can help users to find interesting and relevant news stories and grab key points of a single news story. Moreover, we illustrate how NLP tools in a biochemical literature portal can alleviate the information seeking tasks of scientists.

II. Background and Related Work

Our contribution in this paper is a solution that enables portal users to use a variety of tools for *Natural Language Processing* within a portal environment. We achieve this by extending the *web portal technology* with a number of components that allow invoking NLP applications encapsulated as web services running on the Semantic Assistants framework. In this section, we introduce these two technologies.

A. Portal Technology

A web portal is a web application that provides users with a unified access to various information resources and services. For instance, personal portals like My Yahoo![1] provide users with a wide spectrum of services, such as email, maps, latest news, financial information, weather forecast, entertainment and communication, and many more. Apart from the aggregation of content and services, important properties of a web portal include single sign-on, consistent look-and-feel, and personalization.

One of the most widely used industry standards for portal technology is the Java Portlet Specification JSR286[2]. This standard defines an Application Programming Interface (API) for developing portlet applications in the Java programming language. According to the standard, the portal infrastructure consists of three major components: portlets, portlet container, and portal server. A *portlet* is a pluggable user interface component that provides a specific piece of content or an application. Some concrete examples of portlets are: a map portlet, a portlet for flight booking, a car rental portlet, or a weather forecast portlet. Portlets can be aggregated into a portal page (Fig. 1), where they can work together to support users in complex tasks, e.g., travel planning. Portlets are displayed in portlet windows. Depending on the portal configuration, a portlet window may contain title and control buttons to configure, minimize, maximize, and close the portlet.

[1] My Yahoo!, http://my.yahoo.com/

[2] Java Portlet Specification, http://jcp.org/aboutJava/communityprocess/final/jsr286/

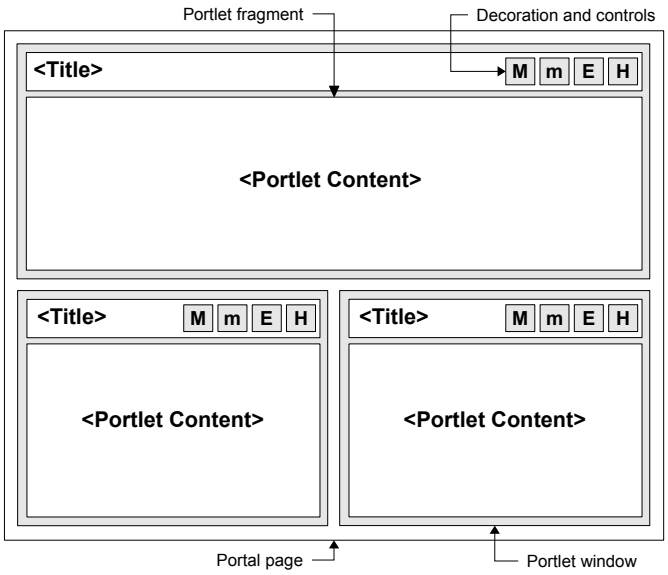

Fig. 1. Portal page (adapted from JSR286)

A *portlet container* is the component that provides the execution environment for portlets and is responsible for managing the life cycle of portlet instances, i.e., their instantiation, initialization, use, and end of service. It is also responsible for handling the inter-portlet communication and storing the portlet preferences. Finally, a *portal server* is a mediator component operating between the client and the portlet container. It is responsible for aggregating portlets into a portal page and submitting user requests from the portal page to the portlet container.

B. Semantic Assistants Framework

The Semantic Assistants [1] architecture, depicted in Fig. 2, is an existing open source service-oriented framework that brokers context-sensitive NLP pipelines as W3C standard web services[3]. The goal of this framework is to bring NLP techniques directly to end users, by integrating them within common desktop applications, such as word processors or email clients. Towards this end, the core idea of the Semantic Assistants approach is to take existing NLP frameworks, like GATE [2], and wrap their concrete analysis pipelines so that they can be brokered through a service-oriented architecture, allowing desktop clients connected to the architecture to consume the NLP services via a plug-in interface.

