SCMS – Semantifying Content Management Systems

Axel-Cyrille Ngonga Ngomo[1], Norman Heino[1], Klaus Lyko[1], René Speck[1], Martin Kaltenböck[2]

[1] University of Leipzig
AKSW Group
Johannisgasse 26, 04103 Leipzig
[2] Semantic Web Company
Lerchenfeldergürtel 43
A-1160 Vienna

Abstract. The migration to the Semantic Web requires from CMS that they integrate human- and machine-readable data to support their seamless integration into the Semantic Web. Yet, there is still a blatant need for frameworks that can be easily integrated into CMS and allow to transform their content into machine-readable knowledge with high accuracy. In this paper, we describe the SCMS (Semantic Content Management Systems) framework, whose main goals are the extraction of knowledge from unstructured data in any CMS and the integration of the extracted knowledge into the same CMS. Our framework integrates a highly accurate knowledge extraction pipeline. In addition, it relies on the RDF and HTTP standards for communication and can thus be integrated in virtually any CMS. We present how our framework is being used in the energy sector. We also evaluate our approach and show that our framework outperforms even commercial software by reaching up to 96% F-score.

1 Introduction

Content Management Systems (CMS) encompass most of the information available on the document-oriented Web (also referred to as Human Web). Therewith, they constitute the interface between humans and the data on the Web. Consequently, one of the main tasks of CMS has always been to make their content as easily processable for humans as possible. Still, with the migration from the document-oriented to the Semantic Web, there is an increasing need to insert machine-readable data into the content of CMS so as to enable the seamless integration of their content into the Semantic Web. Given the sheer volume of data available on the document-oriented Web, the insertion of machine-readable data must be carried out (semi-) automatically. The frameworks developed for the purpose of automatic knowledge extraction must therefore be *accurate* (i. e., display high F-scores) so as to ensure that humans need to curate a minimal amount of the knowledge extracted automatically. This criterion is central for the use of automatic knowledge extraction, as approaches with a low recall lead

to humans having to find the false negatives[3] by hand, while a low precision forces the same humans to have to continually check the output of the knowledge extraction framework. A further criterion that determines the usability of a knowledge extraction framework is its *flexibility*, i. e., how easy it is to integrate this framework in CMS. This criterion is of high importance as the current CMS landscape consists of hundreds of very heterogeneous frameworks implemented in dozens of different languages[4].

In this paper, we describe the SCMS framework[5]. The main goal of our framework is to allow the extraction of structured data (i. e., RDF) out of the unstructured content of CMS, the linking of this content with the Web of Data and the integration of this wealth of knowledge back into the CMS. SCMS relies exclusively on RDF messages and simple Web protocols for its integration into existing CMS and the processing of their content. Thus, it is *highly flexible* and can be used with virtually any CMS. In addition, the underlying approach implements a *highly accurate* knowledge extraction pipeline that can be configured easily for the user's purposes. This pipeline allows to merge and improve the results of state-of-the-art tools for information extraction, to manually post-process the results at will and to integrate the extracted knowledge into CMS, for example as RDFa. The main contributions of this paper are the following:

1. We present the architecture of our approach and show that it can be integrated easily in virtually any CMS, provided it offers sufficient hooks into the life-cycle of its managed content items.
2. We give an overview of the vocabularies we use to represent the knowledge extracted from CMS.
3. We present how our approach is being used in a use case centered around renewable energy.
4. We evaluate our approach against a state-of-the-art commercial system for knowledge extraction in two practical use cases and show that we outperform the commercial system with respect to F-score while reaching up to 96% F-score on the extraction of locations.

The rest of this paper is structured as follows: We start by giving an overview of related work from the NLP and the Semantic Web community in Section **??**. Thereafter, we present the SCMS framework (Section **??**) and its main components (Section **??**) as well as the vocabularies they use. Subsequently, we epitomize the renewable energy use case within which our framework is being deployed in Section **??**. Section **??** then presents the results of an evaluation of our framework in two use cases against an enterprise commercial system (CS) whose name cannot be revealed for legal reasons. Finally, we give an overview of our future work and conclude.

[3] i. e., the entities and relations that were not found by the software

[4] A list of CMS on the market can be found at `http://en.wikipedia.org/wiki/List_of_content_management_systems`

[5] `http://www.scms.eu`

2 Related Work

Information Extraction is the backbone of knowledge extraction and is one of the core tasks of NLP. Three main categories of NLP tools play a central role during the extraction of knowledge from text: Keyphrase Extraction (KE), Named Entity Recognition (NER) and relation extraction (RE). The automatic detection of keyphrases (i. e., multi-word units or text fragments that capture the essence of a document) has been an important task of NLP for decades. Still, due to the very ambiguous definition of what an appropriate keyphrase is, current approaches to the extraction of keyphrases still display low F-scores [**?**]. According to [**?**], the majority of the approaches to KE implement combinations of statistical, rule-based or heuristic methods [**?**,**?**] on mostly document [**?**], keyphrase [**?**] or term cohesion features [**?**].

