Master's Thesis

Semantic Enrichment of Content Management Systems: An Application on Joomla! CMS

in Partial Fulfillment of the Requirements for the Degree

Master of Science (Dual Award)

presented to the Department of Mathematics, Natural and Information Sciences at the Technische Hochschule Mittelhessen and Ege University

by

Umutcan Simsek

Giessen, September 2015

Thesis advisor:	Prof. Dr. Peter Kneisel
Co-advisor:	Assoc. Prof. Dr. Riza Cenk Erdur (Ege University)
Second examiner:	Dennis Priefer, MSc.

There is a significant amount of knowledge hidden in web content management systems (WCMS) in both structured and unstructured data. This work aims to suggest an approach to allow non standardized web content managements systems to communicate with semantic engines, thus those data and moreover the hidden knowledge can be published in a structured way and become more interoperable with the rest of the web.

This approach uses a lightweight tool for creating mappings between relational database of WCMS and knowledge base. It also creates a set of abstract classes as a contract for developers who want to implement a communication layer between WCMS and various semantic engines.

As a use case, Apache Stanbol semantic back-end will be integrated with Joomla! CMS for enhancing textual content with extracted entities and semantically enhanced searching.

As a result, I propose a generic system overview of a semantically enriched web content management system as an implementation of a reference architecture.

"The Semantic Web is not a separate Web but an extension of the current one, in which information is given well-defined meaning, better enabling computers and people to work in cooperation. " - Tim Berners-Lee

Contents

1. Introduction

1.1. Motivation

World Wide Web can be considered as the biggest success story of computer science. As stated by Tim Berners-Lee in 2001, the vision of semantic web was not attempting to create a new web, but enriching it with new technologies and standards (which is called as " semantic stack ") that make the data in the web more machine readable and interoperable.

Web Content Management Systems are designed to allow users publish their content on the web without an extensive technical knowledge.[1] These systems are usually built on relational databases. Besides the data that databases contain, every piece of content also holds unstructured data that might be useful in various cases. Several different approaches have been suggested for mapping between RDBMS and RDF [HRG11]. Additional to these approaches applying some other semantic features to web content management systems will help bringing Web 2.0 and Web 3.0 closer.

"Semantifying" a content management system does not imply creating a semantic content management system as it has been analyzed in [Bur08]. Unlike [Bur08], this work approaches from another angle and gives semantic features to conventional content management systems without modifying internal structure of them drastically.

[LAD+10] offers a semantic back-end approach which is separated from CMS for this purpose, however the well-defined interaction only covers JCR/CMIS[2] compliant content management systems. JCR (Java Content Repository) is a standart for content management systems that defines the data storing mechanism. It also offers a Java API to provide interoperability between content repositories. CMIS (Content Management Interoperability Services) is a similar standart to JCR but it does not enforce the use of any programming language [Gon12]. This compliance provides a common object model that guides developers for applying interaction mechanisms with semantic back-ends. On the other hand as I elaborate in Section 2, major web content management systems have no standard when it

[1] http://www.cmscritic.com/cms-or-wcm-which-is-which/
[2] http://xml.coverpages.org/cmis.html

comes to managing content or data structure. Therefore, defining an interaction interface for these web content management systems still remains as a challenge.

1.2. Suggested Approach

This approach creates an implementation of Interactive Knowledge Stack(IKS) reference architecture [CN11]. Detailed explanation of the architecture can be found in Section 3.1. This architecture consists of four main layers: Presentation and Interaction, Semantic Lifting, Knowledge Representation, Reasoning and Persistence. This work focuses on the bottom three layers. An NLP powered content enhancing engine is used for semantic lifting. For the sake of simplicity and portability, knowledge representation and persistence is left to an external triple store. This allows users to separate their knowledge base from the other elements of the architecture more clearly.

In order to implement the connection between content and knowledge repositories, this work uses an SQL based mapping technique and utilizes the RESTful services of semantic engines. It offers a set of abstract classes to provide a contract for interacting various parts of these layers.

Further details about the approach can be found in Chapter 3 and Chapter 4.

1.3. Contribution

The main contributions of this work that I predict can be summarized in following points:

Defining a common contract between web content management systems and RESTful semantic backends: This enables developers to create more portable extensions for their content management systems. By packing these abstract classes as a library, developers can install them on different web content management systems and create concrete extensions.

Creating a generic SQL based mapping to transfer the data to semantic back-end: This allows CMS administrators to transform their data to RDF and store it in a triple store without using a new custom mapping language.

Creating a use case for IKS project[3] on popular Joomla! CMS: IKS is a quite promising project for bringing conventional content management systems together with semantic web.

[3] http://www.iks-project.eu

However, its applications mostly focused on enterprise content management systems with couple exceptions like Drupal. Creating a use case for another popular generic web content management system would also contribute both semantic web and content management system community.

1.4. Structure of the Thesis

The rest of the thesis is structured as follows; Chapter 2 reviews the related concepts and previous work. While Chapter 3 explains the theory in the background, Chapter 4 gives details about implementation. At last Chapter 5 summarizes the work and discusses the advantages and drawbacks of the approach.

2. Background and Related Work

This chapter explains the background of the work and examines the previous effort that has been put on this topic.

2.1. Content Management System

Content Management System (CMS) is a system that allows users to organize their content in a simple way. The word "content" may refer to wide range of concepts, from simple text articles to videos. There are several different types of CMSs according to their purposes. This work only covers web content management systems and review the three most popular ones (Table 2.1).

CMS	**Market Share (%)**	**Absolute Usage (%)**
Wordpress	58.7	24.2
Joomla	6.8	2.8
Drupal	5.0	2.1

Table 2.1.: Usage statistics of popular web content management systems[usa]

All of the three web content management systems in Table 2.1 are written in PHP and distributed under open licences.

2.1.1. Wordpress

Wordpress[wor] is a blogging platform that provides a simple way for users to publish their content. Although it is categorized as a blogging system, this open source platform is also highly extensible towards a CMS by various plug-ins. Core data structure of Wordpress is straightforward (Figure 2.1). All content elements are stored in *wp_posts* table. Contents can also have comments. Tagging and categorizing are also supported, however it is not

natively possible to define content types with custom metadata.

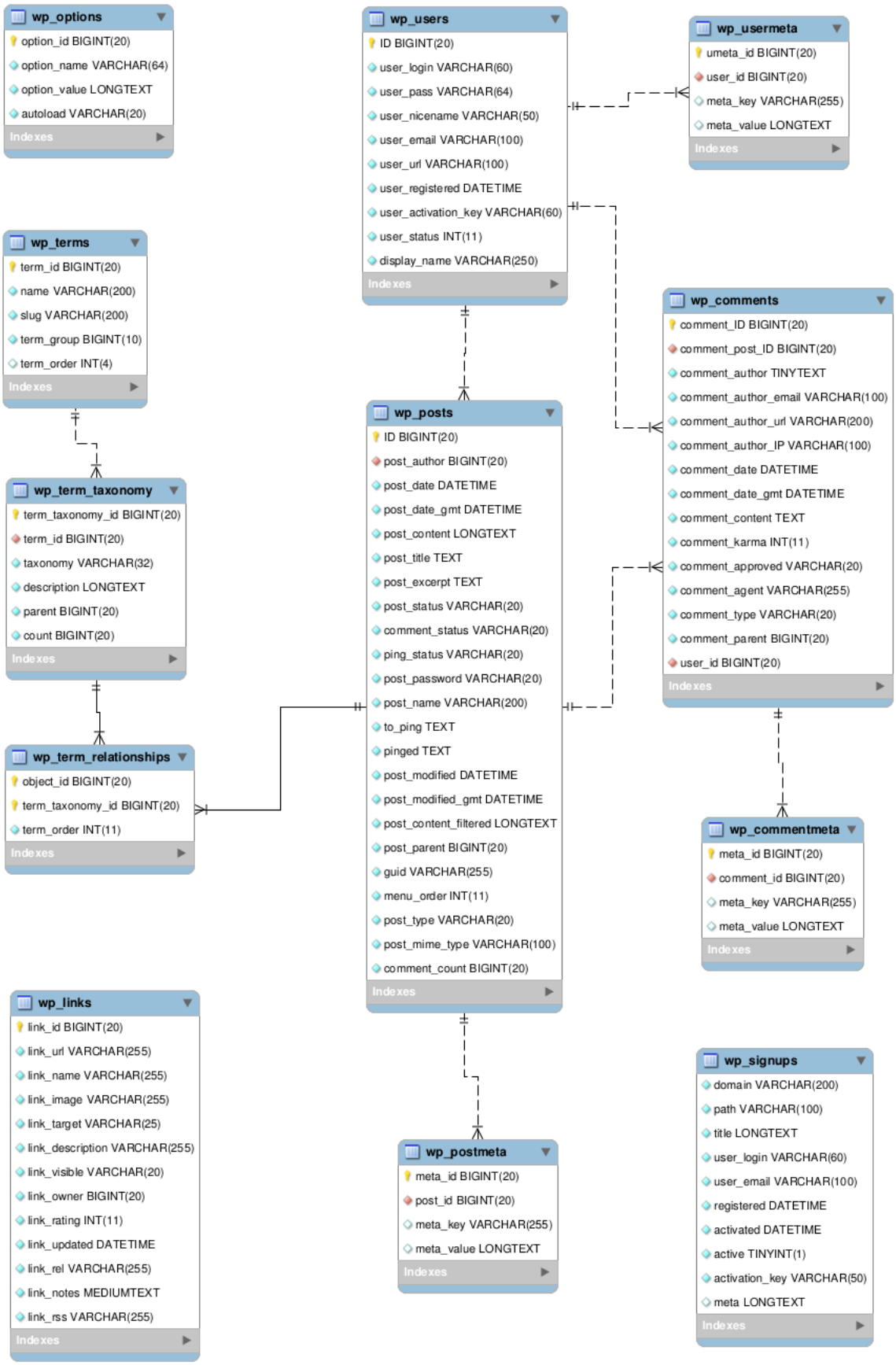

Figure 2.1.: Entity-Relationship Diagram of Wordpress 3.9.4[dat]

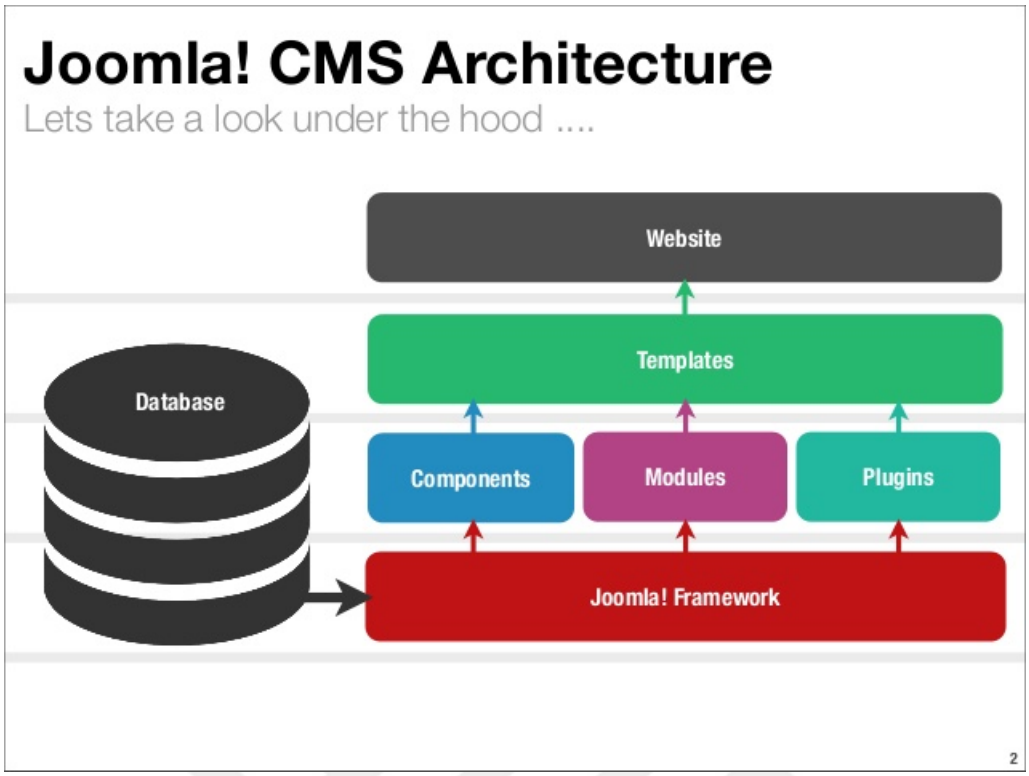

Figure 2.2.: Joomla! CMS Architecture[jooa]

2.1.2. Joomla!

Joomla! [joob] is a highly flexible content management system that allows users to create wide range of web applications from simple web pages to complex e-commerce systems. Its extension based architecture makes Joomla! highly customizable.