The NLP pipelines in the Semantic Assistants repository are formally described using the OWL language, which allows the Semantic Assistants server to dynamically discover them and reason on their capabilities before recommending them to clients. Any service deployed in the repository is automatically available to all clients connected to the architecture, using standard WSDL[4] interface descriptions.

The integration of new clients into the architecture is achieved via designing plug-ins. This is further facilitated by the Semantic Assistants Client-Side Abstraction Layer (CSAL). CSAL is essentially a Java archive library of common communication and data transformation functionality that can be reused by clients to communicate with the Semantic Assistants server and transform NLP results to other useful data types. By integrating a client into the Semantic Assistants architecture, its users are not concerned with the implementation or integration of NLP services: from their point of view, they only see context-sensitive Semantic Assistants relevant for their task at hand. Thus, in this work we investigated how web portals can be integrated into this framework (see Fig. 2). As discussed in Section III, integrating Semantic Assistants with portal technology brings a number of portal-specific challenges.

C. Related Work

The history of theoretical foundations of Natural Language Processing (NLP) can be tracked back to the 1950s. Since then NLP has been introduced to a number of desktop applications, e.g., for summarization of texts in word processors. Also, NLP has been widely used in scientific applications, e.g., for protein annotation [3], extraction of relationships between cancer-related drugs and genes [4], extraction of clinical conditions and diagnoses from medical records [5], and many others.

However, relatively little research has been conducted on integration of the NLP technology with web-based systems oriented on an average web user. There are relatively few web applications that provide an inbuilt NLP support for their users. OpenCalais[5] is one of the few tools delivering NLP functionality that can be integrated into web systems. OpenCalais is a web service that provides functions for named entity extraction and document categorization. Several web applications leverage this service for helping users in understanding web content. For example, Gnosis[6] is a browser extension for Firefox and Internet Explorer that extracts named entities from web pages and enables users to highlight entities of selected types in the text. Also, OpenCalais is used for assisting web developers in generating semantically annotated content and enriching it with additional information. Tagaroo[7] is one of examples of such application. It is a plugin that adds NLP-support to WordPress blogging and content management system. The plugin suggests semantic tags to blog entries or content pages created with WordPress. Using the generated tags, it also automatically retrieves multimedia content from the web and enables the creator to include it in the text. A similar functionality is provided by Zemanta[8], a browser extension that helps users of leading blogging systems to annotate and enrich their blog entries with supplementary content retrieved from the web. In addition to multimedia content, it inserts links to related articles, Wikipedia pages, product descriptions from major shopping websites, etc.

We have not found any evidence of existing NLP support for portal technology. However, we believe that integration of

[3]Web Services Architecture, http://www.w3.org/TR/ws-arch/
[4]Web Services Description Language (WSDL), http://www.w3.org/TR/wsdl
[5]http://www.opencalais.com/
[6]http://www.opencalais.com/Gnosis
[7]http://tagaroo.opencalais.com/
[8]http://www.zemanta.com/