NER aims to discover instances of predefined classes of entities (e. g., persons, locations, organizations or products) in text. Most NER tools implement one of three main categories of approaches: dictionary-based [**?**,**?**], rule-based [**?**,**?**] and machine-learning approaches [**?**]. Nowadays, the methods of choice are borrowed from supervised machine learning when training examples are available [**?**,**?**,**?**]. Yet, due to scarcity of large domain-specific training corpora, semi-supervised [**?**,**?**] and unsupervised machine learning approaches [**?**,**?**] have also been used for extracting named entities from text.

The extraction of relations from unstructured data builds upon work for NER and KE to determine the entities between which relations might exist. Some early work on pattern extraction relied on supervised machine learning [**?**]. Yet, such approaches demanded large amount of training data. The subsequent generation of approaches to RE aimed at bootstrapping patterns based on a small number of input patterns and instances [**?**,**?**]. Newer approaches aim to either collect redundancy information from the whole Web [**?**] or Wikipedia [**?**,**?**] in an unsupervised manner or to use linguistic analysis [**?**,**?**] to harvest generic patterns for relations.

In addition to the work done by the NLP community, several tools and frameworks have been developed explicitly for extracting RDF and RDFa out of NL [**?**]. For example, the Firefox extension Piggy Bank [**?**] allows to extract RDF from web pages by using screen scrapers. The RDF extracted from these webpages is then stored locally in a Sesame store. The data being stored locally allows the user to merge the data extracted from different websites to perform semantic operations. More recently, the Drupal extension OpenPublish[6] was released. The aim of this extension is to support content publishers with the automatic annotation of their data. For this purpose, OpenPublish utilizes the services provided by OpenCalais[7] to annotate the content of news entries. Epiphany [**?**] implements a service that annotates web pages automatically with entities found in the Linked Data Cloud. Apache Stanbol[8] implements similar functionality on

[6] `http://www.openpublish.com`
[7] `http://www.opencalais.org`
[8] `http://incubator.apache.org/stanbol`

a larger scale by providing synchronous RESTful interfaces that allow Content Management Systems to extract annotations from text.

The main drawback of current frameworks is that they either focus on one particular task (e. g., finding named entities in text) or make use of NLP algorithms without improving upon them. Consequently, they have the same limitations as the NLP approaches discussed above. To the best of our knowledge, our framework is the first framework designed explicitly for the purposes of the Semantic Web that combines flexibility with accuracy. The flexibility of the SCMS has been shown by its deployment on Drupal[9], Typo3[10] and conX[11]. In addition, our framework is able to extract RDF from NL with an accuracy superior to that of commercial systems as shown by our evaluation. Our framework also provides a machine-learning module that allows to tailor it to new domains and classes of named entities. Moreover, SCMS provides dedicated interfaces for interacting (e. g., editing, querying, merging) with the triples extracted, making it usable in a large number of domains and use cases.

3 The SCMS Framework

An overview of the architecture behind SCMS is given in Figure **??**. The framework consists of two layers: an *orchestration and curation* layer and an *extraction and storage* layer. The CMS that is to be extended with semantic capabilities resides upon our framework and must be extended minimally via a *CMS wrapper*. This extension implements the in- and output behavior of the CMS and communicates exclusively with the first layer of our framework, thus making the components of the extraction and storage layer of our framework swappable without any drawback for the users.

The overall goal of the first layer of the SCMS framework is to coordinate the access to the data. It consists of two tools: the orchestration service and the data wiki OntoWiki. The *orchestration service* is the input gate of SCMS. It receives the data that is to be annotated as a RDF message that abides by the vocabulary presented in Section **??** and returns the results of the framework to the endpoint specified in the RDF message it receives. *OntoWiki* provides functionality for the manual curation of the results of the knowledge extraction process and manages the data flow to the *triple store* Virtuoso[12], the first component of the *extraction and storage layer*. In addition to a triple store, the second layer contains the Federated knOwledge eXtraction Framework FOX[13], that uses machine learning to combine and improve upon the results of NLP tools as well as converts these results into RDF by using the vocabularies displayed in Section **??**. Virtuoso also contains a crawler that allows to retrieve supplementary knowledge from the Web and link it to the information already available in the CMS by integrating it into

[9] `http://drupal.org`
[10] `http://typo3.org`
[11] `http://conx.at`
[12] `http://virtuoso.openlinksw.com`
[13] `http://fox.aksw.org`