As depicted in Figure 2.2, Joomla! divides conventional data-logic-presentation architecture in several elements. Joomla! processes data via different kind of extensions and templates present processed data to user.

In the Joomla! context, extensions are software packages that extend the CMS in various ways. There are seven types of extensions: component, language, library, module, package, plugin and template. [1]

Component

Components are the major extensions of Joomla!. They can be considered as sub-applications of the CMS that provide a kind of custom functionality. A component can have a backend and frontend part.

[1] https://docs.joomla.org/Category:Extension_development

Module

Modules are small extensions that usually manipulates data created by components from different perspectives. Depending on the template, modules can be located different positions on the page. Modules are usually small boxes that show content or interact with user via HTML forms.

Plugin

Plugins have a similar functionality to "event handlers". Besides the built-in plugins, users can also define custom plugins to be triggered by certain events. For example, one can fire a custom functionality whenever a content is saved to manipulate data before it is stored into database.

Library

Libraries are typically sets of third-party code that provide relevant functionality for other extensions. A third-party code can be included to Joomla! as a library and can be later called from other extensions.

Content structure of Joomla! is similar to Wordpress. Every content element must be stored in *jos_content* table[2]. This forces user to create their content with single type of metadata. Contents also can be classified via the category structure of Joomla!. Natively, Joomla! manages content in com_content component. That means, by default, it is not possible to define entities that require different metadata according to user's domain. However, different types of content can be managed via custom components.

2.1.3. Drupal

Drupal[dru] is an open source content management system with an advanced content structure. Drupal treats every content element as a *node*. Drupal supports various content types and also allows users to create their own types and fields via Content Construction Kit module which is included in Drupal's core starting from version 7. It is also possible to define some constraints on the fields. Contents can be categorized by taxonomies. Among the top three CMSs, Drupal is the one that has the most semantic features. Additionally, users can extend their Drupal installations through modules, as well as they do with

[2]The prefix "jos" is subject to change

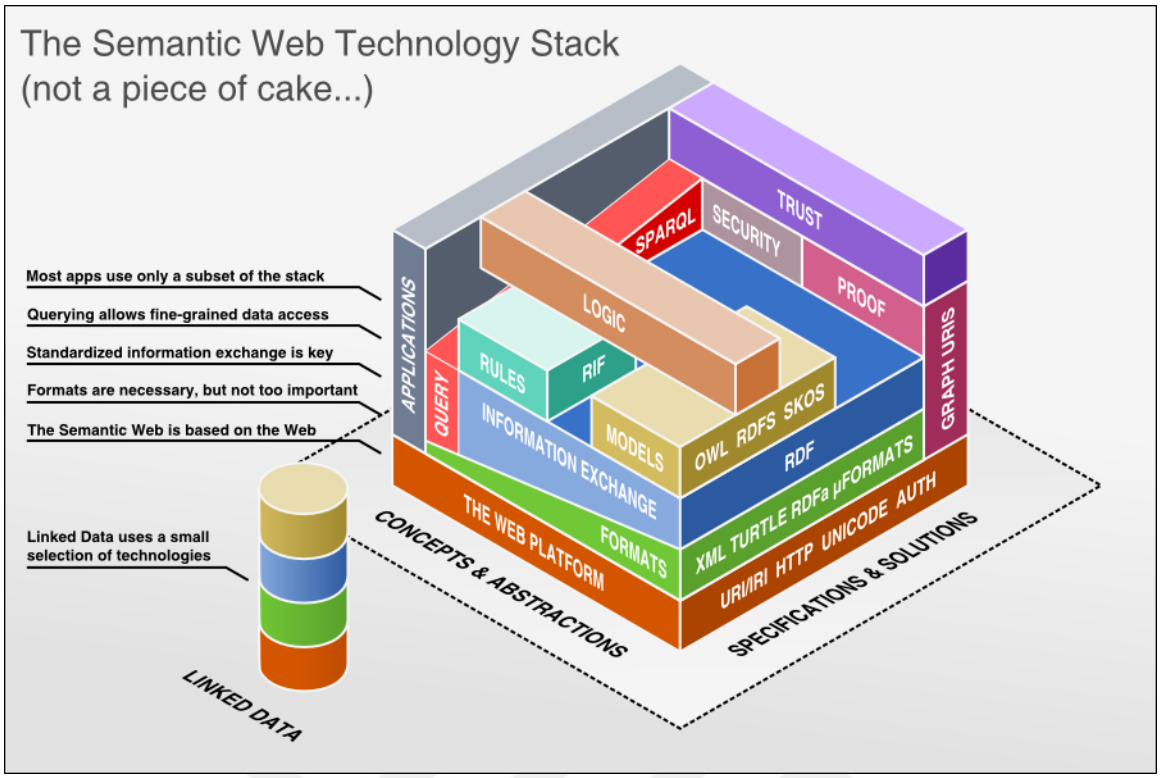

Figure 2.3.: Semantic web technology stack

the other two. Some developments of Drupal regarding to semantic web is mentioned in Chapter 2.3.

2.2. Semantic Web

The semantic web is an extension to the conventional web, that makes it more machine understandable[BLHL01]. In the conventional web, presentation and data live together in HTML documents and they are just some strings between HTML tags and the links between HTML documents are not typed for computers. Semantic Web vision suggests a way to tell the meaning of the data embedded in HTML documents.

As a natural consequence of this vision, it also opens the way towards interoperability of heterogeneous data by using universally unique identifiers for every resource. This allows us to integrate data relatively pain-free, regardless of what the internal data structure of the integrated systems are.

To achive all these goals, semantic web uses a stack of technologies (Figure 2.3[3]),some of which are explained in further sections.

[3]http://bnode.org/media/2009/07/08/semantic_web_technology_stack.png

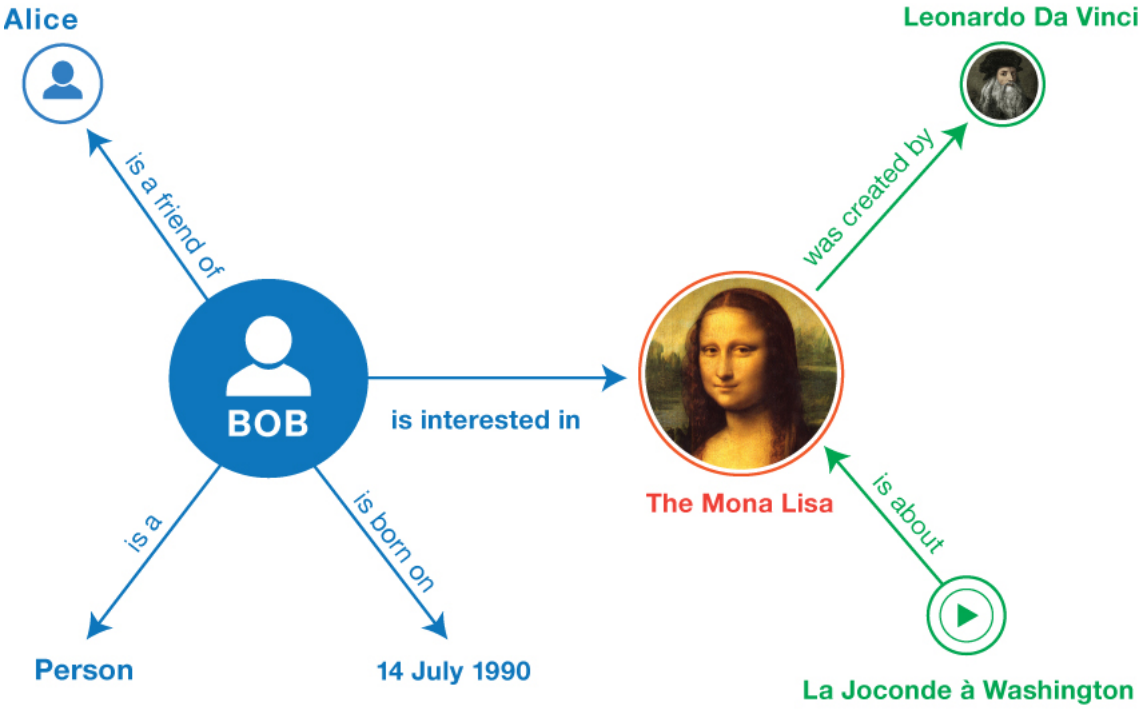

Figure 2.4.: An example RDF graph. [rdf]

2.2.1. Resource Description Framework

Resource Description Framework (RDF) [rdf] provides an elegant way to represent data for semantic web. It is a triple based model that is easy to understand for both humans and machines. Starting from the latest RDF Primer ([rdf]), RDF uses "International Resource Identifier (IRI)" for identifying resources. A resource is typically anything that is identified in an RDF model. Its way to identify resources also allows us to integrate different RDF models without collision.

An RDF model can contain both data and metadata, in fact, the schema language of RDF, RDF Schema is just a extended version of RDF with new elements that can be used to give information about an RDF model [AH08].

RDF is not a serialization method itself but it is an abstract syntax. RDF Primer 1.1 offers various serialization formats as a standart. [4]

An RDF model can be depicted as a graph structure (Figure 2.4).

[4]http://www.w3.org/TR/rdf11-new/#section-serializations

2.2.2. Ontology

As a starting point to describe the term *ontology*, we can use the famous short definition from Tom Gruber: "An ontology is a specification of a conceptualization." [Gru95]

In order to elaborate this short answer, we can think of a real life example. Assume that you are having a conversation about universities. Your brain seamlessly creates a network of concepts. You are prepared to use words like *university, faculty, course, department, project*, on the other hand it is unlikely for you to use the word *spoon*, for example. This is basically creating an ontology in your brain, that has several concepts and relationships of these concepts about university.

In semantic web, ontologies and vocabularies are an abstract modeling of a real world knowledge. The level of details in this modeling, naturally depends on different factors. For instance, an ontology designed for Italian cusine, might only have a concept called *beer glass*, however, a similar one designed for German cusine probably will have more than one type of concept related to beer glass.

Also similar to how human thinking works, ontologies allow us to define rules on concepts. For example when someone gives us a list of countries and the information whether those countries have a coastal border and also names some of those countries as landlocked countries. Afterwards, he gives us the definition of landlocked country as "the countries that have no coastal border". What we naturally do would be going through the list and marking all countries that have no coastal border as landlocked countries. By defining such rules in ontologies, we can extend our knowledge with inference of implicit knowledge.

Although logical expressiveness of knowledge representations vary (Figure 2.5), ontologies and vocabularies help us to work with heterogeneous data from different sources in different levels. For example, being able to define the fact that the concepts *author* and *creator* are the same in some context is a quite valuable feature.

2.2.3. Linked Data

Linked Data is a set of principles for publishing data in a structured and connected manner on the Web [BHBL09].