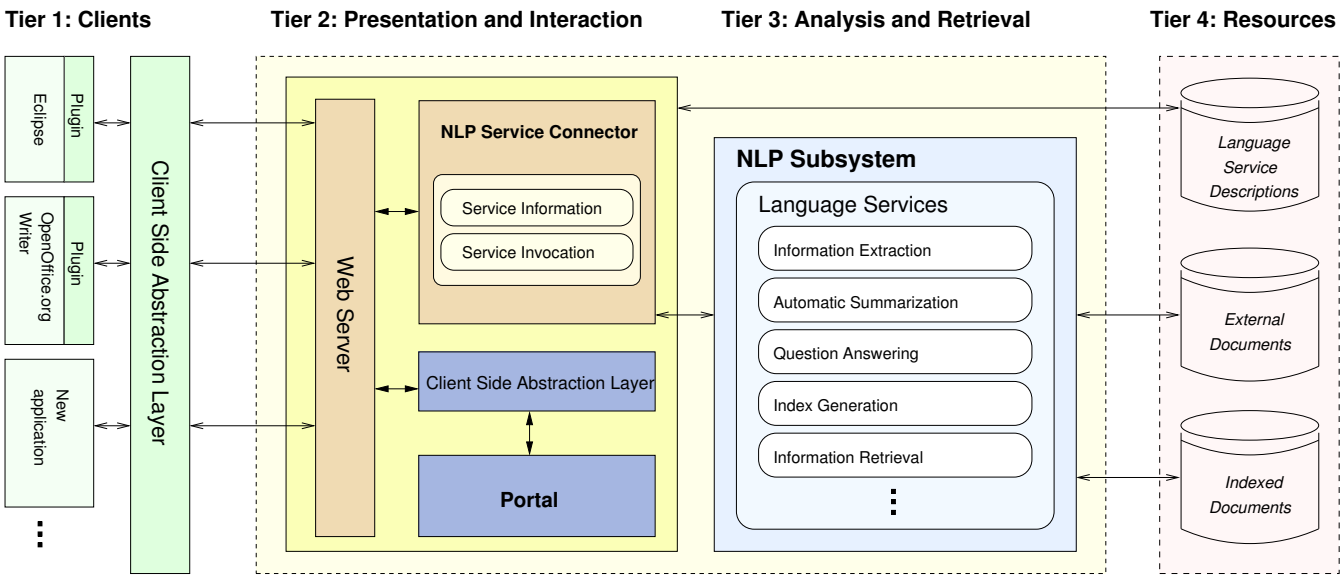

Fig. 2. The Semantic Assistants system architecture

NLP with portal technology can provide portal users numerous benefits. Due to the integration capabilities of portals, NLP support can be provided on a broad spectrum of information and applications delivered through web portals. It includes, but is not limited to, internal documents, news, patents, scientific publications, Enterprise Resource Planning systems, and many others. As we describe in Section IV, portal users can benefit from NLP services in a number of ways.

III. Integrating NLP Services into Web Portals

The application of NLP services in a portal environment differs from their usage in desktop environments, due to a number of portal-specific aspects.

Targeted invocation. A portal page may consist of several portlets, each providing different applications or pieces of content. The user may want to have different types of NLP support for different pieces of information. Moreover, a portal page may contain certain fragments, i.e., navigation menus, header, and footer, which should not be submitted to an NLP service, since they can litter the results. Therefore, it must be possible for the user to request NLP support for individual portlets.

Content preprocessing and results caching. Invoking certain NLP services for a large document or a large set of documents may take considerable amount of time, depending on the complexity of the invoked NLP service. However, an average web user would typically expect the result within a few seconds. If s/he does not get the result within this time, the user may leave the page. Therefore, it must be possible for portlet developers/administrators to configure the portal in such a way that the content of certain portlets is preprocessed with the most popular and/or most time-consuming NLP services. The results of the preprocessing must then be stored in the portal. Also, it must be possible to cache the results of a user-initiated request for each NLP service in the portal and reuse them for subsequent requests with the same invocation parameters.

Dealing with dynamic content. Portlets can display dynamic content that varies depending on the request parameters and session attributes. For example, a portlet for rendering content of a news story selects a document to display based on the document ID, provided either as a request parameter or a user session attribute. Therefore, the method that returns the portlet content for submitting it to an NLP service must take into account all the request parameters and session attributes. They must also be taken into account when caching results of NLP tools in the portal.

User-defined methods for viewing results. Result types of NLP services may vary significantly from one service to another. For instance, a named entity extractor returns a set of named entities found in the submitted document, whereas a summarizer returns a summary in the form of a new text document. To ensure that the user can easily interpret the results, they must be rendered in a suitable way. For example, a set of named entities may be rendered as a sortable list displayed next to the portlet containing the source document, or the entities may be highlighted in the source text. Since it is not always possible to foresee the user's personal preferences and the context of work, the user must be provided with a possibility to select the view method for displaying the results.