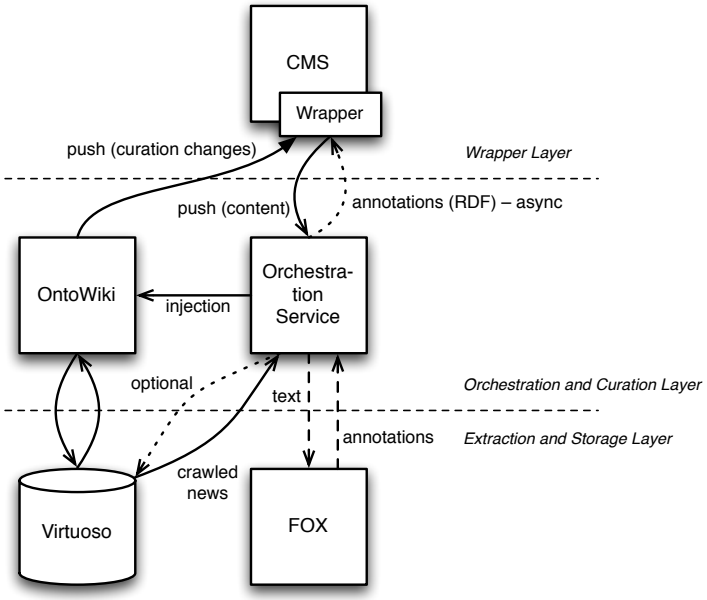


Fig. 1. Architecture and paths of communication of components in the SCMS content semantification system.

the CMS. In the following, we present the central components of the SCMS stack in more detail.

4 Tools and Vocabularies

In this section we describe the main components of the SCMS stack and how they fit together. As running example, we use a hypothetical content item contained in a Drupal CMS. This node (in Drupal terminology) that consists of two parts:

- The `title` "*Prometeus*" and
- a `body` that contains the sentence "*The company Prometeus is an energy provider located in the capital of Hungary, i. e., Budapest.*".

Only the body to the content item is to be annotated by the SCMS stack. Note that for reasons of brevity, we will only show the results of the extraction of named entities. Yet, SCMS can also extract keywords, keyphrases and relations.

4.1 Wrapper

A CMS wrapper (short wrapper) is a component that is tightly integrated into a CMS (see Figure **??**) and whose role is to ensure the communication between the

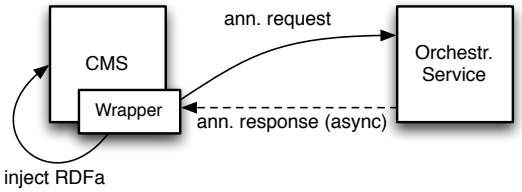


Fig. 2. Architecture of communication between wrapper, CMS and orchestration service.

CMS and the orchestration module of our framework. In this respect, a wrapper has to fulfill three main tasks:

1. *Request generation*: Wrappers usually register for change events to the CMS editing system. Whenever a document has been edited, they generate an annotation request that abides by the vocabulary depicted in Figure **??**. This request is then sent to the orchestration service.
2. *Response receipt*: Once the annotation has been carried out, the annotation results are sent back to the wrapper. The second of the wrapper's main tasks is consequently to react to those annotation responses and to store the annotations to the document appropriately (e. g., in a triple store). Since the annotation results are sent back asynchronously (i. e., in a separate request), the wrapper must provide a callback URL for this purpose.
3. *Data processing*: Once the data have been received and stored, wrappers usually integrate the annotations into the content items that were processed by the CMS. The integration of annotations is most commonly carried out by "injecting" the annotations as RDFa into the document's HTML rendering. The data injection is mostly realized by registering to document viewing events in the respective CMS and writing the RDFa from the wrapper's local triple store into the content items that are being viewed.

An example of a wrapper request for our example is shown in Listing **??**. The `content:encoded` of the Drupal node `http://example.com/drupal/node/10` is to be annotated by FOX. In addition, the whole node is to be stored in the triple store for the purpose of manual processing. Note that the wrapper can choose not to send portions of the content item that are not to be stored in the triple store, e. g., private data. In addition, note that the description of a document is not limited to certain properties or to a certain number thereof, which ensures the high level of flexibility of the SCMS stack. Moreover, the RDF data extracted by SCMS can be easily merged with any structured information provided natively by the CMS (i.e., metadata such as author information). Consequently, SCMS enables CMS that already provide metadata as RDF to answer complex questions that combine data and metadata, e.g., `Which authors wrote documents that are related to Budapest?`