To reveal the true power of semantic web, putting the data on the Web is not enough. The data needs to be connected in some manner. With the fast growth of the semantic web, there has been lots of open data publishing on the web, but a significant amount of the data could not benefit from the advantages of semantic web because of the poor

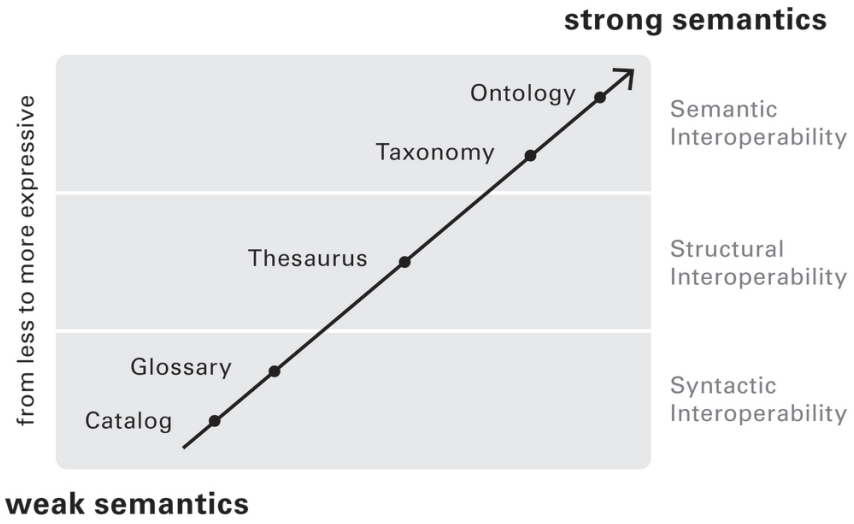

Figure 2.5.: Semantic spectrum

linking. In order to challenge this issue, Tim Berners Lee offered four ground rules for publishing linked data [BL09]:

1. Use URIs as names for things
2. Use HTTP URIs so that people can look up those names.
3. When someone looks up a URI, provide useful information, using the standards
4. Include links to other URIs. so that they can discover more things.

These four rules provide a simple and effective framework for linked data publishers.

The biggest leap that linked data had was the Linked Open Data (LOD) project [lin]. The project started in 2007 and since then, with DBPedia[5] in its heart, it has grown rapidly (Figure 2.6).

After the high adoption of linked data principles and success of LOD project, Tim Berners Lee proposed a new 5-star scale for open data (Table 2.2) [BL09].

2.2.4. SPARQL

SPARQL is a query language for querying and manipulating RDF models [spa]. SPARQL engines match triple patterns with triples in RDF models and returns the matching variables for the SELECT queries. Users can also apply several filters to results. With INSERT and DELETE queries, users can manipulate the graphs by adding and removing

[5]http://dbpedia.org

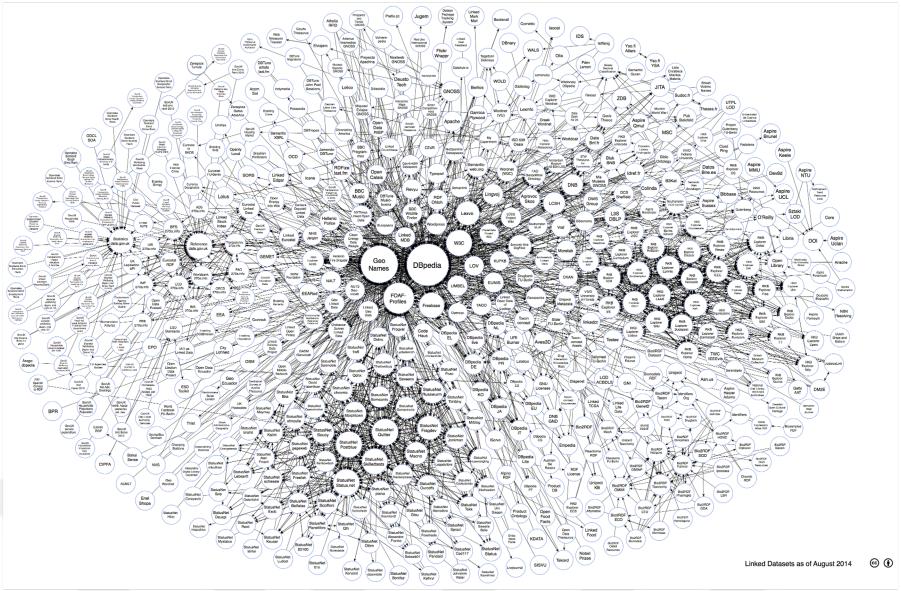

Figure 2.6.: Linked Open Data cloud in 2014[SBJC14]

*	Available on the web (whatever format) but with an open licence, to be Open Data
**	Available as machine-readable structured data (e.g. excel instead of image scan of a table)
***	as (2) plus non-proprietary format (e.g. CSV instead of excel)
****	All the above plus, Use open standards from W3C (RDF and SPARQL) to identify things, so that people can point at your stuff
*****	All the above, plus: Link your data to other people's data to provide context

Table 2.2.: 5-star scale from open data[BHBL09]

new triples. An example SPARQL SELECT query (Listing 2.2) runs on an example RDF model (Listing 2.1) and its result are shown in Table 2.3 [spa] .

Listing 2.1: An example RDF model

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .

<http://example.org/alice#me> a foaf:Person .
<http://example.org/alice#me> foaf:name "Alice" .
<http://example.org/alice#me> foaf:mbox <mailto:alice@example.org> .
<http://example.org/alice#me> foaf:knows <http://example.org/bob#me> .
<http://example.org/bob#me> foaf:knows <http://example.org/alice#me> .
<http://example.org/bob#me> foaf:name "Bob" .
<http://example.org/alice#me> foaf:knows <http://example.org/charlie#me> .
<http://example.org/charlie#me> foaf:knows <http://example.org/alice#me> .
<http://example.org/charlie#me> foaf:name "Charlie" .
<http://example.org/alice#me> foaf:knows <http://example.org/snoopy> .
<http://example.org/snoopy> foaf:name "Snoopy"@en .
```

Listing 2.2: An example SPARQL SELECT query

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name (COUNT(?friend) AS ?count)
WHERE {
    ?person foaf:name ?name .
    ?person foaf:knows ?friend .
} GROUP BY ?person ?name
```

?name	?count
"Alice"	3
"Bob"	1
"Charlie"	1

Table 2.3.: Results of the SPARQL query in Listing 2.2

2.3. Related Work

There are several works related to this work in different levels. The main difference of this work is that being a complete framework that contains enhancement, search and knowledge extraction feature. Also it suggests a common contract for interacting with semantic engines to apply these features. Most of the tools are coupled with specific systems, on the other hand, I aim to implement IKS reference architecture [CN11] by implementing custom CMS semantics to interact with the semantic back-end structures. This SQL based approach adapted from Triplify gives a certain level of generality to knowledge extracting from JCR/CMIS non-compliant web content management systems.

Drupal IKS Enhancement Module is a Drupal extension that enhances and annotates content text with entities indexed into Apache Stanbol[SG12] semantic engine. This module indexes person, organization and place type entities defined in a Drupal site to Apache Stanbol and enhances content when there is an occurrence of these entities [Zie12].

Drupal RDF CCK is a Drupal module for creating and consuming linked data in Drupal. This module aims to expose the data model and the data of a Drupal web site as linked data, along with two other modules, that can create a SPARQL endpoint and import external vocabularies to Drupal. [CDC+09].

FLERSA Tool is a custom semantic annotation tool developed for Joomla! CMS. [NGS10]

Jowlproject is also a work that aims semantifying Joomla! CMS [jow]. A proper documentation or publication for this work could not be found. Therefore any further details cannot be given.

Semantic Media Wiki is an extension to the wiki software Media Wiki. It allows wiki users to annotate their wiki content manually and create linked data from the content [KVV06].

SCMS is a flexible framework for content management systems that allows user to extract structured knowledge, including relationships from unstructured data. It connects to CMS with a "wrapper" component to communicate with the enhancement engine FOX (Federated Knowledge Extraction Framework) [NHL+11].

There are several more frameworks that have been reviewed in [Gon12] including the one this work uses; Apache Stanbol. The framework is explained further in next chapters.

3. Theory and Architecture

This chapter explains the theory behind this work and describes the architectural decisions.

3.1. Interactive Knowledge Stack

Interactive Knowledge Stack (IKS) is a community who develops projects to enhance content management systems semantically. [abo]

[LAD+10] offers an approach for a semantic backend for content management systems as a part of the work of the IKS community. This work is also shaped around that approach. Although [LAD+10] focuses on JCR/CMIS compliant CMSs, they also suggest a RESTful interface as a communication mechanism between JCR/CMIS non-compliant CMSs and the semantic back-end. The back-end uses JCR/CMIS object models as a generic reference for creating knowledge models from relational data structure. So called "semantic bridges" described in [LAD+10] interact with JCR/CMIS content repositories and reflect the changes to the semantic back-end. For non-JCR/CMIS content management systems, [LAD+10] defines RESTful interfaces that interact with semantic back-end, whenever a change occurs on CMS.

A more abstract representation of their architecture can be shown as in Figure 3.1 [1]

This work will try to create a standardized way to realize "Custom CMS Semantics" interaction, by suggesting a interaction method based on common standards. In CMS world, every vendor has different data structure decisions, however there is a significant amount of products that support SQL based database management systems. Using this feature and SPARQL over HTTP protocol will provide us an adequate generality to implement such an interaction layer.

Despite these similarities in persistence technologies, the way different content management systems approach to *content* varies. While Drupal allows users to create different types of contents in terms of metadata[2] (properties related to content types), Joomla!

[1] Re-drawn from Fig.7 in referenced paper
[2] https://www.drupal.org/documentation/structure

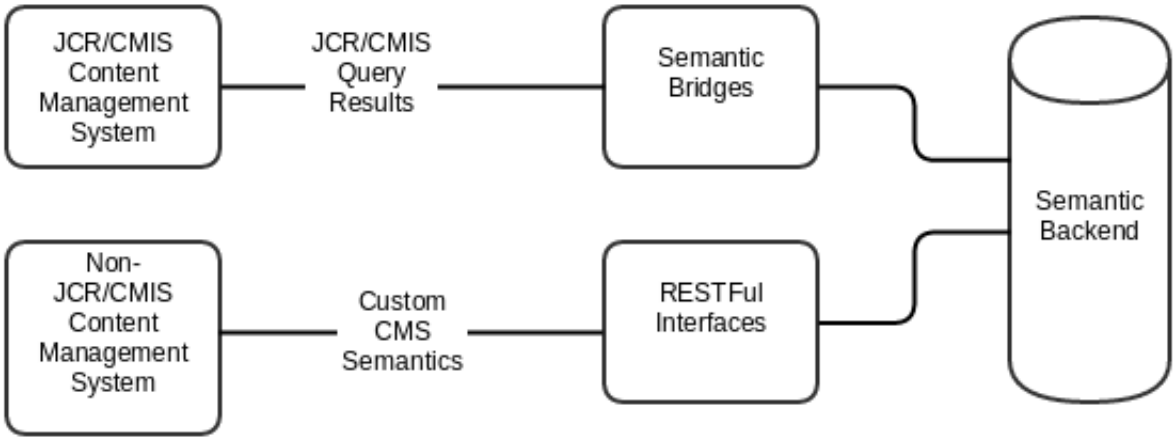

Figure 3.1.: Abstract representation of the backend architecture [LAD+10]

and Wordpress adopt a more rigid approach that forces users to fit their content to one schema. Although this restrictions can be eliminated by developing custom extensions, it is an obstacle on the way towards providing a generic way to communicate with an external structure, in this case, a semantic engine.

A reference architecture for such a backend has been proposed in the scope of IKS as well. This architecture (Figure 3.3) defines a correspondence between conventional CMS architectures and a knowledge management system. In Figure 3.3, the communication between *content* and *knowledge* columns is provided by a gate. This gate implemented as a "CMS Adapter" by [Gon12] for JCR/CMIS content management systems [CN11]. A concrete implementation of this architecture called Apache Stanbol [SG12] uses different modules for different elements of the *knowledge* column in the referenced architecture proposed by [CN11]. Apache Stanbol also provides RESTful services.

A partial overview of the interface this work proposes is shown in Figure 3.2. The connection to persistence component is provided by RESTful services.

3.2. Creating Knowledge Models

This section describes the theory behind the knowledge extraction from relational database.

3.2.1. Extracting Knowledge from Relational Database

In order to enable semantic features on a conventional CMS, the first thing we need to do is mapping the relational data structure to RDF. There are various tools that use different techniques for creating RDF data from relational database. Since one of the goals is intervening to the conventional CMS structure as little as possible, In this work a lightweight

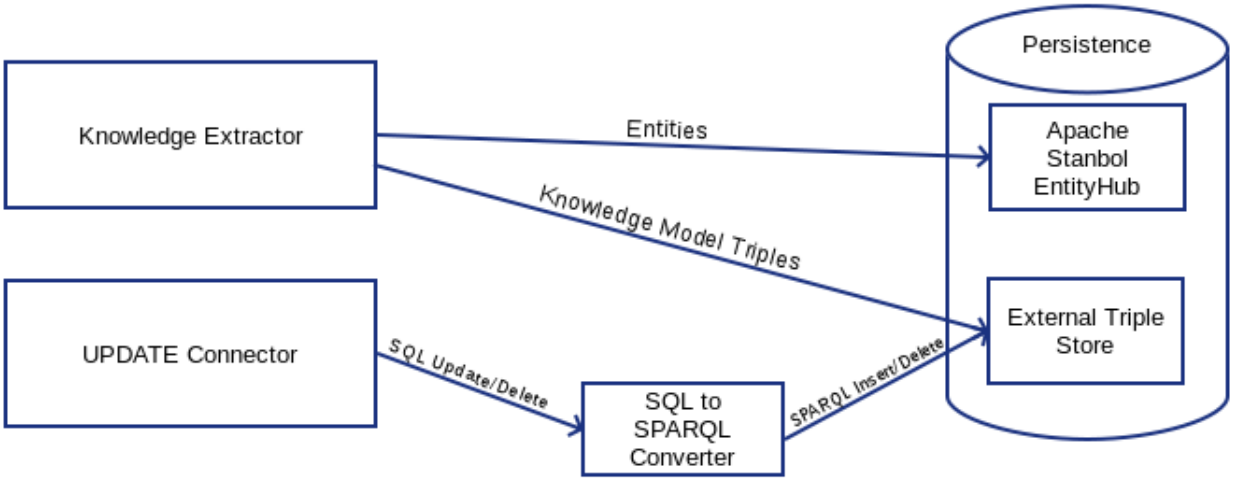

Figure 3.2.: Partial Interface Structure for Knowledge Transfer

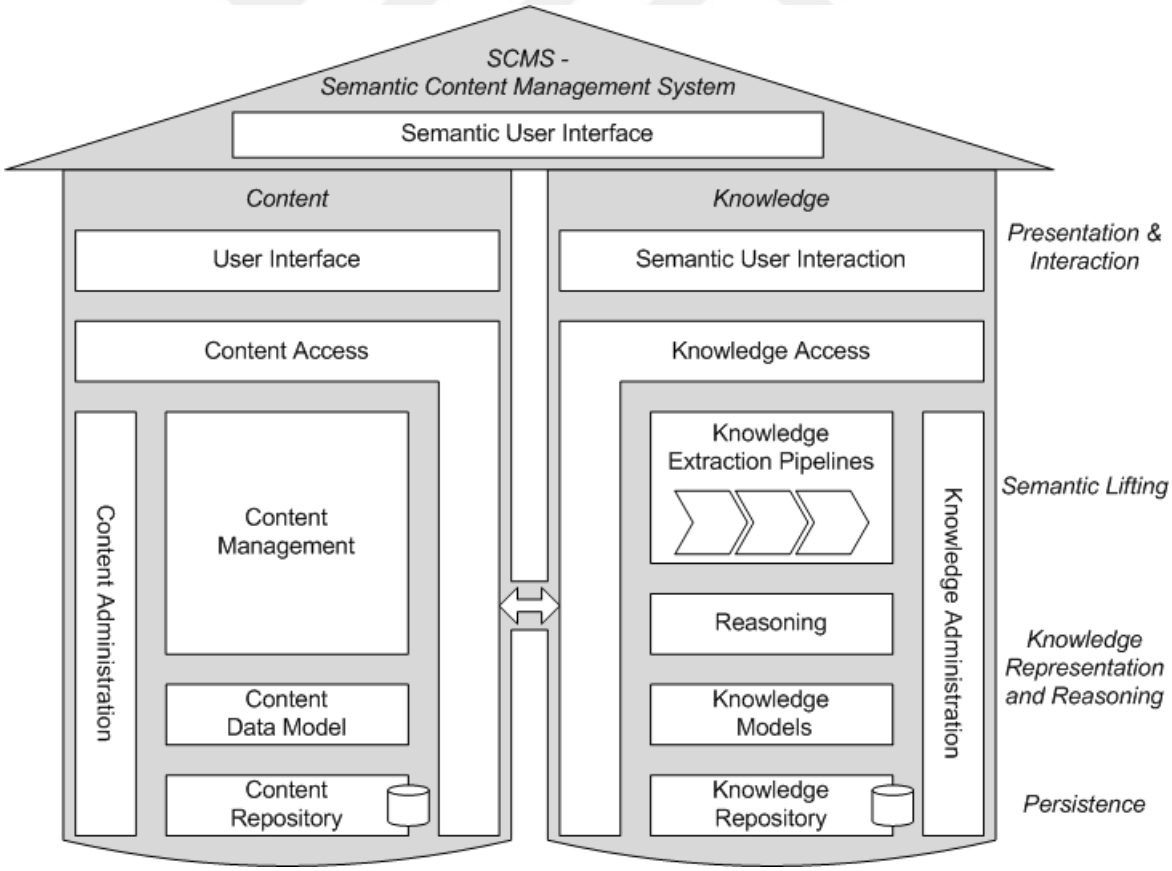

Figure 3.3.: IKS Reference Architecture - Reprinted from: [CN11]

and easy to configure method to convert relational data to RDF have been chosen. Among the tools that have been tested in [HRG11], this work adopts Triplify [ADL+09]. Triplify offers a lightweight tool which uses SQL as mapping language, therefore it allows us to use almost every aspect of relational databases. It does not require any type of server installation which supports its simplicity. Its domain driven nature makes it suitable for custom mappings other than table to class and column to predicate mappings. [ADL+09] This approach is used for both entity extraction and creating a knowledge model of CMS data structure. Figure 3.2 shows the extraction process. SQL to SPARQL Converter module parses the SQL query and loads them into software objects. Afterwards, this software objects are used to generate SPARQL queries.

3.2.2. Loading Knowledge to Semantic Engine

To be able to operate on the RDF triples we have extracted we need to load the triples to our persistent storage. This work uses two different persistence structures. The first one is Apache Stanbol Entityhub, where the entities relevant to content enhancements are stored[apa]. The second one is a standalone triple store that stores all the extracted knowledge from CMS to enable features like semantic search and contributing to linked open data cloud. Entityhub provides a RESTful API for access [BG13]. Extracted entities can be loaded to a ManagedSite[3] and managed locally. This Restful API also allows users to query entities with keywords and filter the results with LDPath programs. This querying facility is useful when Entity Linking process cannot link a recognized entity with vocabulary. Additional to ManagedSite, it also provides an interface called ReferencedSite[4]. This interface allows user to benefit external resources during the enhancements process that is explained in Section 3.3.

3.3. Content Enhancement

Content enhancement in semantic web context means that, enriching any kind of unstructured, semi structured or structured content with extracted (i.e through NLP or image processing methods) or discovered (i.e ontology exploration) semantic metadata. In this work, for the sake of simplicity, the word content refers only to textual content. Textual content is processed by the NLP enhancement engines of Apache Stanbol Enhancer component.

[3] https://stanbol.apache.org/docs/trunk/components/entityhub/managedsite.html
[4] How to configure: https://stanbol.apache.org/docs/trunk/customvocabulary.html

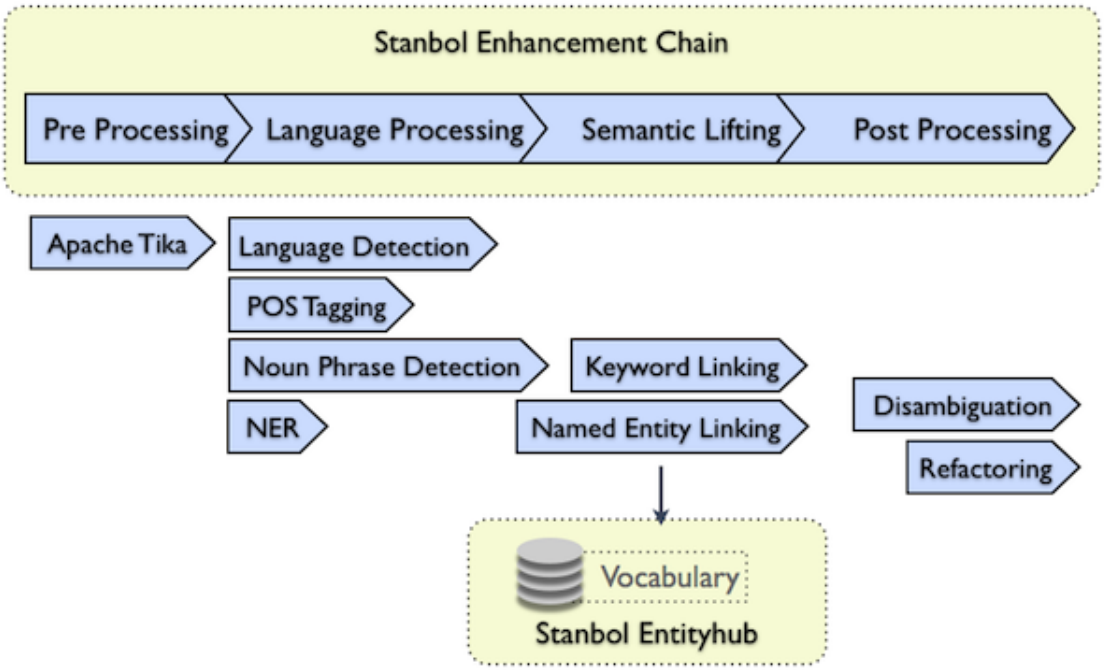

Figure 3.4.: Stanbol Enhancement Workflow

The RESTful nature of Apache Stanbol allows us to communicate with the enhancer over HTTP. The component works with so called "enhancement chains" and every chain consists of several NLP engines.[5] A typical NLP based enhancement process of Stanbol works as shown in the Figure 3.4. [6]

The main reason that Apache Stanbol is chosen as the enhancement engine is its ability to work with local custom vocabularies. It is shown in a benchmark study that Stanbol performs better than its competitors when it comes to recognizing and linking entities to a custom vocabulary [HHHR14].

In the enhancement process, two key engines are used: Named entity tagging and entity linking. Named entity tagging engine applies a NER (Named Entity Recognition) operation and finds the occurrences of common entities that have pre-defined classification such as person, place and organization. Entity linking [RMD13], links the recognized entity to the loaded vocabulary. A TextAnnotation is created for the occurrence and an EntityAnnotation is created for entity linking. (Figure 3.5) [7]

After the enhancements are obtained, this approach applies a disambiguation process. Since the vocabulary already exists, when multiple linking options for an entity are sug-

[5] List of available engines: https://stanbol.apache.org/docs/trunk/components/enhancer/engines/list.html
[6] Figure reprinted from: https://stanbol.apache.org/docs/trunk/customvocabulary.html
[7] The detailed information about the enhancement structure can be found at https://stanbol.apache.org/docs/trunk/components/enhancer/enhancementstructure.html

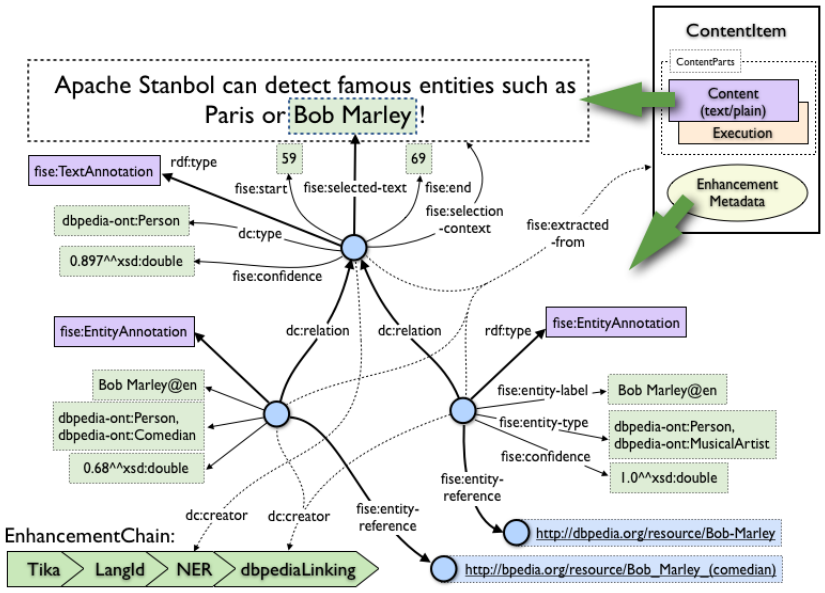

Figure 3.5.: Enhancement structure of Apache Stanbol

gested, my approach considers the entity with the other enhancements. This means set of queries are used to find out whether there is a semantic relationship between the entities. It adjusts the confidence values according to the results of these queries and suggests the one that has higher relationship ranking above the others.

3.4. Semantic Search

In the context of this work, semantic search is used for improving keyword search in content management systems with the aid of recognized entities and extracted knowledge from relational database. Semantic search can be defined as retrieving information by using the semantics of the concepts that are searched for. It helps predicting the context and the intention of the user's question [wha]. This retrieval is usually done by running queries on ontologies. As it is pointed out in [TCRS07] users usually use keywords instead of full natural language queries. The semantic search process is conducted by matching concepts with keywords similar to the approach described in [TCRS07]. Since we have entities related to domain stored in EntityHub, this matching process is even simpler. Once the concepts are found, SPARQL queries run on the knowledge base and retrieve contents related to entities, as well as their class types. This SPARQL queries can be also extended by their translations to different languages with the aid of BabelNet [bab]. Process is summarized in Figure 3.7. First the keywords are parsed and a keyword set is obtained. For every member of this set, *find* function of Apache Stanbol Entityhub

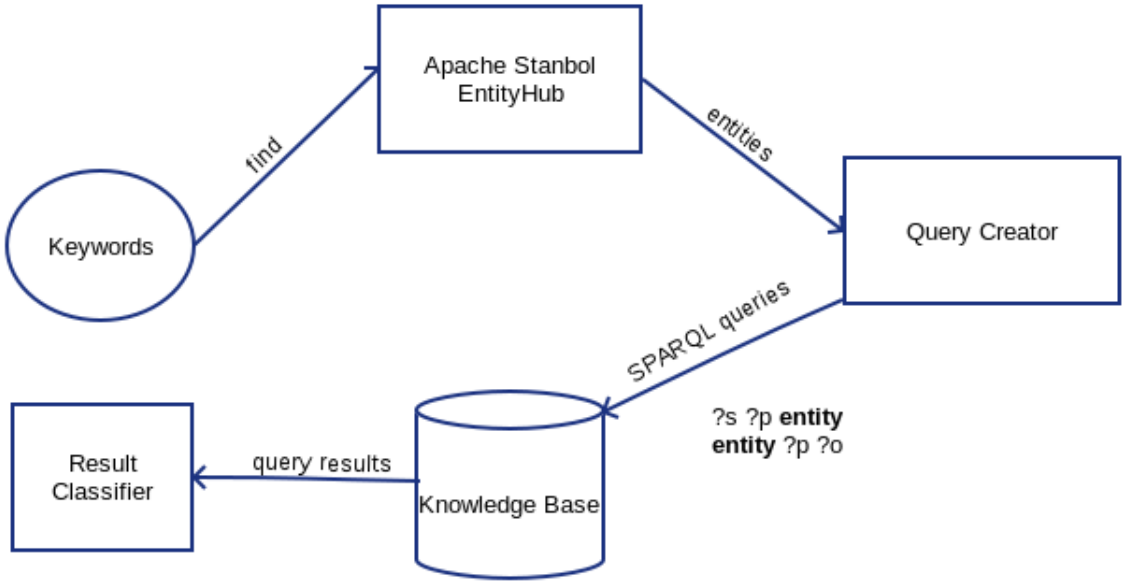

Figure 3.7.: Semantic search process

is called. In case of another semantic engine is used, EntityHub's purpose can be easily replaced with a store that has a text indexing functionality. Each function call returns a set of results. The final result set that is sent to Query Creator module consists of union of these sets. (Figure 3.6)

$$R_f = R_1 \cup R_2 \cup ..R_n | R_i : i^{th} resultset, n : keywordcount, n \epsilon N^+$$

Figure 3.6.: Formal definition of final entity set that is sent to Query Creator

Query Creator, creates two types queries for each entity in the set. That are, one query where the entity is the object of the triple pattern and the other where it is the subject of the triple pattern. By creating proper links on results of this query, it also allows users to explore in the knowledge base. Afterwards, the query results are sent to the Result Classifier, where the results are classified by their type. A result can be directly an entity in the domain or a content element such as an article. This classification allows us to provide different presentations according to type.

4. Implementation

This chapter gives details about the implementation of the work described in previous chapters.

4.1. Semantic Interface for Web Content Management Systems (SIWCMS)

As it is addressed in Chapter 2, most of the web content management systems have neither any standart way to manage content nor a common object model to use as an interface like JCR/CMIS. In order to challenge this issue, this work proposes a semantic interface which is a set of abstract classes that helps system developers to create semantic extensions for web content management systems. An example SIWCMS implementation is done in PHP. Therefore it benefits some features of PHP such as weak typing, especially some functions can return mixed datatypes. Naturally, porting this interface to other web programming languages with different inheritance and typing strategies may cause slight changes in implementation. Consequently, this PHP implementation utilizes several third-party libraries written with PHP as well. It uses EasyRDF [eas] for RDF manipulations, Httpful[htt] library for RESTful operations and php-sql-parser[php] for parsing SQL queries.

At the highest level, SIWCMS consists of few abstract classes and interfaces shown in (Figure 4.1).

SemanticEngine is the class that represents the semantic back-end that content management system uses. A content management system may use one or more semantic back-ends for different purposes. For each of them, one concretion of this abstract class should be made. Each engine can contain multiple amounts of modules.

SemanticEngineOptions is an abstract class with which set of options for the semantic backend can be concretized. Developers can manipulate these options later in the methods of SemanticEngine class.

SemanticEngineModule represents a single module of semantic backend. (i.e enhancement, search, store). A module has an access URL for RESTful operations. It can also contain multiple operations.

Operation is the abstract class, concretions of which represents a basic operation in a semantic back-end module. This class uses the template method design pattern, by which developers can define custom operations before data is sent and after results are retrieved. The encapsulated sendRequest method in this class, sends an HTTP request to RESTful services and receives the response. Public run method calls abstract preRun, sendRequest and abstract processResults methods respectively (Listing 4.1). The data to be sent can be manipulated in preRun and HTTP response can be manipulated in processResults.

ICustomOperation is an interface that defines a operation function which allows developers to implement their custom operation and pass it to Operation objects constructor. The run method runs the default sendRequest function unless an instance of a class that implements ICustomOperation interface is passed to operation.

Listing 4.1: Method run in class Operation

```
1     /**Function that runs an operation of a semantic engine module after
          preRun. processResults must be implemented
2      *
3      * @param string $url     semantic engine module url
4      * @param mixed  $data    data to be processed
5      * @param array  $options module options
6      * @param null   $auth    authentication values
7      * @param null   $apiKey  API-Key for module
8      */
9     public final function run($url, $data, array $options=null, $auth=null,
           $apiKey = null)
10    {
11
12        $this->setData($data);
13        $this->preRun();
14        if (!$this->_customOperation)
15            $this->sendRequest($url, $this->getData(), $options, $auth,
                  $apiKey);
16        else
17            $this->_result = $this->_customOperation->operation($url, $this
                  ->getData(), $options, $auth, $apiKey);
```

```
18          return $this->processResult();
19
20      }
```

SIWCMS also offers some supplementary abstract classes for operations of semantic engines that are assumed to be common in this work. For example for content enhancement, SIWCMS offers Enhancement and Entity abstract classes. Enhancement defines common properties and operations that can an enhancement engine have. Entity class, which is a subclass of EasyRDF_Resource defines a super type for any kind of entity that a domain can have. It implements its own type mapping system for matching software classes with RDF types. This allows us to create different type of entity objects dynamically.

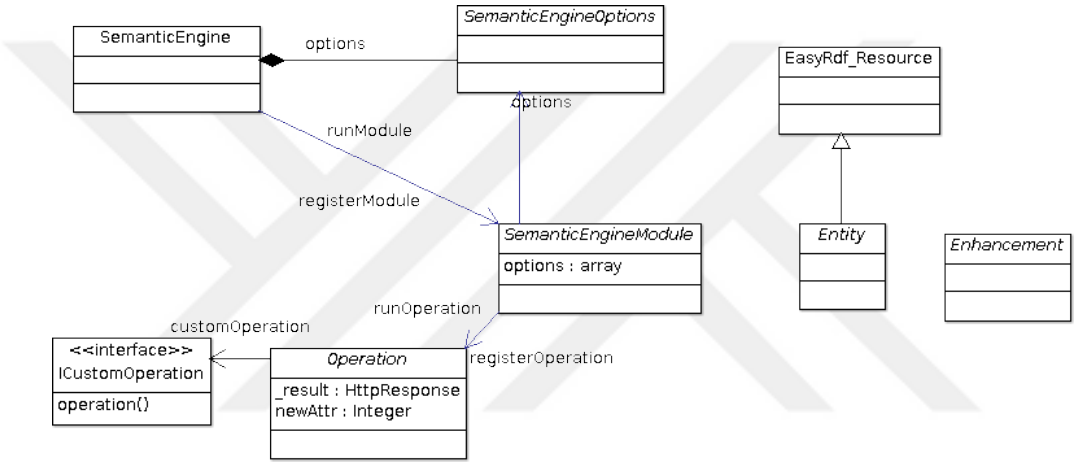

Figure 4.1.: SIWCMS basic class structure

4.2. SJ! Extension

The SJ! Extension is a collection of components, modules and plugins for Joomla! 2.5.

4.2.1. Enhancer Plugin

Editor Plugin

This plugin is an editor-xtd button, that provides an entry point for content enhancement (Figure 4.2).

Figure 4.2.: Semantify editor-xtd button

The plugin button calls some JavaScript functions to retrieve HTML text from the editor area. Then it sends this content to back-end via an AJAX call. (Listing 4.2) This back-end file calls a a SemanticEngine implementation with suitable operation. SemanticEngine is simply an implementation of the SIWCMS classes described in Section 4.1 (Listing 4.3). The semantic engine that is connected via RESTful operations runs an NLP pipeline on the content end extracts entities. The results obtained after AJAX calls are stored in a session variable, then showed by the view file. (Figure 4.3)

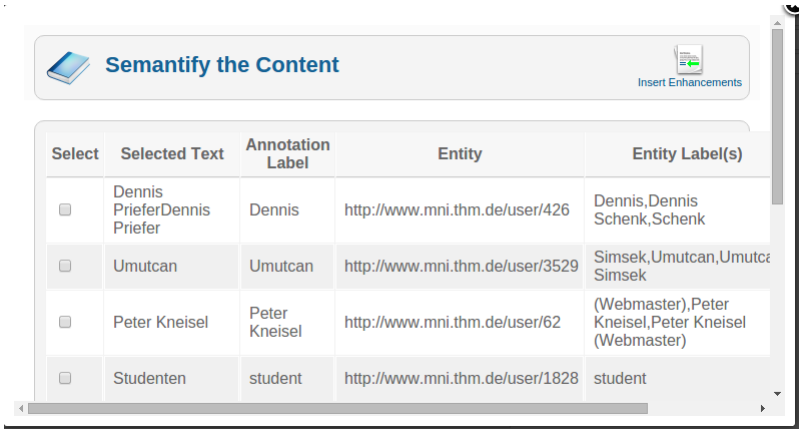

Figure 4.3.: List of enhancements

User selects the entities that have an occurrence in the text. After clicking "Insert Enhancements" button, plugin automatically creates links on entity occurrences to related pages.

Listing 4.2: Editor-xtd plugin button

```
1   public function onDisplay($name)
2   {
3
4       $editor = JFactory::getEditor();
5       $doc = JFactory::getDocument();
6       $getContents = $editor->getContent($name);
7
8       $baseFileURL = JURI::root() . '/plugins/editors-xtd/sj_enhancer/';
9
10      JFactory::getDocument()->addStyleSheet($baseFileURL . 'assets/
            subjects.css');
11
12      // Reference to current object languages
13      $lang = JFactory::getLanguage();
14      $languageTag = $lang->getTag();
15      $lang->load('plg_editors-xtd_sj_enhancer');
16
17      $link = '../plugins/editors-xtd/sj_enhancer/view.php?';
18      $link .= "&e_name=$name&languageTag=$languageTag";
19
20      JHtml::_('behavior.modal');
21
22      $postContent = '
23      function enhance(x){
24      var y = encodeURIComponent(x);
25      var postData="stanbol='.
26          $this->params->get('txtEnhancerSite').'&contentText="+y;
27      $.ajax({
28      data:postData,
29      url:"'.$link.'",
30      type: "POST",
31      success:function(result){
32
33      }
34      });
35
36      return true;}';
37
38      $script = 'function getEditorText() {return '.$getContents.'}'.
            $postContent;
39
40      $doc->addScript('./../libraries/jquery/js/jquery.min.js');
```

```
41          $doc->addScriptDeclaration($script);
42          // Button
43          $button = new JObject;
44          $button->set('modal', true);
45          $button->set('text', JText::_('PLG_SJ_ENHANCER_BUTTON_LABEL'));
46          $button->set('name', 'btnSemantify');
47          $button->set('options', "{handler: 'iframe', size: {x: 770, y:
                400}}");
48          $button->set(
49              'onclick', "enhance(WFEditor.getContent('jform_articletext'));"
50          );
51          $button->set('link', $link);
52
53          return $button;
54      }
```

Listing 4.3: Model file which calls runModule method of semantic engine with suitable module and operation

```
1       /**
2        * Function that enhances the text
3        * @param string $text Text to be enhanced
4        * @return array enhancements array
5        */
6       public function enhance($text)
7       {
8           $enhancement=$this->_semanticEngine->runModule("enhancer", "enhance
                ", $text);
9
10          return $enhancement;
11
12      }
```

Content Plugin

This is a lightweight plugin that stores the annotations created by using the enhancements to triple store. An annotation is basically a set of RDF triples that connects annotated article with the extracted entity (Figure 4.4) from the textual content of that article.

For this purpose, a storing module on an SemanticEngine implementation is defined. In the implementation of preRun method of store operation, the article URL is added to

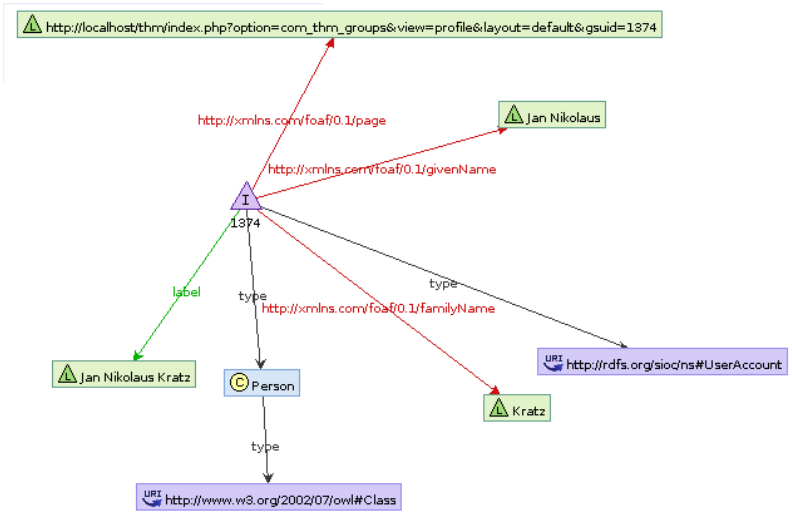

Figure 4.4.: An example entity

entity with about property of SIOC[1] ontology. (Listing 4.4)

Listing 4.4: Example RDF graph in Turtle format with an article and its entities

```
1 @prefix sioc:  <http://rdfs.org/sioc/ns#> .
2
3 <http://localhost/thm/administrator/index.php?com_content&view=article&id
      =4462>
4         sioc:about  <http://www.mni.thm.de/user/604> , <http://www.mni.thm.
              de/user/535> , <http://www.mni.thm.de/user/360> , <http://www.
              mni.thm.de/user/473> .
```

Stored annotations are used in search process later. (Subsection 4.2.3)

Note that, this usage of the annotations is only an example. Naturally, obtained enhancements can also be used for creating RDFa or JSON-LD scripts for better search engine optimization. In the case of Joomla!, this can be used for creating automated JMicrodata[2] instances.

4.2.2. Conceptualization of Knowledge Bridge Component

This component is the core of the semantic enrichment process. It is designed as a Joomla! component, that provides a user interface for defining SQL queries, that transfers the relational data to knowledge persistence modules of semantic back-end. As Figure 3.1 shows, there are two persistence modules, Apache Entityhub; a triple store with text

[1] Semantically-Interlinked Online Communities - http://rdfs.org/sioc/spec/
[2] https://docs.joomla.org/Microdata#JMicrodata_documentation

indexing store and an external triple store. Apache Entityhub naturally can be replaceable with any other similar structure.

My Triplify based approach basically converts relational data to RDF triples. As a use case for prototype, Queries for the user data is created for the THM's website database. This data is converted to a simple FOAF (Friend of a Friend) graph. It is up to developers to how detailed the queries for each persistence modules should be, but the processing of an example query is shown in Listing 4.5.

Listing 4.5: An example triplify mapping query

```
1 SELECT id as id, SUBSTRING_INDEX(name,' ', LENGTH(name)-LENGTH(REPLACE(name
      ,' ',''))) as 'foaf:givenName@de' , SUBSTRING_INDEX(name, ' ', -1) as '
      foaf:familyName@de', 'sioc:UserAccount' as 'rdf:type', name as 'rdfs:
      label@de', CONCAT('".$bsURI."/index.php?option=com_thm_groups&view=
      profile&layout=default&gsuid=',id) AS 'foaf:page'  FROM jos_users",
```

The details of the parsing approach of the SQL queries for mapping can be found in [ADL+09]. One point one should notice is, every query has a unique name defined by the user. This name is used to map RDF types to query results. An example name may be *user* for the query in Listing 4.5. A mapping to foaf:Person type would be suitable. The column with the id alias represents the subject of a triple that is converted from a row of the query result, where the other columns are represented as object and aliases as property of a triple. Aliases also contain the language tags for literals. One issue needs to be taken care of is the distinction between properties that have literal and resource ranges. This distinction by default is done by a seperate mapping for object properties. As a more intuitive approach, ranges of desired properties are used to create correct mapping for a property. An example triple extracted by the query in Listing 4.5 is shown in Listing 4.6.

Listing 4.6: An example extracted set of triples

```
1 <http://www.mni.thm.de/user/360> <http://www.w3.org/1999/02/22-rdf-syntax-
      ns#type> <http://xmlns.com/foaf/0.1/Person> .
2 <http://www.mni.thm.de/user/360> <http://xmlns.com/foaf/0.1/givenName> "
      Dennis"@de .
3 <http://www.mni.thm.de/user/360> <http://xmlns.com/foaf/0.1/familyName> "
      Priefer"@de .
4 <http://www.mni.thm.de/user/360> <http://www.w3.org/1999/02/22-rdf-syntax-
      ns#type> <http://rdfs.org/sioc/ns#UserAccount> .
5 <http://www.mni.thm.de/user/360> <http://www.w3.org/2000/01/rdf-schema#
      label> "Dennis Priefer"@de .
```

```
6 <http://www.mni.thm.de/user/360> <http://xmlns.com/foaf/0.1/page> "http://
      localhost/thm/index.php?option=com_thm_groups&view=profile&layout=
      default&gsuid=360"
```

Another key point of concept of this component is keeping the data transferred to knowledge base synchronized. This remains as an important challenge as different web content management systems have different policies by the means of triggering events after certain operations. One approach can be leaving this operation to user via a specific administration option. This means, in a certain frequency that user decides, he/she triggers the synchronizing event manually. The synchronization mechanism works through converting data extraction queries into SPARQL INSERT/UPDATE queries. Another approach might be suitable for Joomla! for instance, mapping components which are responsible for creating the data to be extracted and defining custom events to be triggered after operations of these components. JEvent[3] class of Joomla! framework can be a legitimate way to apply this approach.

With a similar rationale that the implementation in [CDC+09], an interface for importing external ontologies and vocabularies should be provided to encourage the reuse of existing concepts and properties. To create a more improved experience an auto-completion feature will be added to SQL query writing process with the guidance of vocabularies imported. Additionally, some vocabularies such as Schema.org, FOAF and SIOC can be included in the semantic backend by default.

4.2.3. Semantic Search

This feature enriches the conventional keyword search of Joomla! by bringing more accurate results for entities. Conventional search engine of Joomla! does a simple text-based search. (Figure 4.5) The search feature can be extended to have it work on the data of custom components via search plugins. However, this still doesn't go beyond syntactic search and requires programming effort.

As it is mentioned in Section 3.4, the RDF graph that is extracted from relational database, can be stored in two different persistence modules. One module, Apache Stanbol Entityhub in this use-case, operates as an entity repository with advanced indexing facilities. By the nature of RDF, it eliminates the sharp distinction between metadata and data. This allows us to create better search features than the syntactic one. For instance,

[3] https://docs.joomla.org/API16:JEvent

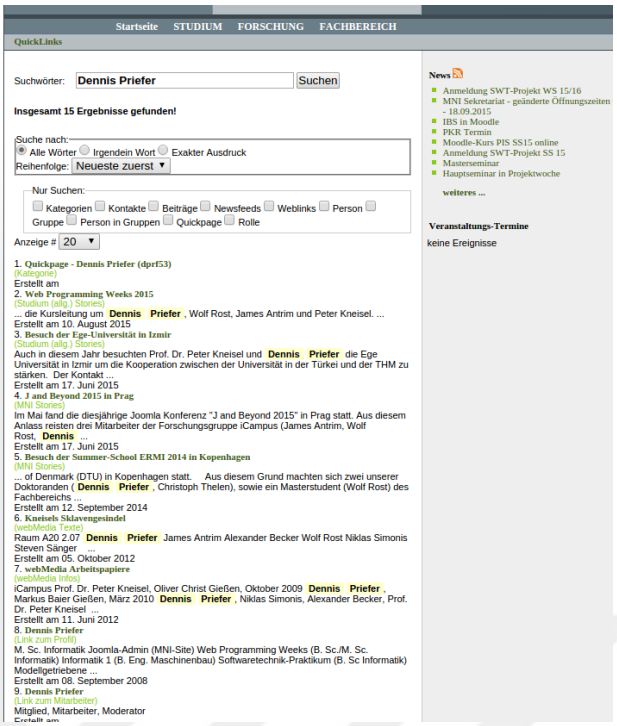

Figure 4.5.: Conventional Joomla! search results

in this prototypical EntityHub graph, based on Technische Hochschule Mittelhessen's data; Dennis Priefer is a Person and an UserAccount. Therefore, when a keyword based search is made, the type of the entity can be recognized and basic structured information from triple store can be retrieved about the entity. Furthermore by using semantic reasoning on triple store. New triples can be discovered and also retrieved during the search operation.

This extension is prototyped as a template add-on and a module. The plugin uses a custom jquery autocomplete widget[4] to send keywords to Apache Stanbol Entityhub's RESTful endpoint.[5] (Listing 4.7) By that, one can obtain results in form of RDF resources with categories. (Figure 4.6) The endpoint gets two parameters name and ldpath. name is simply the parameter that holds the keyword. ldpath holds a LDPath script. LDPath [6] is an XPath-like path based query language. An LDPath script is used to tell Entityhub, which properties of found resources should be retrieved. In this case, it returns the labels and types of the found resource.

Listing 4.7: JQuery script for semantic search

```
2           $.widget( "custom.catcomplete", $.ui.autocomplete, {
3               _create: function() {
```

[4]https://jqueryui.com/autocomplete/
[5]example endpoint: /entityhub/site/thmgroups/find
[6]http://marmotta.apache.org/ldpath/

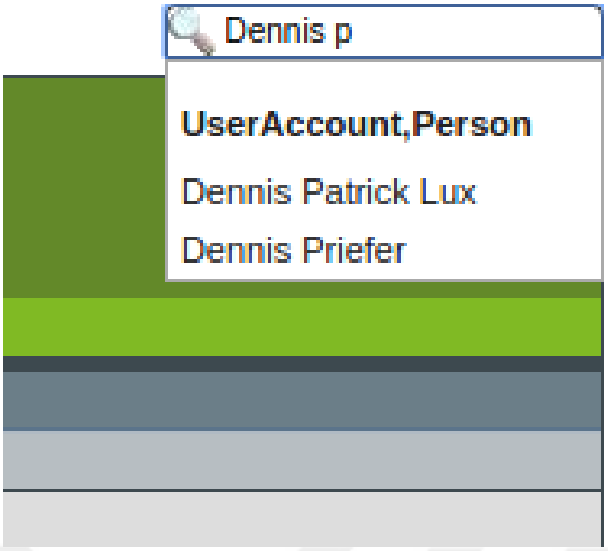

Figure 4.6.: Jquery entity search plugin

```
4                   this._super();
5                   this.widget().menu( "option", "items", "> :not(.ui-
                        autocomplete-category)" );
6               },
7               _renderMenu: function( ul, items ) {
8                   var that = this,
9                       currentCategory = "";
10                  $.each( items, function( index, item ) {
11                      var li;
12                      if ( item.category != currentCategory ) {
13                          ul.append( "<li class='ui-autocomplete-category'>
                                " + item.category + "</li>" );
14                          currentCategory = item.category;
15                      }
16                      li = that._renderItemData( ul, item );
17                      if ( item.category ) {
18                          li.attr( "aria-label", item.category + " : " +
                                item.label );
19                      }
20                  });
21              }
22          });
23          $(function() {
24              function log( message ) {
25                  $( "<div>" ).text( message ).prependTo( "#log" );
```

```
26                  $( "#log" ).scrollTop( 0 );
27              }
28
29              $( "#mod_search_searchword" ).catcomplete({
30                  source: "modules/mod_sj_search/Ajax_findResource.php",
31                  minLength: 3,
32                  select: function( event, ui ) {
33                      $.ajax({
34                          url: "modules/mod_sj_search/Ajax_findResource.php
                                ",
35                          method:"POST",
36                          data: "search=1&resource="+ui.item.id ,
37                          success: function( data ) {
38
39                              $("#searchResults").html(data);
40                          }
41                      });
42
43                  },
44                  open: function() {
45                      $( this ).removeClass( "ui-corner-all" ).addClass( "
                            ui-corner-top" );
46                  },
47                  close: function() {
48                      $( this ).removeClass( "ui-corner-top" ).addClass( "
                            ui-corner-all" );
49                  }
50              });
51          });
```

After the retrieval of the matching entities, user selects an entity and this selection triggers an another AJAX call to finalize search operation. In this prototype, search operation is simply a SPARQL query that returns The backend file exists in the Joomla! module that shows the search results.

In the backend, SIWCMS structure is used. Similar to content enhancement operation, there is a semantic engine that has modules and operations for finding the resource and conducting the search. By default, every class that extends abstract Operation class of SIWCMS uses the predefined RESTful operations. To allow developers to define their custom operations, as I briefly mentioned in Section 4.1, there is an interface. In case of an

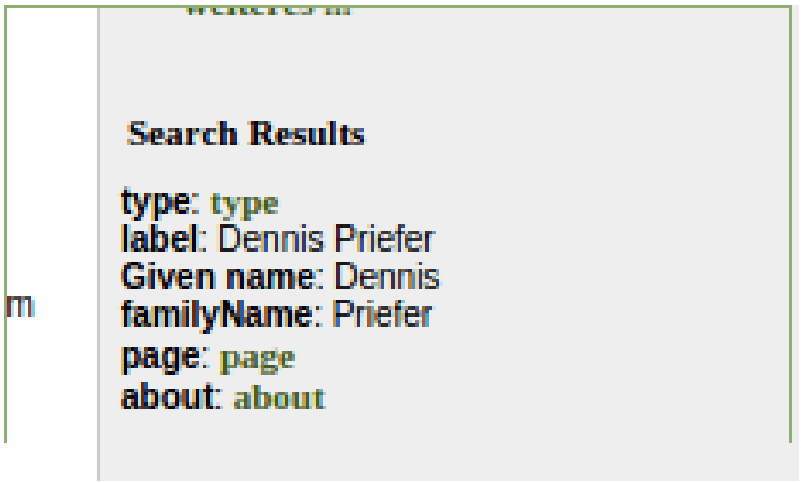

Figure 4.7.: Prototype of search results module

instance of a class that implements this interface, running operation will call the custom operation function instead of the default one. (Listing 4.1)

After the search operation completed, the success function of AJAX call changes the html content of Joomla! module and shows the results. (Figure 4.7)

5. Conclusion

5.1. Summary and Outlook

This work proposes an approach to enhance web content management systems with semantic features. The Interactive Knowledge Stack architecture have been taken as a reference and it is adapted to collaborate with web content management systems. IKS suggests different aspects for having content management systems to gain semantic properties. However the mechanism of communication between semantic back-end. [Gon12] introduces a method for JCR/CMIS compliant content management systems.

One challenge this work addressed about web content management systems was that they didn't have any standart way to handle data and content. This makes it difficult to define a standart mechanism for interaction. In order to create such standart mechanism one can benefit the fact that most of those use SQL to manipulate relational data and by nature, HTTP as access protocol. Since most of the semantic back-end services provide RESTful APIs, the challenge has been reduced to defining an interface to allow web content management systems to interact with those RESTful APIs. To achieve this goal, The SIWCMS, set of abstract classes and interfaces that defines a contract for this interaction has been created (Figure 5.1).

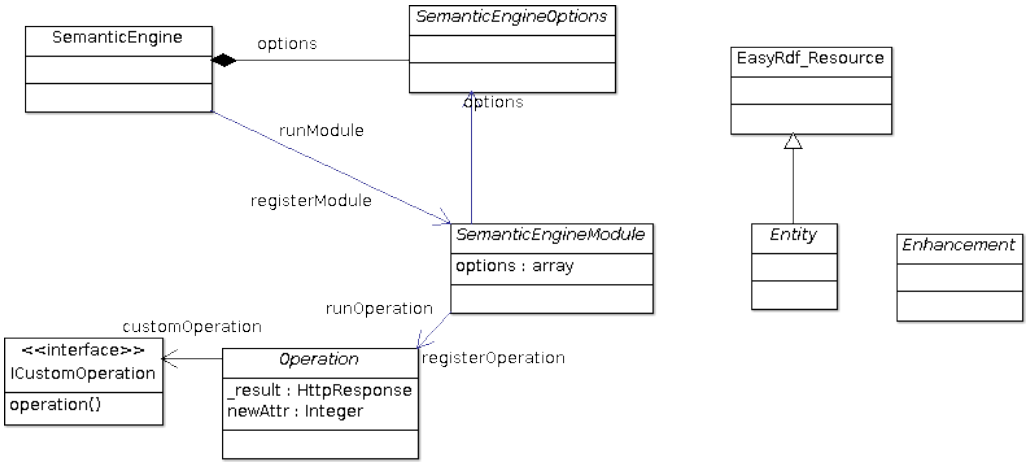

Figure 5.1.: SIWCMS class structure

For the implementation of the suggested approach, Apache Stanbol and an external triple store as semantic back-end have been chosen. Apache Stanbol offers a set of modules that matches with layers of IKS. Even though this work gives examples by using Apache Stanbol for the implemented semantic features such as enhancement and search, it can be adapted any kind of engine that provides RESTful API.

As an example content management system, the Joomla! CMS based website of Technische Hochschule Mittelhessen University of Applied Sciences, Mathematics, Natural Science and Informatics Deparment has been used and the semantic elements has been implemented as Joomla! extensions.

The overall system is shown in Figure 5.2. The semantic index module, can be part of the triple store or the semantic engine, also can be a completely separate module. The SIWCMS is distributed as a library, so it can be included any kind of web content management system with a little amount of programming effort. It is also predicted that migration between different CMS version would not be so painful as long as there is no drastic changes.

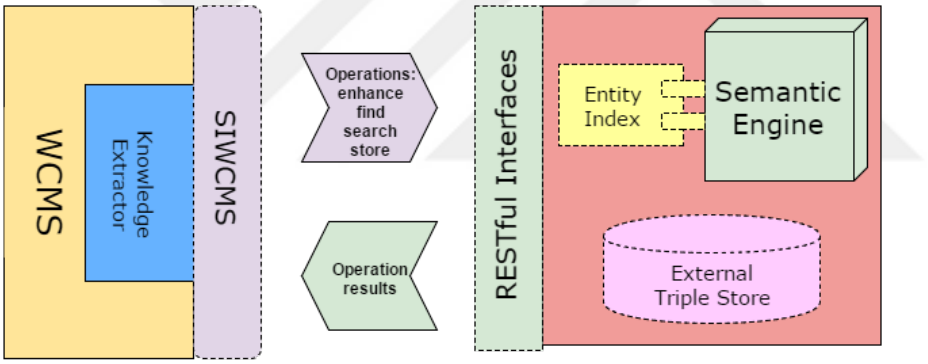

Figure 5.2.: General view of the semantically enhanced web content management system

5.2. Discussion

Since the work is mostly conceptual, there is no standart way to evaluate its performance, but it proposes an approach for bringing content management systems and semantic web together. Naturally this is not the first effort ever put on this subject, but it differs from the others:

- It is not specific to any content management system. The SIWCMS classes have no dependency in a specific content management system.
- Focused on web content management systems
- It does not offer a new tool, but a contract for accessing such semantic tools
- It allows users to make the metadata and data distinction according to their needs

[CDC+09] points out a "semantic hijacking problem". This term essentially implies that the user defined concepts might affect the semantics of existing vocabularies. They offer a a sub-classing method to avoid this situation. However, in many cases even sub-classing might not cover the intended semantics. Therefore it is suggested to allow users to benefit from knowledge organization system properties. (see SKOS[1])

There are of course some weaknesses of this approach that is predicted:

- **Too many things to configure:** Having a seperate triple store and a semantic engine can be painful to configure and maintain.
- **Need for expert knowledge:** Users still needs to know about the semantics of existing vocabularies in order to create high quality data.
- **Lack of an intuitive user interface:** Creating mapping queries can be challenging for users who are not highly skilled at writing SQL queries.

5.3. Further research

As a short term (2-4 weeks) goal, improving the prototype and creating a fully working implementation would be a non-trivial. An implementation on different versions of a CMS and on different content management systems would be good test for applicability of interface.

[1] http://www.w3.org/TR/2008/WD-skos-reference-20080829/skos.html

As a longer term goal, further research involves mostly challenging the weaknesses mentioned in Section 5.2. A survey based on user experiment to measure the advantages of semantically enhanced content management systems over conventional ones can be designed. The metric framework for this empirical work needs to be well defined. Definition of these metrics may vary according to evaluated content management system. Alongside the some general metrics that can be defined for all content management systems, there should be some CMS specific metrics, since every CMS has a different way to handle content.

Bibliography

[abo] About Us | IKS - The Semantic CMS Community - Open Source. URL: http://iks-project.eu/about-us.

[ADL+09] Sören Auer, Sebastian Dietzold, Jens Lehmann, Sebastian Hellmann, and David Aumueller. Triplify: Light-weight Linked Data Publication from Relational Databases. In *Proceedings of the 18th International Conference on World Wide Web*, WWW '09, pages 621–630, New York, NY, USA, 2009. ACM. URL: http://doi.acm.org/10.1145/1526709.1526793, doi:10.1145/1526709.1526793.

[AH08] Dean Allemang and James Hendler. *Semantic Web for the Working Ontologist: Effective Modeling in RDFS and OWL*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2008.

[apa] Apache Stanbol - Apache Entityhub. URL: https://stanbol.apache.org/docs/trunk/components/entityhub/.

[bab] BabelNet™ | A very large multilingual encyclopedic dictionary and semantic network. URL: http://babelnet.org/.

[BG13] Reto Bachmann-Gmur. *Instant Apache Stanbol*. Packt Publishing, 2013. URL: http://public.eblib.com/choice/publicfullrecord.aspx?p=1336421.

[BHBL09] Christian Bizer, Tom Heath, and Tim Berners-Lee. Linked Data - The Story So Far. *Int. J. Semantic Web Inf. Syst.*, 5(3):1–22, 2009. doi:10.4018/jswis.2009081901.

[BL09] Tim Berners-Lee. Linked Data - Design Issues, June 2009. URL: http://www.w3.org/DesignIssues/LinkedData.html.

[BLHL01] Tim Berners-Lee, James Hendler, and Ora Lassila. The Semantic Web. *Scientific American*, 284(5):34–43, May 2001. URL: http://www.sciam.com/article.cfm?articleID=00048144-10D2-1C70-84A9809EC588EF21.

[Bur08] Okan Bursa. Analysis of Semantic Web Portals Technologies and Developing an Application. Master's thesis, Ege University, Department of Computer Science, December 2008.

[CDC+09] Stéphane Corlosquet, Renaud Delbru, Tim Clark, Axel Polleres, and Stefan Decker. Produce and Consume Linked Data with Drupal! In Abraham Bernstein, David R. Karger, Tom Heath, Lee Feigenbaum, Diana Maynard, Enrico Motta, and Krishnaprasad Thirunarayan, editors, *The Semantic Web - ISWC 2009*, number 5823 in Lecture Notes in Computer Science, pages 763–778. Springer Berlin Heidelberg, 2009. URL: http://link.springer.com/chapter/10.1007/978-3-642-04930-9_48.

[CN11] Fabian Christ and Benjamin Nagel. A reference architecture for semantic content management systems. In *In Proceeding of the Enterprise Modelling and Information Systems Architectures Workshop 2011 (EMISA'11), volume P-190*, pages 135–148, 2011.

[dat] Database Description « WordPress Codex. URL: https://codex.wordpress.org/Database_Description.

[dru] Drupal - Open Source CMS | Drupal.org. URL: https://www.drupal.org/.

[eas] EasyRdf - RDF Library for PHP. URL: http://www.easyrdf.org/.

[Gon12] Suat Gonul. *Enhancing Content Management Systems with Semantic Capabilities.* MSc Thesis, Middle East Technical University, Ankara, 2012. URL: https://etd.lib.metu.edu.tr/upload/12614567/index.pdf.

[Gru95] Thomas R. Gruber. Toward principles for the design of ontologies used for knowledge sharing? *International Journal of Human-Computer Studies*, 43(5–6):907–928, November 1995. URL: http://www.sciencedirect.com/science/article/pii/S1071581985710816, doi:10.1006/ijhc.1995.1081.

[HHHR14] Timm Heuss, Bernhard Humm, Christian Henninger, and Thomas Rippl. A Comparison of NER Tools W.R.T. A Domain-specific Vocabulary. In *Proceedings of the 10th International Conference on Semantic Systems*, SEM '14, pages 100–107, New York, NY, USA, 2014. ACM. URL: http://doi.acm.org/10.1145/2660517.2660520, doi:10.1145/2660517.2660520.

[HRG11] Matthias Hert, Gerald Reif, and Harald C. Gall. A Comparison of RDB-to-RDF Mapping Languages. In *Proceedings of the 7th International Conference on Semantic Systems*, I-Semantics '11, pages 25–32, New York, NY, USA, 2011. ACM. URL: http://doi.acm.org/10.1145/2063518.2063522, doi:10.1145/2063518.2063522.

[htt] Httpful: The REST Friendly PHP HTTP Client Library. URL: http://phphttpclient.com/.

[jooa] Joomla CMS Architecture. URL: http://www.slideshare.net/dustinczysz/joomla-cms-architecture.

[joob] Joomla! The CMS Trusted By Millions for their Websites. URL: http://www.joomla.org/.

[jow] jowlproject - Implementing the Semantic Web in Joomla! - Google Project Hosting. URL: https://code.google.com/p/jowlproject/.

[KVV06] Markus Krötzsch, Denny Vrandečić, and Max Völkel. Semantic MediaWiki. In Isabel Cruz, Stefan Decker, Dean Allemang, Chris Preist, Daniel Schwabe, Peter Mika, Mike Uschold, and Lora M. Aroyo, editors, *The Semantic Web - ISWC 2006*, number 4273 in Lecture Notes in Computer Science, pages 935–942. Springer Berlin Heidelberg, 2006. URL: http://link.springer.com/chapter/10.1007/11926078_68.

[LAD+10] G. B. Laleci, G. Aluc, A. Dogac, A. Sinaci, O. Kilic, and F. Tuncer. A semantic backend for content management systems. *Knowledge-Based Systems*, 23(8):832–843, December 2010. URL: http://www.sciencedirect.com/science/article/pii/S0950705110000894, doi:10.1016/j.knosys.2010.05.008.

[lin] The Linking Open Data cloud diagram. URL: http://lod-cloud.net/.

[NGS10] José L. Navarro-Galindo and José Samos. Manual and Automatic Semantic Annotation of Web Documents: The FLERSA Tool. In *Proceedings of the 12th International Conference on Information Integration and Web-based Applications & Services*, iiWAS '10, pages 542–549, New York, NY, USA, 2010. ACM. URL: http://doi.acm.org/10.1145/1967486.1967570, doi:10.1145/1967486.1967570.

[NHL+11] Axel-Cyrille Ngonga Ngomo, Norman Heino, Klaus Lyko, René Speck, and Martin Kaltenböck. SCMS – Semantifying Content Management Systems. In Lora Aroyo, Chris Welty, Harith Alani, Jamie Taylor, Abraham Bernstein, Lalana Kagal, Natasha Noy, and Eva Blomqvist, editors, *The Semantic Web – ISWC 2011*, number 7032 in Lecture Notes in Computer Science, pages 189–204. Springer Berlin Heidelberg, 2011. URL: http://link.springer.com/chapter/10.1007/978-3-642-25093-4_13.

[php] php-sql-parser - A pure PHP SQL (non validating) parser w/ focus on MySQL dialect of SQL - Google Project Hosting. URL: https://code.google.com/p/php-sql-parser/.

[rdf] RDF 1.1 Primer. URL: http://www.w3.org/TR/2014/NOTE-rdf11-primer-20140624/.

[RMD13] Delip Rao, Paul McNamee, and Mark Dredze. Entity Linking: Finding Extracted Entities in a Knowledge Base. In Thierry Poibeau, Horacio Saggion, Jakub Piskorski, and Roman Yangarber, editors, *Multi-source, Multilingual Information Extraction and Summarization*, Theory and Applications of Natural Language Processing, pages 93–115. Springer Berlin Heidelberg, 2013. URL: http://link.springer.com/chapter/10.1007/978-3-642-28569-1_5.

[SBJC14] Max Schmachtenberg, Christian Bizer, Anna Jentzsch, and Richard Cyganiak. Linking Open Data cloud diagram, 2014. http://lod-cloud.net/.

[SG12] Ali Anil Sinaci and Suat Gonul. Semantic Content Management with Apache Stanbol. In *The Semantic Web: ESWC 2012 Satellite Events*, pages 371–375. Springer, 2012.

[spa] SPARQL 1.1 Overview. URL: http://www.w3.org/TR/2013/REC-sparql11-overview-20130321/.

[TCRS07] Thanh Tran, Philipp Cimiano, Sebastian Rudolph, and Rudi Studer. Ontology-Based Interpretation of Keywords for Semantic Search. In Karl Aberer, Key-Sun Choi, Natasha Noy, Dean Allemang, Kyung-Il Lee, Lyndon Nixon, Jennifer Golbeck, Peter Mika, Diana Maynard, Riichiro Mizoguchi, Guus Schreiber, and Philippe Cudré-Mauroux, editors, *The Semantic Web*, number 4825 in Lecture Notes in Computer Science, pages 523–536. Springer Berlin Heidelberg, 2007. URL: http://link.springer.com/chapter/10.1007/978-3-540-76298-0_38.

[usa] Usage Statistics and Market Share of Content Management Systems for Websites, August 2015. URL: http://w3techs.com/technologies/overview/content_management/all.

[wha] What is Semantic Search? - Definition from Techopedia. URL: http://www.techopedia.com/definition/23731/semantic-search.

[wor] WordPress › Blog Tool, Publishing Platform, and CMS. URL: https://wordpress.org/.

[Zie12] Wolfgang Ziegler. IKS Content Enhancements Demo, December 2012. URL: https://www.drupal.org/project/iksce.

List of Tables

List of Figures

Listings

A. Information About the Third-Party Tools Used in Implementation

Name	Version	Remarks
Apache Stanbol	1.0.0	Java
LAMPP Stack	5.x	PHP 5.x, MYSQL
Apache Jena Fuseki	2.3.0	Triple Store
Linux Mint	17.0	Operating System
PHPStorm	8.x	IDE
Joomla!	2.5	CMS
JQuery UI	1.11.4	JQuery Library
Httpful	2.3.0	PHP HTTP Client
EasyRDF	0.8.0	RDF Library

B. CD-ROM

The following content is part of the CD-ROM:

- PDF formatted version of the thesis
- Pre-configured external triple store
- Joomla! 2.5 instance with necessary extensions
- Pre-configured Apache Stanbol semantic engine

Declaration of Authorship

I hereby declare that I have written this thesis without any help from others and without the use of documents and aids other than those stated. I have mentioned explicitly all used sources of information and I have cited them correctly according to established academic citation rules.

(Place, date) (Signature)