A. System Architecture

In our research, we focus on portals that support the Java Portlet Specification JSR286. The solution that we propose in this paper can be applied to any portal server that supports this standard, e.g., IBM WebSphere Portal, Liferay, or GateIn. A JSR286-compliant portal can be integrated with the Semantic Assistants framework by means of the following five extensions (see Fig. 3).

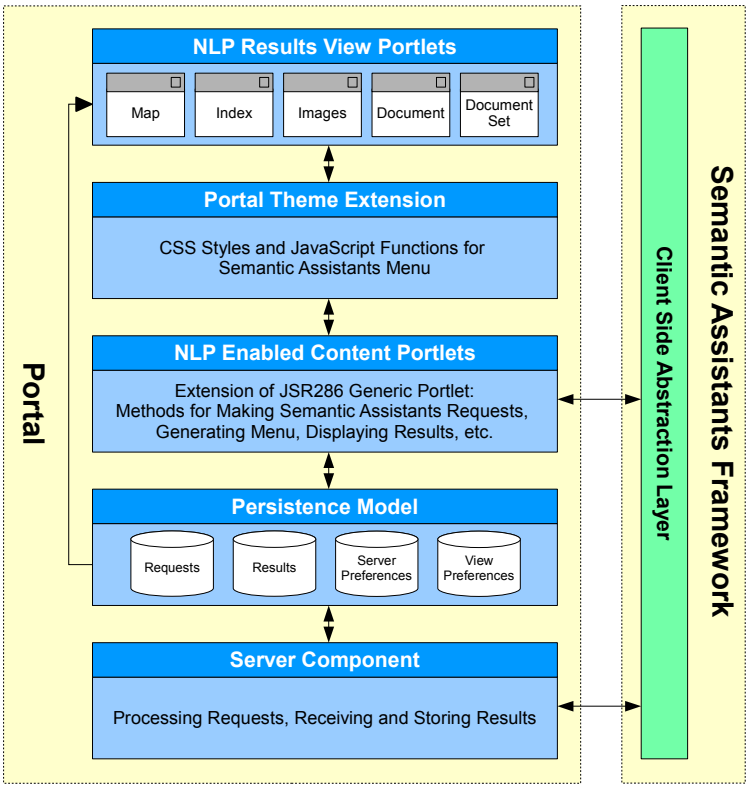

Fig. 3. Web Portal extension for Semantic Assistants NLP services

NLP-Enabled Content Portlet is a Java class that inherits the default functionality of JSR286-compliant portlets and additionally provides methods for interacting with the Semantic Assistants framework. In particular, it adds the *Semantic Assistants* menu (Fig. 6) to the portlet title bar and provides methods for submitting requests to Semantic Assistants and displaying their results. Portlet application programmers can develop portlets with support for Natural Language Processing by simply extending the NLP-Enabled Content Portlet Class, which encapsulates all the necessary methods for interacting with NLP services.

NLP Results View Portlets are intended for rendering results of an NLP service requested for a certain content-providing portlet. In our implementation, we developed the following five view portlets:

- *Map Portlet* is intended for rendering geographic places on a map. It renders entities of geographic location type, e.g., continent, country, or city. It shows the detected geographic names as bubbles on an interactive map[9].
- *Index Portlet* can display a set of annotations as a sortable list. The user can sort the list either alphabetically or by the occurrence of the annotation in the source text. By clicking an item in the list, the portlet will display additional information, i.e., features of the entity, in a popup window.
- *Images Portlet* is designed for displaying images for the named entities found in the text of content providing portlets. For each detected entity, the portlet can display one or several images fetched from the web.[10]
- *Document Portlet* renders text of a single document returned from an NLP service. For instance, it can render a summary generated by a summarizer.
- *Document Set Portlet* displays a set of documents as a list of hyperlinks. By clicking a hyperlink the user will be provided with the content of the corresponding document displayed in a separate window. For example, this portlet can display a set of documents that were retrieved by an NLP service based on the named entities found in the text of the submitted source document.

Portal Theme Extension. The user interaction with the NLP services in the portal is carried out in an asynchronous way, i.e., the user can request NLP support and, once the result becomes available, display it in view portlets without the need to reload the portal page. To achieve that, we extended the portal theme by inserting a set of Cascading Styles Sheets (CSS) and JavaScript functions. In particular, this extension includes styles and functions that provide interactivity for the *Semantic Assistants* menu (Fig. 6) and the JavaScript functions for triggering the methods of the NLP Enabled Content Portlets, e.g., methods for making a request for NLP support, updating the view portlets, or changing the view and server preferences.

Persistence Model is introduced to enable the content

[9]Google APIs are used for the geocoding and maps

[10]Microsoft Bing APIs are used for the image search

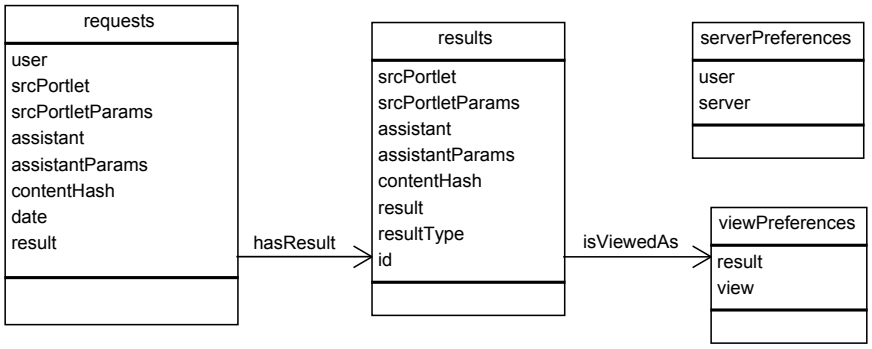

Fig. 4. Persistence model for NLP service results

preprocessing and results caching. It consists of the four following storage classes (Fig. 4):

- *Requests* is the storage of the user-generated requests for NLP support.
- *Results* is the storage of responses generated by NLP services.
- *View Preferences* is the storage of user preferences for the view methods for the result of a certain service.
- *Server Preferences* is the storage of user preferences on the server.

Server Component is responsible for a number of back-end operations, such as sending user requests to the Semantic Assistants framework, receiving and storing the results. The component is essential for enabling the asynchronous interaction with NLP services. It allows that the user, after having made a request on a page, can leave the page and return to it to view the results later. The requests made by users are stored in the *Requests* storage. As soon as a new request appears in the storage, the component first checks if the *Results* storage already contains a suitable result – the component searches for an entry with the invocation parameters identical to the ones of the request, i.e., it searches on five fields: *srcPortlet*, *srcPortletParams*, *assistant*, *assistantParams*, and *contentHash*. If a result entry is found (e.g., another user has previously requested the same assistant on the same content), it writes the result ID to the request entry. If no entry is found, the component sends the request to the Semantic Assistants server and waits for the response. When a result is received, the component stores it in the *Results* storage and writes its ID to the request entry. As soon as the request entry is assigned a result ID, the component notifies the user interface, which informs the user that the result is available and can be viewed.

B. Graphical User Interface

As a proof of concept, we implemented a functional prototype according to the above described system architecture using IBM WebSphere Portal. Fig. 5 displays a portal page with one NLP-enabled content providing portlet and three portlets for viewing NLP-results. The content providing portlet displays the text of a news story. The *Semantic Assistants* button, shown as a battler icon and located on the right side of the portlet title bar, indicates that the user can request NLP support for the portlet content. A mouse click on the button will open the *Semantic Assistants* menu (Fig. 6).

Using this menu, the user can connect to a Semantic Assistants server and view the list of offered assistants. This list is presented as a collapsible menu. In the default state, it displays the name and status of assistants. A click on an assistant name will open a submenu providing the assistant description and a number of controls for invoking the assistant and viewing the results. In order to invoke an assistant, the user needs to set the run parameters, if applicable, and click the *Run Assistant* button. At this moment, the assistant status will be set to *requested*. As soon as the result is available in the portal, either found in the portal cache or newly received from the Semantic Assistants server, the assistant status will be changed to *result available* and its submenu will display options for viewing the result, i.e., a list of view portlets. The list of view portlets is determined based on the result type. For example, the *Information Extractor* assistant, selected in Fig. 6, extracts named entities from the source text and returns them as an annotation set. Results of this type can be rendered in four ways, namely on a map, as images, as an index, or the entities can be highlighted in the source text (Fig. 5). Users can select the view options they like and click the *Display Results* button to view the assistant's outcomes. If the user wants to display results of several assistants in one portlet, the results will be aggregated.

IV. Application Scenarios

In this section, we describe the application of our extension on two concrete scenarios: news analysis and biochemical literature analysis. Both scenarios demonstrate how users can benefit from semantic assistance in web portals provided by tools for NLP and text mining.

A. News Portal

News portals are one of the most wide spread applications of portal technology. They provide one-point access to a broad spectrum of news content from one or multiple media agencies. There are several aspects that strengthen the need for NLP

Fig. 5. News portal with Semantic Assistants NLP services

support in news portals. First, the majority of news portals, especially news aggregating portals, provide an enormous amount of content on various topics. Second, the set of available news stories is updated very frequently, sometimes on an hourly basis. Both aspects may impede users in navigating through the portal and in finding interesting news stories. Also, due to the temporariness of user interest in happenings or situations described by news stories, users tend to scan through stories, instead of thoroughly reading them. They want to grab key points of the story in the shortest possible amount of time. Hence, users need intelligent support, assisting them in finding the most interesting and relevant news stories, and helping them in their analysis.

To address these challenges, we set up a news aggregating portal with integrated NLP support. The portal (Fig. 5) aggregates news stories from multiple leading news content providers and delivers them to users through several category-pages, such as politics, business, or science and technology. Portal users can request NLP support for a set of news stories of a certain category or for a single story. For example, using the *Semantic Assistants* menu (Fig. 6) users can request *Person and Location Extractor* on the portlet listing news stories in the *politics* category. Once the result is available, users can display the extracted person and location names in one or multiple view portlets: in source, map, image, or index portlets. If chosen, the map portlet will display all locations as bubbles on the map. This can help users to get an overview of all locations mentioned in the news stories related to politics. By clicking a bubble on the map, the portal will highlight the news stories containing the place that bubble corresponds to. In this way, users can quickly find the stories of interest. Also, users can request NLP support for a single news story. For instance, they can request the *Information Extractor* assistant, which will return a set of named entities contained in the text of the story, e.g., organizations, people, locations, or dates. Users can also view the extracted entities as an index, which can help them to grab key points of the story and to decide whether it is worth to read it. Additionally, they can request a summary for the story and display it in the *Document Portlet* next to the story. This can also help users to quickly get a rough idea about the happening or situation described in the story.

B. Biochemical Literature Portal

This application scenario was developed within the Genozymes project at Concordia's Centre for Structural and

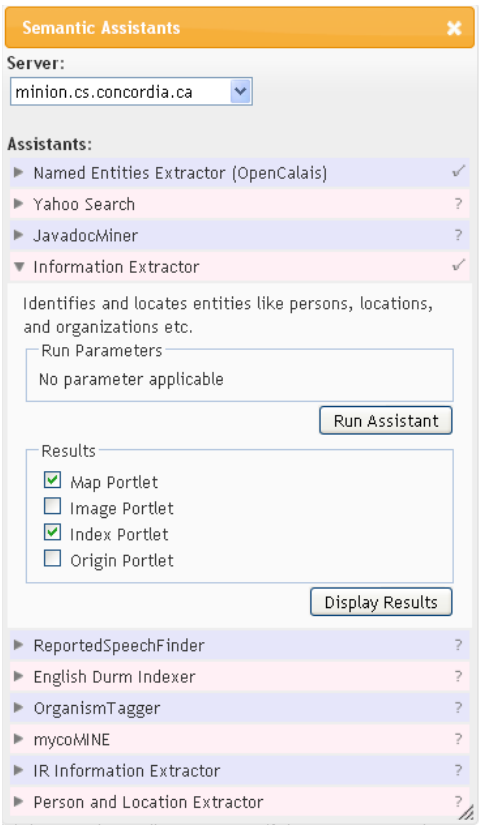

Fig. 6. Semantic Assistants menu for portals

Functional Genomics[11]. We created a web portal for biologists, biochemists and geneticists that work on lignocellulose research. The goal of this research is to find novel ways of creating bioproducts and biofuels from green waste. Part of this work is the curation of characterized glycoside hydrolases[12] of fungal origin from the domain literature. Towards this end, literature from the PubMed[13] portal needs to be evaluated for relevance, which is a time-consuming task. While PubMed allows keyword search, much more advanced semantic support can be provided by domain-specific NLP analysis.

To support these researchers, we automatically import new articles appearing on PubMed into a portal, processing them with the mycoMINE NLP pipeline [6], which extracts entities and facts related to fungal enzymes, such as enzymes, organisms, assays, genes, substrates and pH, temperature or activity assay conditions. Fig. 7 shows an example of queries defined by a user interested in analyzing the genome of the thermophilic fungus *Myceliophthora thermophila*. The *Query* portlet displays the search queries relevant for the user in terms of keywords or sets of keywords s/he has defined. These queries can be hierarchically organized and modified by adding, renaming or deleting keywords. The *Listing* portlet presents the most relevant papers found among new articles appearing on PubMed with regards to all or a selected subset of the user queries. In our example (Fig. 7), the mention of Glutamine Synthetase has been selected in the *Query* portlet and the user has requested mycoMINE on the papers appearing on the *Listing* portlet. The *Index* portlet displays the mycoMINE results in terms of entities and facts mentioned in the papers. These mentions are provided with their associated features. An example of features related to the carbamoyl phosphate synthetase enzyme is shown in the gray pop-up window in Fig. 7. The number of different occurrences is indicated for each type of entities and facts (33 different enzymes have been found in the document set presented in our example). Each reported entity or fact is linked to the corresponding mention in the texts. The mentions are underlined, then highlighted when the user selects the corresponding entity or fact in the *Index* portlet. The user is thereby able to focus on the interesting sections of the relevant papers presented through the portal. In such a way, portal users can apply NLP tools not only on scientific publications, but also on variety of other resource and applications that can be aggregated using portal technology, such as patents, databases of genes and drugs, samples, or observation and sensor data.

V. Conclusion and Future Work

In this paper, we presented a novel technical solution and interaction patterns for the integration of NLP tools with portal technology, in order to provide semantic assistance to users of web portals. Our approach allows users to benefit from a broad spectrum of various NLP services, such as named entity extractors, summarizers, indexers, and others. Users can use these services on a multitude of content and applications delivered through modern portals. We have illustrated this using two application scenarios, namely, a news aggregating portal and a portal for biochemical literature. We have showed how this integration can help users to find interesting and relevant news stories and to quickly grab key points from the text of individual stories. Also, we have illustrated how NLP support in a biochemical literature portal can assist scientists in accomplishing their information seeking tasks.

Currently, we are working on enhancing NLP support in portals by introducing personalization features using our previous work on semantic user modeling [7], [8] and personalization of portal resources [9]. We believe that portal users can benefit more from this support if the results of NLP tools are tailored to individual users, i.e., to their interests, background, and expertise. We are working on methods for automatic generation of ontology-based user models using the semantic output of NLP tools. Also, we are developing methods for adapting the NLP results based on the user model. These methods include but are not limited to filtering, sorting, and highlighting extracted entities according to user interests.

Finally, we are currently evaluating the effectiveness of this integration using the biochemical literature portal described in Section IV. The portal is being used by biologists, biochemists, and geneticists from the Genozymes project at Concordia Centre for Structural and Functional Genomics. In the near

[11]CSFG, http://genomics.concordia.ca/
[12]family of enzymes used to break down plant cell walls
[13]PubMed http://www.ncbi.nlm.nih.gov/pubmed/

Fig. 7. Biochemical literature portal with Semantic Assistants NLP services

future, we will analyze the portal usage data, interview users, and ask them to evaluate the system using the USE questionnaire [10], which takes into account four usability aspects, namely, usefulness, satisfaction, ease of use, and ease of learning.

Acknowledgements. Part of this work has been funded by Genome Canada and Génome Québec. The work on the integration of portal technology with NLP was sponsored by the IBM Ph.D. Fellowship Awards Program and carried out in the framework of the Minerva Portals Project in cooperation IBM Deutschland Research & Development GmbH.

References

[1] R. Witte and T. Gitzinger, "Semantic Assistants – User-Centric Natural Language Processing Services for Desktop Clients," in *3rd Asian Semantic Web Conference (ASWC 2008)*, ser. LNCS, vol. 5367. Bangkok, Thailand: Springer, 2008, pp. 360–374. [Online]. Available: http://rene-witte.net/semantic-assistants-aswc08

[2] H. Cunningham, D. Maynard, K. Bontcheva, V. Tablan, N. Aswani, I. Roberts, G. Gorrell, A. Funk, A. Roberts, D. Damljanovic, T. Heitz, M. A. Greenwood, H. Saggion, J. Petrak, Y. Li, and W. Peters, *Text Processing with GATE (Version 6)*. University of Sheffield, Department of Computer Science, 2011. [Online]. Available: http://tinyurl.com/gatebook

[3] H. Xie, A. Wasserman, Z. Levine, A. Novik, V. Grebinskiy, A. Shoshan, and L. Mintz, "Large-scale protein annotation through gene ontology," *Genome Research*, vol. 12, no. 5, p. 785794, 2002.

[4] T. Rindflesch, L. Tanabe, J. Weinstein, and L. Hunter, "EDGAR: extraction of drugs, genes and relations from the biomedical literature," in *Pacific Symposium on Biocomputing*, 2000, pp. 517–28.

[5] H. Ware, C. Mullett, and V. Jagannathan, "Natural language processing framework to assess clinical conditions," *Journal of the American Medical Informatics Association*, vol. 16, no. 4, pp. 585–589, 2009.

[6] M.-J. Meurs, C. Murphy, I. Morgenstern, G. Butler, J. Powlowski, A. Tsang, and R. Witte, "Semantic text mining support for lignocellulose research," *BMC Medical Informatics and Decision Making, Vol 12 Suppl 1*, 2012. [Online]. Available: http://www.biomedcentral.com/1472-6947/12/S1/S5

[7] 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. [Online]. Available: http://www.minerva-portals.de/publications/refereed-publications/a-hybrid-approach-to-identifying-user

[8] ——, "Introspectiveviews: An interface for scrutinizing semantic user models," in *Proc. of the 18th Int. Conf. on User Modeling, Adaptation, and Personalization*, 2010. [Online]. Available: http://www.minerva-portals.de/publications/refereed-publications/introspectiveviews-an-interface-for-scrutinizing

[9] A. Nauerz, F. Bakalov, B. König-Ries, and M. Welsch, "Personalized recommendation of related content based on automatic metadata extraction," in *Proc. of the 18th Int. Conf. on Computer Science and Software Engineering*, 2008. [Online]. Available: http://www.minerva-portals.de/publications/refereed-publications/personalized-recommendation-of-related-content

[10] A. Lund, "Measuring usability with the USE questionnaire," *Usability Interface*, vol. 8, no. 2, 2001.