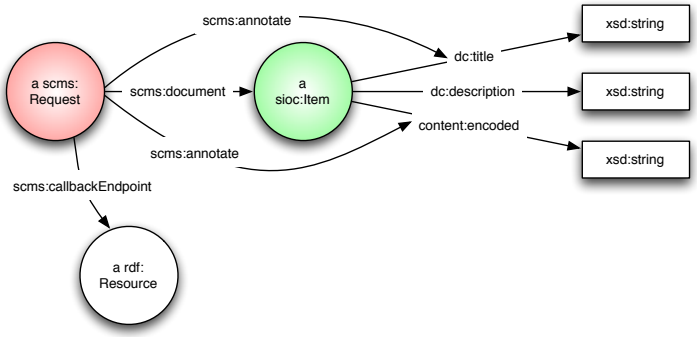


Fig. 3. Vocabulary used by the wrapper requests.

```
@prefix content: <http://purl.org/rss/1.0/modules/content/> .
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix sioc: <http://rdfs.org/sioc/ns#> .
@base <http://ns.aksw.org/scms/> .

<http://example.com/wrapperRequest/1> a <Request> ;
    <document> <http://example.com/drupal/node/10> ;
    <callbackEndpoint> <http://example.com/wrapper> ;
    <annotate> content:encoded .

<http://example.com/drupal/node/10> a sioc:Item ;
    dc:title "Prometeus" ;
    content:encoded "The company Prometeus is an energy provider located in
        the capital of Hungary, i.e., Budapest." .
```

Listing 1. Example annotation request as sent by the Drupal wrapper.

4.2 Orchestration Service

The main tasks of the orchestration service are to capture state information and to distribute the data across SCMS' layers. The first of the tasks is due to the FOX framework having been designed to be stateless. The orchestration service captures state information by splitting up each document-based annotation requests by a wrapper into several property-based annotation requests that are sent to FOX. In our example, the orchestration service detects that solely the `content:encoded` property is to be annotated. Then, it reads the content of that property from the wrapper request and generates the annotation request "*The company Prometeus is an energy provider located in the capital of Hungary, i. e., Budapest.*" for FOX. Note that while this property-based annotation request consists exclusively of text or HTML and does not contain any RDF, the response returned by FOX is a RDF document serialized in Turtle or RDF/XML.

The annotation results returned by FOX are combined by the orchestration service into the annotation response. Therewith, the relation between the input document and the annotations extracted by FOX is re-established. When

all annotations for a particular request have been received and combined, the annotation response is sent back to the wrapper via the provided callback URL. In addition, the results sent back to the wrapper are stored in OntoWiki to facilitate the curation of annotations extracted automatically. The annotation response generated by the orchestration service for our example is shown in Listing **??**. It relies upon the output sent by FOX. The exact meaning of the predicates used by FOX and forwarded by the orchestration service are explained in Section **??**

```
@prefix scmsann:  <http://ns.aksw.org/scms/annotations/> .
@prefix ctag:     <http://commontag.org/ns#> .
@prefix xsd:      <http://www.w3.org/2001/XMLSchema#> .
@prefix rdf:      <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix ann:      <http://www.w3.org/2000/10/annotation-ns#> .
@prefix scms:     <http://ns.aksw.org/scms/> .

[]     a       ann:Annotation , scmsann:LOCATION ;
       scms:annotates <http://example.com/drupal/node/10> ;
       scms:property <http://purl.org/rss/1.0/modules/content/encoded> ;
       scms:beginIndex "70"^^xsd:int ;
       scms:endIndex "77"^^xsd:int ;
       scms:means <http://dbpedia.org/resource/Hungary> ;
       scms:source <http://ns.aksw.org/scms/tools/FOX> ;
       ann:body "Hungary"^^xsd:string .

[]     a       ann:Annotation , scmsann:ORGANIZATION ;
       scms:annotates <http://example.com/drupal/node/10> ;
       scms:property <http://purl.org/rss/1.0/modules/content/encoded> ;
       scms:beginIndex "12"^^xsd:int ;
       scms:endIndex "21"^^xsd:int ;
       scms:means <http://scms.eu/Prometeus> ;
       scms:source <http://ns.aksw.org/scms/tools/FOX> ;
       ann:body "Prometeus"^^xsd:string .

[]     a       ann:Annotation , scmsann:LOCATION ;
       scms:annotates <http://example.com/drupal/node/10> ;
       scms:property <http://purl.org/rss/1.0/modules/content/encoded> ;
       scms:beginIndex "85"^^xsd:int ;
       scms:endIndex "93"^^xsd:int ;
       scms:means <http://dbpedia.org/resource/Budapest> ;
       scms:source <http://ns.aksw.org/scms/tools/FOX> ;
       ann:body "Budapest"^^xsd:string .
```

Listing 2. Example annotation response as sent by the orchestration service.

4.3 FOX

The FOX framework is a stateless and extensible framework that encompasses all the NLP functionality necessary to extract knowledge from the content of CMS. Its architecture consists of three layers as shown in Figure **??**.

FOX takes text or HTML as input. This data is sent to the *controller layer*, which implements the functionality necessary to clean the data, i.e., remove HTML and XML tags as well as further noise. Once the data has been cleaned,

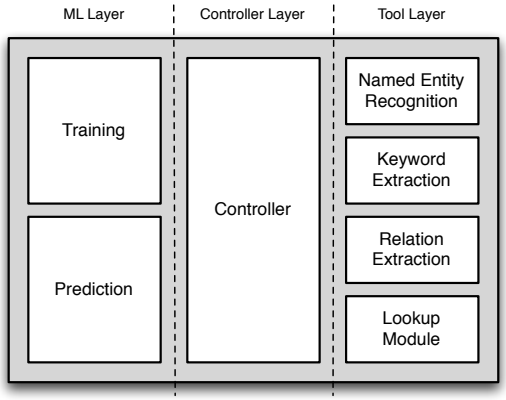


Fig. 4. Architecture of the FOX framework.

the controller layer begins with the orchestration of the tools in the *tool layer*. Each of the tools is assigned a thread from a thread pool, so as to maximize usage of multi-core CPUs. Every thread runs its tool and generates an event once it has completed its computation. In the event that a tool does not complete after a set time, the corresponding thread is terminated. So far, FOX integrates tools for KE, NER and RE. The KE is realized by PoolParty[14] for extracting keywords from a controlled vocabulary, KEA[15] and the Yahoo Term Extraction service[16] for statistical extraction and several other tools. In addition, FOX integrates the Stanford Named Entity Recognizer[17] [**?**], the Illinois Named Entity Tagger[18] [**?**] and commercial software for NER. The RE is carried out by using the CARE platform[19].

The results from the tool layer are forwarded to the *prediction module* of the *machine-learning layer*. The role of the prediction module is to generate FOX's output based on the output the tools in FOX's backend. For this purpose, it implements several ensemble learning techniques [**?**] with which it can combine the output of several tools. Currently, the prediction module carries out this combination by using a feed-forward neural network. The neural network inserted in FOX was trained by using 117 news articles. It reached 89.21% F-Score in an evaluation based on a ten-fold-cross-validation on NER, therewith outperforming even commercial systems[20].

Once the neural network has combined the output of the tool and generated a better prediction of the named entities, the output of FOX is generated by

[14] `http://poolparty.biz`
[15] `http://www.nzdl.org/Kea/`
[16] `http://developer.yahoo.com/search/content/V1/termExtraction.html`
[17] `http://nlp.stanford.edu/software/CRF-NER.shtml`
[18] `http://cogcomp.cs.illinois.edu/page/software_view/4`
[19] `http://www.digitaltrowel.com/Technology/`
[20] More details on the evaluation are provided at http://fox.aksw.org

using the vocabularies shown in Figure **??**. These vocabularies extend the two broadly used vocabularies Annotea[21] and Autotag [22]. In particular, we added the constructs explicated in the following:

- `scms:beginIndex` denotes the index in a literal value string at which a particular annotation or keyphrase begins;
- `scms:endIndex` stands for the index in a literal value string at which a particular annotation or keyphrase ends;
- `scms:means` marks the URI assigned to a named entity identified for an annotation;
- `scms:source` denotes the provenance of the annotation, i. e., the URI of the tool which computed the annotation or even the system ID of the person who curated or created the annotation and
- `scmsann` is the namespace for the annotation classes, i.e, location, person, organization and miscellaneous.

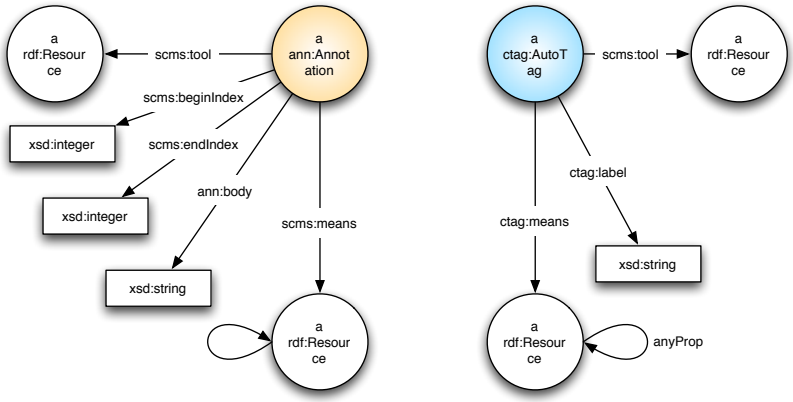
Given that the overhead due to the merging of the results via the neural network is of only a few milliseconds and thank to the multi-core architecture of current servers, FOX is almost as time-efficient as state-of-the-art tools. Still, as our evaluation shows, these few milliseconds overhead can lead to an increase of more than 13% F-Score (see Section **??**). The output of FOX for our example is shown in Listing **??**. This is the output that is forwarded to the orchestration service, which adds provenance information to the RDF before sending an answer to the callback URI provided by the wrapper. By these means, we ensure that the wrapper can write the RDFa in the write segment of the item content.

4.4 OntoWiki

OntoWiki is a semantic data wiki [**?**] that was designed to facilitate the browsing and editing RDF knowledge bases. Its browsing features range from arbitrary concept hierarchies to facet-based search and query building interfaces. Semantic content can be created and edited by using the RDFauthor system which has been integrated in OntoWiki [**?**].

OntoWiki plays two key roles within the SCMS stack. First, it serves as entry point for the triple store. This allows for the triple store to be exchanged without any drawback for the user, leading to an easy customization of our stack. In addition, OntoWiki plays the role of an annotation consolidation and curation tool and is consequently the center of the curation pipeline. To ensure that OntoWiki is always up-to-date, the orchestration service sends its annotation responses to both OntoWiki and the wrapper's callback URI. Thus, OntoWiki is also aware of the wrapper (i. e., its callback URI) and can send the results of any manual curation process back to wrapper. Note that manually curated annotations are saved with a different (if manually created) or supplementary (if manually curated) value in their `scms:source` property. This gives consuming

[21] `http://www.w3.org/2000/10/annotation-ns#`
[22] `http://commontag.org/ns#`



(a) named entity annotation (b) keyword annotation

Fig. 5. Vocabularies used by FOX for representing named entities (a) and keywords (b)

tools (e. g., wrappers) a chance to assign higher trust values to those annotations. In addition, if a new extraction run is performed on the same document, manually created and curated annotations can be kept for further use. Note that the crawler in Virtuoso can be used to fetch even more data pertaining to the annotations computed by FOX. This data can be sent directly to FOX and inserted in Virtuoso so as to extend the knowledge base for the CMS.

5 Use Case

The SCMS framework is being deployed in the renewable energy sector. The renewable energy and energy efficiency sector requires a large amount of up-to-date and high-quality information and data so as to develop and push the area of clean energy systems worldwide. This information, data and knowledge about clean energy technologies, developments, projects and laws per country worldwide helps policy and decision makers, project developers and financing agencies to make better decisions on investments as well as clean energy projects to set up. The REEEP – the Renewable Energy and Energy Efficiency Partnership[23] is a non-governmental organization that provides the aforementioned information to the respective target groups around the globe. For this purpose, REEEP has developed the `reegle.info` Information Gateway on Renewable Energy and Energy Efficiency [24] that offers country profiles on clean energy, an Actors Catalog that contains the relevant stakeholders in the field per country. Furthermore, it supplies energy statistics and potentials as well as news on clean energy.

[23] `http://www.reeep.org`
[24] `http://www.reegle.info`

```
@prefix scmsann:  <http://ns.aksw.org/scms/annotations/> .
@prefix ctag:     <http://commontag.org/ns#> .
@prefix xsd:      <http://www.w3.org/2001/XMLSchema#> .
@prefix rdf:      <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix ann:      <http://www.w3.org/2000/10/annotation-ns#> .
@prefix scms:     <http://ns.aksw.org/scms/> .

[]     a       ann:Annotation , scmsann:LOCATION ;
       scms:beginIndex "70"^^xsd:int ;
       scms:endIndex "77"^^xsd:int ;
       scms:means <http://dbpedia.org/resource/Hungary> ;
       scms:source <http://ns.aksw.org/scms/tools/FOX> ;
       ann:body "Hungary"^^xsd:string .

[]     a       ann:Annotation , scmsann:ORGANIZATION ;
       scms:beginIndex "12"^^xsd:int ;
       scms:endIndex "21"^^xsd:int ;
       scms:means <http://scms.eu/Prometeus> ;
       scms:source <http://ns.aksw.org/scms/tools/FOX> ;
       ann:body "Prometeus"^^xsd:string .

[]     a       ann:Annotation , scmsann:LOCATION ;
       scms:beginIndex "85"^^xsd:int ;
       scms:endIndex "93"^^xsd:int ;
       scms:means <http://dbpedia.org/resource/Budapest> ;
       scms:source <http://ns.aksw.org/scms/tools/FOX> ;
       ann:body "Budapest"^^xsd:string .
```

Listing 3. Annotations as returned by FOX in Turtle format.

The motivation behind applying SCMS to the REEEP data was to facilitate the integration of this data in semantic applications to support efficient decision making. To achieve this goal, we aimed to expand the `reegle.info` information gateway by adding RDFa to the unstructured information available on the website and by making the same triples available via a SPARQL endpoint. For our current prototype, we implemented a CMS wrapper for the Drupal CMS and imported the actors catalog of reegle within in (see Figure **??**). This data was then processed by the SCMS stack as follows: All actors and country descriptions were sent to the orchestration service, which forwarded them to FOX. The RDF data extracted by FOX were sent back to the Drupal Wrapper and written via OntoWiki into Virtuoso. The Drupal wrapper then used the keyphrases to extend the set of tags assigned to the corresponding profile in the CMS. The named entities were integrated in the page by using the positional information returned by FOX. By these means, we made the REEEP data accessible for humans (via the Web page) but also for machines (via OntoWiki's integrated SPARQL endpoint and via the RDFa written in the Web pages).

Our approach also makes the automated integration of novel knowledge sources in REEEP possible. To achieve this goal, several selected sources (web sources, blogs and news feeds) are currently being crawled and then analyzed by FOX to extract structured information out of the masses of unstructured text from the Internet.

Fig. 6. Screenshots of SCMS-enhanced Drupal

6 Evaluation

The usability of our approach depends heavily on the quality of the knowledge returned via automated means. Consequently, we evaluated the quality of the RDFa injected into the REEEP data by measuring the precision and recall of SCMS and compared it with that of a state-of-the-art commercial system (CS) whose name cannot be revealed for legal reasons. We chose CS because it outperformed freely available NER tools such as the Stanford Named Entity Recognizer[25] [**?**] and the Illinois Named Entity Tagger[26] [**?**] in a prior evaluation on a newspaper corpus. Within that evaluation, FOX reached 89.21% F-score and was 14% better than CS w.r.t. F-score[27]. As it can happen that only segments of multi-word units are recognized as being named entities, we followed a token-wise evaluation of the SCMS system. Thus, if our system recognized *United Kingdom of Great Britain* as a `LOCATION` when presented with *United Kingdom of Great Britain and Northern Ireland*, it was scored with 5 true positives and 3 false negatives.

Our evaluation was carried out with two different data sets. In our first evaluation, we measured the performance of both systems on country profiles crawled from the Web, i. e., on information that is to be added automatically to the REEEP knowledge bases. For this purpose, we selected 9 country descriptions randomly and annotated 34 sentences manually. These sentences contained 119

[25] `http://nlp.stanford.edu/software/CRF-NER.shtml`
[26] `http://cogcomp.cs.illinois.edu/page/software_view/4`
[27] More details at `http://fox.aksw.org`

named entities tokens, of which 104 were locations and 15 organizations. In our second evaluation, we aimed at measuring how well SCMS performs on the data that can be found currently in the REEEP catalogue. For this purpose, we annotated 23 actors profiles which consisted of 68 sentences manually. The resulting reference data contained 20 location, 78 organization and 11 person tokens. Note that both data sets are of very different nature as the first contains a large number of organizations and a relatively small number of locations while the second consists mainly of locations.

The results of our evaluation are shown in Table **??**. CS follows a very conservative strategy, which leads to it having very high precision scores of up to 100% in some experiments. Yet, its conservative strategy leads to a recall which is mostly significantly inferior to that of SCMS. The only category within which CS outperforms SCMS is the detection of persons in the actors profile data. This is due to it detecting 6 out of the 11 person tokens in the data set, while SCMS only detects 5. In all other cases, SCMS outperforms CS by up to 13% F-score (detection of organizations in the country profiles data set). Overall, SCMS outperform CS by 7% F-score on country profiles and almost 8% F-score on actors.

		Country Profiles		**Actors Profiles**	
Entity Type	**Measure**	**FOX**	**CS**	**FOX**	**CS**
Location	Precision	98%	100%	83.33%	100%
	Recall	94.23%	78.85%	90%	70%
	F-Score	**96.08**%	88.17%	**86.54**%	82.35%
Organization	Precision	73.33%	100%	57.14%	90.91%
	Recall	68.75%	40%	69.23%	47.44%
	F-Score	**70.97**%	57.14%	**62.72**%	62.35%
Person	Precision	–	–	100%	100%
	Recall	–	–	45.45%	54.55%
	F-Score	–	–	62.5%	**70.59**%
Overall	Precision	93.97%	100%	85.16%	98.2%
	Recall	91.60%	74.79%	70.64%	52.29%
	F-Score	**92.77**%	85.58%	**77.22**%	68.24%

Table 1. Evaluation results on country and actors profiles. The superior F-score for each category is in bold font.

7 Conclusion

In this paper, we presented the SCMS framework for extracting structured data from CMS content. We presented the architecture of our approach and explained how each of its components works. In addition, we explained the vocabularies

utilized by the components of our framework. We presented one use case for the SCMS system, i. e., how SCMS is used in the renewable energy sector.

The SCMS stack abides by the criteria of accuracy and flexibility. The flexibility of our approach is ensured by the combination of RDF messages that can be easily extended and of standard Web communication protocols. The accuracy of SCMS was demonstrated in an evaluation on actor and country profiles, within which SCMS outperformed even commercial software. Our approach can be extended by adding support for negative statements, i. e., statements that are not correct but can be found in different knowledge sources across the data landscape analyzed by our framework. In addition, the feedback generated by users will be integrated in the training of the framework to make it even more accurate over time.

References

1. Benjamin Adrian, Jörn Hees, Ivan Herman, Michael Sintek, and Andreas Dengel. Epiphany: Adaptable rdfa generation linking the web of documents to the web of data. In *EKAW*, pages 178–192, 2010.
2. Eugene Agichtein and Luis Gravano. Snowball: Extracting relations from large plain-text collections. In *In ACM DL*, pages 85–94, 2000.
3. R. Amsler. Research towards the development of a lexical knowledge base for natural language processing. *SIGIR Forum*, 23:1–2, 1989.
4. Sören Auer, Sebastian Dietzold, and Thomas Riechert. Ontowiki - a tool for social, semantic collaboration. In *ISWC 2006*, volume 4273 of *LNCS*, pages 736–749. Springer, 2006.
5. Sergey Brin. Extracting patterns and relations from the world wide web. In *WebDB*, pages 172–183, London, UK, 1999. Springer-Verlag.
6. Sam Coates-Stephens. The analysis and acquisition of proper names for the understanding of free text. *Computers and the Humanities*, 26:441–456, 1992. 10.1007/BF00136985.
7. James R. Curran and Stephen Clark. Language independent ner using a maximum entropy tagger. In *HLT-NAACL*, pages 164–167, 2003.
8. Thomas G. Dietterich. Ensemble methods in machine learning. In *MCS*, pages 1–15, London, UK, 2000. Springer-Verlag.
9. Oren Etzioni, Michael Cafarella, Doug Downey, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S. Weld, and Alexander Yates. Unsupervised named-entity extraction from the web: an experimental study. *Artif. Intell.*, 165:91–134, June 2005.
10. J. Finkel, T. Grenager, and C. Manning. Incorporating non-local information into information extraction systems by gibbs sampling. In *ACL*, pages 363–370, 2005.
11. Eibe Frank, Gordon W. Paynter, Ian H. Witten, Carl Gutwin, and Craig G. Nevill-Manning. Domain-specific keyphrase extraction. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, IJCAI '99, pages 668–673, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.
12. R. Grishman and R. Yangarber. Nyu: Description of the Proteus/Pet system as used for MUC-7 ST. In *MUC-7*. Morgan Kaufmann, 1998.
13. Sanda Harabagiu, Cosmin Adrian Bejan, and Paul Morarescu. Shallow semantics for relation extraction. In *IJCAI*, pages 1061–1066, 2005.

14. David Huynh, Stefano Mazzocchi, and David R. Karger. Piggy bank: Experience the semantic web inside your web browser. In *ISWC*, volume 3729 of *Lecture Notes in Computer Science*, pages 413–430. Springer, 2005.
15. Su Nam Kim and Min-Yen Kan. Re-examining automatic keyphrase extraction approaches in scientific articles. In *MWE '09*, pages 9–16, 2009.
16. Su Nam Kim, Olena Medelyan, Min-Yen Kan, and Timothy Baldwin. Semeval-2010 task 5: Automatic keyphrase extraction from scientific articles. In *SemEval '10*, pages 21–26, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.
17. Y. Matsuo and M. Ishizuka. Keyword Extraction From A Single Document Using Word Co-Occurrence Statistical Information. *International Journal on Artificial Intelligence Tools*, 13(1):157–169, 2004.
18. D. Nadeau. *Semi-Supervised Named Entity Recognition: Learning to Recognize 100 Entity Types with Little Supervision*. PhD thesis, University of Ottawa, 2007.
19. David Nadeau, Peter Turney, and Stan Matwin. Unsupervised named-entity recognition: Generating gazetteers and resolving ambiguity. *Advances in Artificial Intelligence*, pages 266–277, 2006.
20. Dat P. T. Nguyen, Yutaka Matsuo, and Mitsuru Ishizuka. Relation extraction from wikipedia using subtree mining. In *AAAI*, pages 1414–1420, 2007.
21. Thuy Nguyen and Min-Yen Kan. Keyphrase Extraction in Scientific Publications. *Asian Digital Libraries*, pages 317–326, 2007.
22. Patrick Pantel and Marco Pennacchiotti. Espresso: Leveraging generic patterns for automatically harvesting semantic relations. In *ACL*, pages 113–120, 2006.
23. Youngja Park, Roy J Byrd, and Branimir K Boguraev. Automatic glossary extraction: beyond terminology identification. In *COLING*, pages 1–7, 2002.
24. Marius Pasca, Dekang Lin, