Application of Semantic Search in Idea Management Systems

Geovanny Poveda
Departamento de Sistemas Telemáticos
Universidad Politécnica de Madrid
Madrid, Spain
gpoveda@dit.upm.es

Adam Westerski
Departamento de Sistemas Telemáticos
Universidad Politécnica de Madrid
Madrid, Spain
westerski@dit.upm.es

Carlos A. Iglesias
Departamento de Sistemas Telemáticos
Universidad Politécnica de Madrid
Madrid, Spain
cif@dit.upm.es

Abstract— **The following paper describes the design, architecture and use of a semantic search model in open innovation support systems. We explore the relationships between the user submitted content to improve interaction and simplify the current schemes of analysis of Idea Management System data. In order to accomplish this, we propose a model where users can build their own scenarios for business analysis. In particular, we present a methodology for collecting, organization and search of ideas submitted in a number of distributed public brainstorming systems. The description of our model is accompanied with a set of use cases**

Keywords-component; semantic; search; RDF; ontology; idea managament

I. INTRODUCTION

During recent years, Idea Management Systems (IMS) have become widely used in many organizations and have expanded their area of applications [1]. Among those, IMSes are used as a **"common base of ideas"** that centralizes the management of innovation regardless of the source that it originates from (the organization itself, its partners or clients). While Idea Management Systems enable to collect the ideas, their ultimate goal is to enable the organizations to manage the discovery, incubation, acceleration, and scaling of ideas to create commercial value through the development of innovative products and processes.

Idea Management Systems address many challenges in the innovation process. One of those is the need to provide users with functions that allow can find information from the robust collection of ideas. Currently, the search functions implemented by the great majority of IMSes are based on regular search engines, in which the returned information often contains results not related to the search context due to queries being made using traditional keyboard based approach.

In order to improve information search in IMS, we propose to adapt semantic search mechanisms. In contrast to traditional keyboard based search, semantic search allows users to execute more contextualized queries, allowing them to get more specific information about ideas, related comments, ratings, reviews, etc.

Using a mechanism based on Semantic Web technologies [2], we propose a model where users of Idea Management Systems can retrieve information based on contextual information and data relationships. Our model has been defined as a layer on top of the Semantic Idea Management System proposal by Westerski et al. [3]. In particular, we utilized the existing ontology and methodologies for interlinking Idea Management Systems and extend the vision presented in Gi2MO project [4] to add semantic search functionalities that take advantage of the rich metadata layer.

This paper is organized as follows: section 2 presents the basic concepts about Gi2MO IdeasStream project; section 3 presents a case study to show the potential benefits of semantic search process in IMS. Next, section 4 introduces the solution architecture of the Generic Idea and Innovation Search Engine (Gi2SE) module describing the most important components. In section 5 we present related work in the area of semantic search engines and describe the main characteristics of similar contributions. In section 6 we discuss the obtained results as part of validation of our proposal and the experimentation process. Finally, section 7 summarizes the results and points out the possible future research directions.

II. BACKGROUND

Gi2MO IdeasStream is an open-source Idea Management System based on the component architecture of a Content Management System called Drupal. The main objective of IdeaStream is to offer a test bed platform [5] for experimentation of projects related to ideas management concept. IdeaStream is extended with an ontology called Gi2MO which aims to clip all the phases of idea management process together. Gi2MO ontology schema was designed to maintain the integrity with Semantic Web trends and standards yet keep the ontology simple and put impact on its usability and ease to appliance to encourage other developers. Gi2MO ontology scheme has been modeled using the following concepts:

- Dependencies with internal assets (other) and external assets (ideas from another system or other media resources) using existing ontologies such as dcterms, scot and foaf.

- Deployment metrics (most often business metrics such as return of investment, total cost, etc).

- Idea improvement data (comments, users, ratings, idea version, etc).

In order to implement the aforementioned elements in practice, Gi2MO platform delivers a module called RDFme [6]. Using RDFme users can import mapping rules to allow portability and quick migration between different systems. RDFme plugin delivers a variety of additional futures useful for publishing and consuming RDF data:

- **RDF data import:** the data that has been exported in RDF following the defined mappings can be also imported back into the system (this means that a number of distributed Idea Management Instances can be connected together). This functionality is enabled both via UI and as a REST service.

- **RDF data expose:** the data that has been mapped on Gi2MO platform is exposed via REST service, thus users can get all ideas in a single RDF resource accessible via a HTTP URL.

Usability and interoperability are the most important reasons why we chosen to Gi2MO IdeaStream to do our work. The easy integration with other Semantic Web modules and the connectivity facilities with external ontologies are good references to use such platform and extend its functionalities by incorporating new modules to support semantic search process. Using the Gi2MO IdeaStream as a base, we intend to improve the search problems of the traditional IMSes.

In the next chapter we show a use case in which we highlight the importance to incorporate semantic search functions inside Idea Management System. The primary goal of the presented example is to expose the benefits of semantic search for business analysis processes that use ideas management platforms.

III. USE CASE STUDY

John is working in a project for the department of innovation of ACME Company. In such a project the company needs to implement a solution to measure the degree of satisfaction of the clients by analyzing comments and suggestions made by users with respect to quality of product offering of the company. Few months ago, the company has implemented an instance of Gi2MO IdeaStream. Using such platform, business leaders have obtained the feedback from the clients and have taken decisions based on the information processed by external analytical tools (e.g. such as Pentaho). After using the system for a long time, John discovers that feedback information has grown so much that nobody is capable to analyze within reasonable time and effort costs the comments and suggestions stored in the Idea Management System. At the same time, John discovers that the company staff spends a lot of time during this idea revision process because of analyzing which data they need to export for the analytical tools.

Despite the fact that the Idea Management System provides a valuable supplement to the innovation management process, the analysis process of information collected has to be re-evaluated and supported by means of tools that decrease the effort of analysis for business leaders and decision makers. Therefore, John turns for help to emerging technologies and convinces his company management to invest in integration of system with semantic search module.

Using the new semantic search module, John ’ s company could improve analysis of information:

- John can see that search process provides updated information, because when he executes a same query in different moments, he gets different results. The last result always contains more information. When a query is made, Gi2SE module enables a function in which frequently check whether new ideas, comments or suggest has been made from Ideas Management System or from the external site used during the extracting process.

- John can see that new search utility provides quick responses. In spite of Gi2SE using information from external sites, the new module allows the users to save collected information locally. Such type of operations improves the response times and allows the resources to be always available. When a user run a query, the search process does not depend on response time of external services neither the availability of external resources.

- Using Gi2SE module, John can choose the output format of queries his queries. He can view the results using faceted browsing or normal web search results view (e.g. such as in Google). Therefore, John discovers that results can be interpreted in a better way than in comparison to raw SPARQL query result lists based on serializations such as RDF/JSON, RDF/XML.

- Using Gi2SE module, John can execute complex queries without the need to use external analytical tools. Queries like: ” Comments that have ” **great word** ” and whose author has a number of posts greater than 10 and whose author has a rating value greater than 10 and whose posts have a number of reviews greater than 15 that include the words ” valuable” or ” absolutely helpful” ”

IV. SOLUTION ARCHITECTURE

To enable the presented use cases, we have designed and implemented a module called Gi2SE which provide search functionalities for the Gi2MO platform. The solution architecture of our approach is presented on Figure 1. In order to adapt Gi2SE to Gi2MO platform, we have used the plug-in mechanism defined by the Drupal system. As we mentioned above, the semantic matching, depends on data extraction and mapping process. In order to do such activity, we downloaded and parsed web content from Ubuntu Brainstorm website [7] to generate a RDF file that contains ideas, comments, suggests and ratings scrapped from the Ubuntu website. The collected information was put into the Gi2MO IdeaStream system using data import utility of RDFme module. Afterwards, we used RDFme again to expose the mapped concepts in single RDF file. From now, we will refer such file as RDF mapped. Next section provides detailed information about Gi2SE functions.

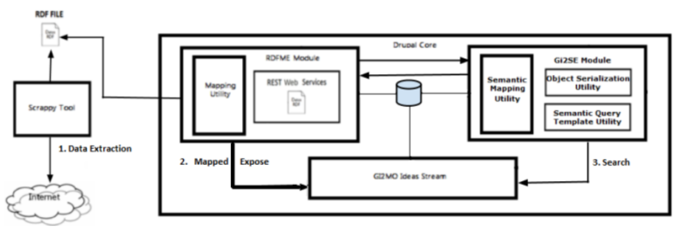


Figure. 1. Solution Architecture.

A. *Providing Configuration Options*

In order to adapt Gi2SE to requirements of Gi2MO IMS platform, we have implemented a function where users can configure different options related to the search process. We have considered the following configurable settings:

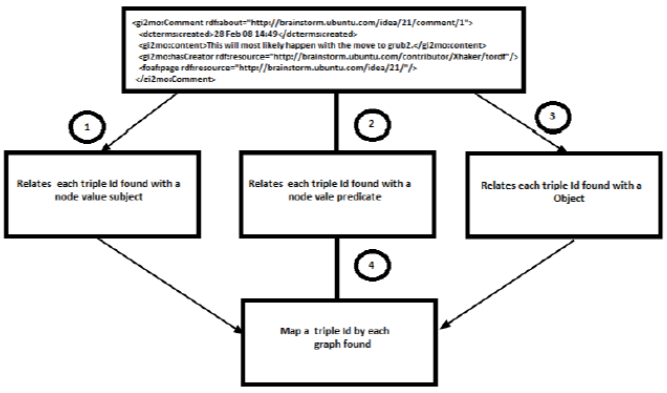
- Create, update, delete and select URL of the RDF resources used during search (Data Source).
- Specify the URL of SPARQL Endpoint (required).
- Specify graph name to filter information contained in the added data sources (optional).

The “update” option is one the most important features of our work that impact further user interaction during the search process. With this option the data is indexed for later queries as the most recent concepts are added by external users on Ubuntu Brainstorm Website - ideas, comments, suggest, ratings etc. Update activity is made using an algorithm that checks the modification date as well as size of the RDF mapped file. After to verify an update, the algorithm compares one to one the concepts and selects only the new information. On the other hand, we have implemented a utility where users can configure different SPARQL URL Endpoints. Implementing such an option we wanted to lay the foundations for the future to extend our domain search (ideas) to other areas.

B. *Mapping Process*

Another important feature of Gi2SE is the proposed model to query and save the data attached into the Gi2MO platform. In order to do this, we have used the triple store concept in conjunction with the mapping notion [8]. Using those concepts, we designed an algorithm to map the entire RDF mapped file into the relational database. The reasons to use triple stores in conjunction with relational database are: i) relational database are mature and widely used. ii) triple store provides more flexibility [9] (e.g. users can add new predicates on the fly without the need to make schemas beforehand). iii) triple store produces better performance when it comes to complicated queries, given this robustness and usability.

Taking into account those reasons, we propose a model in which a triple store enables smart integration with the relational database by adding intelligent metadata on top of a database engine. Our objective is to make the most of both: objects explicitly modeled (RDF triples) and relational database system. In order to achieve this, our model proposes a mapping process in which each RDF property belonging to mapped file is translated into a database entity. Figure 2 shows us the workflow inside the mapping process on search engine.



```
<gi2mo:Comment rdf:about="http://brainstorm.ubuntu.com/idea/21/comment/1">
  <dcterms:created>28 Feb 08 14:49</dcterms:created>
  <gi2mo:content>This will most likely happen with the move to grub2.</gi2mo:content>
  <gi2mo:hasCreator rdf:resource="http://brainstorm.ubuntu.com/contributor/Xhaker/tordf"/>
  <foaf:page rdf:resource="http://brainstorm.ubuntu.com/idea/21/"/>
</gi2mo:Comment>
```

1
2
3
Relates each triple Id found with a node value subject
Relates each triple Id found with a node vale predicate
Relates each triple Id found with a Object
4
Map a triple Id by each graph found

Figure 2. General Mapping Workflow in Gi2MO platform.

Each time that a user defines a URL of an RDF resource in the configuration options, Gi2SE executes a set of operations in which a database system relates a Id number field with a set of fields that contains information related to RDF properties. In order to do such an activity, we used an algorithm to create separate tables for each entity -subject, predicate, and object - found. Such approach will allow the engine to make multiple combinations of subject, predicate and objects easily. Using relational tables we want to represent as data the relation between each field and each object, in such a way that each subject and predicate can have an identifier that is not just an integer number that means only something inside the database. This identifier is a URI that may have a distinct meaning. Mapping process used in this architecture involves the following entities:

Graph - Such entity has been modeled to create a referential value (Graph Id) for each RDF resource used during mapping process. On the other hand, mapping process creates another entity called **Subject** in which how the engine relates an Id number with the *node value* according to the number of triples found. That is, each triple found during mapping process can be identified by a unique number.

At the same time, mapping process creates another entity called Predicate to specify only one Id for a group properties found on a namespace. Thus, each property such as title, content or creator is identified by a general Id.

After identifying subjects and predicates, our algorithm creates another entity to store all element objects found during mapping process. In Object entity the engine relates an Id number for each object element found in a predicate. In this way, on this entity it is possible to see information related to Ideas content, title content, comment content, etc. The end step in the mapping activity is the creation of one entity in which system joins all attributes belonging to all modeled entities as tables subject, predicate and object.

C. Object Serialization Query

Taking into account that our search process involves both: SQL and RDF technologies, one of the most important decisions that we took during the design phase of our plug-in was to propose a method where queries could combine the RDF properties with relational database technologies. In order to provide a mechanism that implements this, our plug-in has extended any functionalities of ARC2 framework. Using scripts based on the method chaining concept and SPARQL+ technologies [10] our search engine can query an object processing and integrate data from multiple sources in a single operation. When we say a single operation we are referring to a PHP sentence that uses SPARQL operations as parameters to instantiate SQL operations. Such methods are defined as intermediate entities between SPARQL sentences and SQL operations. Thus, SPARQL operations such as LOAD, INSERT and SELECT can be interpreted by SQL language. At this point of the explanation, we have to remark that we use a serialization process in which we transforming and storing the current state of a RDF object on a database system.

D. Semantic Search Query Template

In order to provide a utility in which users can easily execute search processes on Gi2MO platform, we have designed a graphical interface in which users can easily specify how many and what type of classes and attributes they will use during the search process. In contrast to other semantic search frameworks where users can type any kind of information on search boxes, our solution proposes the design of a framework in which system guides the users during the search process, providing an auto assisted interface. Designing an auto assisted graphical interface experience; we want to avoid the search engine to process incoherent information. Taking this into account, the Gi2SE supports a search processes related to information analysis; we want to guarantee search processes with a high degree of precision. For this reason we decided to built an interface in which users can minimize the risk to type incoherent information, because the values typed by each user are associated automatically to specific properties and classes.

The design our interface has followed a model in which search entities are related according to the following structure: Class -> Property -> value of property. That is, each time a user creates one class in the search interface, then the engine automatically creates a default property and suggested property value. Thus, search engine forces the users to create triples of information as input information for search process. Another important feature in our module is the possibility to link as many triples as possible in dynamic way. That is, users can do more specific queries creating as many classes as possible.

In order to accomplish this, we created a semantic query template to process all information typed by users during the search process. Such template uses an algorithm that checks the classes, attributes and attributes values chosen by users to build automatic SPARQL sentences. This algorithm has been built following the basic structure of a SPARQL query [11]. We have considered all possible section options of a SPARQL query, except the query result selection option (LIMIT n and OFFSET m) and query data set source option – often used to add triples to the graphs - .

Our algorithm generates SPARQL sentences from the information typed by users and a series of methods that we have encoded for representing standard SPARQL operation. Aforementioned methods have been designed to implement specific functionalities for each SPARQL section. Thus, to build the prologue section our algorithm use a method that instantiates the name spaces to be used as prefix in the SPARQL from database system. Such selection is made according to the properties selected by users when they choose the terms from the form interfaces. This approach allows the system to load only the necessary name spaces for the query. After loading the name spaces, the method delegates control to a function in charge of building a SPARQL object.

Another important feature is related to query results *form* section. In such activity a method checks the RDF properties selected by users and transforms them as SPARQL SELECT parameters. Optionally, in this section it is possible to use the count operator. When this option is activated, Gi2SE users can make queries like:"Number of ideas that has a rating value greater than 10 and whose author has a rating value greater than 100 and the content of idea have crash Gnome". Count operator is associated to result form section only if user has chosen such option in the graphical interface. When users mark this option, our script identifies which properties have been selected and relate them with count sentences following a structure like: {? parameter (0), parameter (1) .... parameter (n) (count (? parameter (n) as? count))}.

One important feature related to query design template is when a user tries to make a simple query with only one count value, in such case, the system only retrieve one property - only one simple data number-. In general, this would be the normal response. However, users need to get more detailed response, in which system not only returns a single text; users need to get additional information such as date, title, link or description. In order to solve this, our algorithm implements an activity for automatic generation of relation for simple queries, building internal relationships between the specified parameter and the properties close to such term.

Another important feature of our algorithm is the automatic generation of UNION operators by each triple put as input information. Based on the number of classes and properties used by the users in the graphical interface, our algorithm automatically links such values by means of UNION operators. In general, our algorithm creates sentences with the following structure: WHERE {Graph Pattern (1) U Graph Pattern (2) U .... Graph Pattern (n) }. These kinds of operations are instanced when users create complex queries in which two o more conditional sentences are used.

E. Output Result Serialization

An important feature of Gi2SE is generation of the output results of queries in the requested format. Many triple store frameworks such as Sesame, Jena and ARC2, include serialization functions to generate multiples output formats such as XML, RDF, JSON and NTRIPLES. However, such serializers have been designed to return information as plain text on normal SPARQL console.

Taking into account that our module is deployed as a plug-in for the Gi2MO platform (Drupal system), we built a method in which responses are transformed into HTML structures. Using render functions, our module provides results similar to Google search engine, following a structure like: **linkable title -> date -> short description**. In order to make this possible, our algorithm takes the returned query object and serializes it into HTML sentences. When this action is made, our algorithm uses an **array map** function to store all property values found into an array structure. By storing the information into arrays we want to classify the result in different categories, in such a way that our method can build HTML structures related and modularized. That is, our method builds different HTML sections (title, date and description) keeping the original RDF graph relations. Thus, when users click on a linkable title, search engine shows in more detail results associated it.

At the same time, we have also designed an algorithm for a second result visualization. In this scenario, the output format is showed in a manner facilitating faceted browsing [12]. Using SIMILE tool our algorithm takes the mapped RDF and serializes it into a exhibit-json file. In order to make such process, Gi2SE has used a utility which consumes the Babel RESTFUL Web Services [13] using as parameter the mapped RDF file, which contains all triples extracted from the Gi2MO IMS platform. When Babel service generates the output file as a response - exhibit-json - our method downloads such file and uses it to build a script that provides filtering, sorting and searching capabilities in the faceted browsing interface. Figure 3 show us the entire semantic search process on Gi2MO platform.

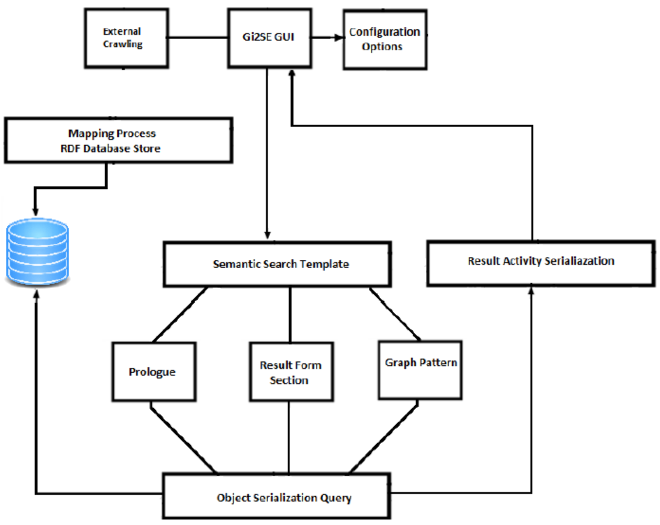


Figure. 3. Gi2SE General Workflow Semantic Search Process.

V. Semantic Search Evaluation

In our research, we have identified that the two most critical problems in implementing the semantic search engines in IMSes are: i) crawl - indexing and ii) query 's response time. In order to evaluate solution to the first problem, we referenced the crawl and index process used during deployment activity of Gi2SE. On the other hand, for evaluating query 's response time we have referenced the internal mapping process used during deployment activity. Thus, we divided the evaluation process into two separate phases. Firstly, we wanted to measure the time spent when users run the extraction and data integration process using data from Ubuntu Brainstorm website. Secondly, we wanted to evaluate the integrity and interoperability of our search engine, measuring the time spent during mapping process between RDF properties and SQL entities.

A. Crawler and Mapping Evaluation

For the first evaluation task we used the data available from Ubuntu Brainstorm website. Using a crawler process we got 20152 ideas and additional data which contains comments, reviews, idea statuses and other related metadata. After that, we ran an automatic mapping process using web service technology to import the data into Gi2MO IMS. In order to demonstrate the effectiveness of crawler and mapping process during the deployment activity, we repeated the experiment three times for different number of ideas (100,200, 500). The results regarding time spent by the algorithm during those experiments are presented in table 1.

Number of Ideas	Number of Triples	Crawler time (s)	Mapping time (s)
100	1795	614,67	892,16
200	13760	5582,12	4578,36
500	7843	4189,40	4093,61

Table. 1. Time spent by different number ideas on Gi2MO platform.

Analyzing the above results we can make one interesting observation: Running all test, we could see that amount of ideas does not always has a direct impact on time spent to process it. That is, the time spent depends of number of connections associated by each property, especially on number of comments made by the user. The most important part of this experiment was associated to third test, where we got 500 ideas - all information entered by Ubuntu Brainstorm users in a time greater than 2 months - in about 30 minutes. At the same time, we note that we managed to import the same quantity of ideas - 7843 triples - into Gi2MO in about 40 minutes. Crawler and our mapping ontology processes are a good alternative to build easily and quickly a triple store from an external website. This kind of actions allow the business analyst users to create their own triple stores, avoiding spending too much money and time collecting external information manually.

B. *Queries Response Time Evaluation*

Another important feature that we wanted to test was the time spent during internal mapping activity between RDF properties and SQL entities. In order to demonstrate the effectiveness of such mapping process during the search activity, we repeat the search activity three times for a different number of ideas (100,200,500). The times spent by the algorithm to return the results for the aforementioned experiments are presented in table 2.

Number of ideas	Number of triples	Number triples added	Additional spent time (s)
100	1795	10	57,21
200	13760	25	126,13
500	7843	34	180,08

Table. 2. Time spent by different number ideas on Gi2MO platform.

In the second and third experiments we included two modifications: i) in order to test interoperability concept of Gi2MO ontology we included the ideas and comments made directly by users in the Gi2MO IdeaStream platform ii) in order to test the integrity of our search engine, we executed an auto update process using the most recent ideas submitted by Brainstorm Ubuntu users. The most important part our second experiment was related to third test, in which we included 34 new triples generated from Ubuntu Brainstorm ideas website. In such process, our algorithm detected in only 30 seconds the new ideas and mapped it in about 2 minutes.

Finally, after executing extraction and the mapping tests, we decided to measure total time spent for a executing a query. Particularly, we executed the following query : ” Ideas that has WIFI word and whose proposed solutions contain some of these words: use |change |add |remove and whose comments were made by user with a rating greater than 10” Table 3 shows time spent for scenarios when a different amount of ideas has to be processed.

Number Ideas Used	Spent time (s)
100	112
200	442
500	356

Table. 3. Time spent by different number ideas on Gi2MO platform.

According to information showed in table 3, we can say that time spent during the query does not depend on quantity of information that algorithm needs to process. It depends on the graph model querying implemented in the algorithm (node search). At the same time, the results obtained in the last experiment allow us to discover the importance of including Semantic Web capabilities for Idea Management Systems. When we used such an approach, the results were not only text that contains a lot of unneeded information. The user could interact dynamically with the result list and adjust the amount of information that we wanted to obtain. Using the capabilities of semantic relationships established by the usage of Gi2MO ontology, we have shown how information contained in an Idea Management System can be exploited using tools such as semantic search.

On the other hand, times obtained during the last experiment show us that triple store based on relational database is a good option to build a search engine. Using such approach we were able to take advantage both: extensibility on RDF graph and robustness on SQL queries. Last value obtained in the last experiment shows how users can get important information for example from competitors in a little time. Thus, we wanted to contribute with a tool that can be used in analytical business process.

VI. Related Work

In the paper we discuss the lack of semantic search mechanism in IMS to find useful information in the collected ideas and propose a solution by establishing a semantic search engine. While to our particular solution for semantic search has been tested in context of Idea Management System there is a number of different approaches that relate to our work in other contexts such as Enterprise Content Management (ECM), Knowledge Management (KM) and regular Content Management Systems (CMS). However, since Idea Management Systems are a rising technology there has not been much research done in terms of semantic search tools specifically for them.

To our knowledge Penela et al [14] are the only ones who present a similar attempt to ours. However, their solution applies a different approach; they use the microbloging paradigm combined with a special way of semantic indexing. In such solution, authors propose a framework in which users have interaction with the engine around a user interface with a single input option. Using a single input parameter, the content of the message itself is used to query a status repository that contains the message that has been indexed when users post new messages into the system. Furthermore their solution is deployed as a standalone services with no connections with other environments. The aforementioned characteristic makes the model implemented by this framework needs to adapt with the particular knowledge of each company, because semantic capabilities are provided by an ontology defined on a generic domain.

In relation to our primary goal, semantic search in Idea Management Systems, the most interesting work, from our point of view, are by Boye at el [15] who tries to incorporate semantic functions on an Enterprise Content Management called Alfresco. In this model authors have defined a semantic layer on top Alfresco System using tools such as Jena Framework and Apache Stanbol. The semantic search model proposed in this tool has been defined according to the following concepts: i) ontology and taxonomy searching ii) data relationships and iii) external data source Integration. However, application domain is related to content administration such as documents and digital assets. At the same time, we consider this kind of models are not representing a good base for semantic search models related to business analysis process because dependencies with external asset including data sources outside the CMS content and businesses model. In particular, this model keeps relations with external data sources such as DBpedia. Additionally, both of those aforementioned attempts do not refer to Semantic Web technologies in such extent as our research.

VII. Conclusions

In the paper we have shown how Gi2SE can support and improve search process in companies that use an Idea Management System. Using Semantic Web principles, we proposed a semantic search architecture in which we got similar results as if we were using external analytical tools but with greater savings in terms of time and effort that is wasted on manual information search and cleaning result lists based on keyword search. However, we presented only an approximation to real semantic business intelligence scenario, because the data contained in an IMS needs to be interpreted in a best way after executing the semantic search processes. Therefore, in terms of future work, we envision to implement a module in which the obtained information during the search process, can be represented using clustering techniques. With such an idea, we think that users could interpret in a best way the information gathered in the enterprise. On the other hand, we will continue improving the basic functionalities related to semantic search engine (Gi2SE). For that, we would like to propose a new functionality in which search process does not only support information sources coming from the Idea Management domain. The objective is to support any RDF source that has coherent structure with previous mapping process in any ontology domain.

References

[1] Rozwell, C.: Ideas Management On Gartner’s Hype Cycle 2011 (2011), http://www.cometoknow.com/idea-management-on-gartners-hypecycle-2011

[2] T. Berners-Lee, J. Hendler, and O. Lassila. Semantic web. Scientic American, May 2000

[3] Westerski, A.: A Model for Integration and Interlinking of Idea Management Systems. In: 4th Metadata and Semantics Research Conference (MTSR 2010), Alcala de Henares, Spain, 2010

[4] Gi2mo project homepage, http://www.gi2mo.org/

[5] Galan, F., Fernandez, D., Lopez de Vergara, J.E., Casellas, R., ” Using a model-driven architecture for technology-independent scenario configuration in networking testbeds ” , Communications Magazine, IEEE, vol.48, no.12, pp.132-141, December 2010

[6] RDFME module project homepage, http://www.gi2mo.org/apps/drupalrdfme-plugin/

[7] Ubuntu brainstorm homepage, http://brainstorm.ubuntu.com/

[8] Sint, Rolf, Schaffert, Sebastian and Stroka, Stephanie. Combining Unstructured, Fully Structured and Semi-Structured Information in SemanticWikis. Heraklion, Greece : 4th Workshop on Semantic Wikis, June 2009.

[9] Andreas Brodt, Oliver Schiller, and Bernhard Mitschang. 2011. Efficient resource attribute retrieval in RDF triple stores. In Proceedings of the 20th ACM international conference on Information and knowledge management (CIKM ’ 11), Betting Berendt, Arjen de Vries, Wenfei Fan, Craig Macdonald, Iadh Ounis, and Ian Ruthven (Eds.). ACM, New York, NY, USA, 1445-1454.

[10] Benjamin Nowack: SPARQL+, SPARQLScript, SPARQL Result Templates - SPARQL Extensions for the Mashup Developer. International Semantic Web Conference (Posters Demos) 2008

[11] Martin G. Skjveland: Sgvizler: A JavaScript Wrapper for Easy Visualization of SPARQL Result Sets. 9th Extended Semantic Web Conference (ESWC2012) 2012

[12] Erling, O., Mikhailov, I. Faceted Views over Large-Scale Linked Data. Linked Data on the Web (LDOW2009). http://events.linkeddata.org/ldow2009/

[13] Babel service project homepage, http://service.simile-widgets.org/babel/

[14] Penela, V., Ruiz, C., Cordoba, C., Carbone, F., Gomez, J.: miKrow: Semantic Intra-enterprise Micro-Knowledge Management System In: Simperl, E., Parsia, B., Grobelnik, M., Antoniou, G. (eds.) lESWC2011. LNCS, vol. 6644, pp. 154168. Springer, Heidelberg (2011)

[15] J. Boye Conference Zaizi Semantic Search Tool. Aarhus November 08 2011.

[16] Hamid, S.: Semantic Search in Linked Data: Opportunities and Challenges, Proceedings of the 24th AAAI Conference on Artificial Intelligence (AAAI10), Atlanta, Georgia, USA, July 11-15, 2010

[17] A. Harth, A. Hogan, R. Delbru, J. Umbrich, S. ORiain, and S. Decker. SWSE: Answers before links! In Proc. SemanticWeb Challenge 2007, CEUR Workshop Proceedings 295. CEUR-WS.org, 2007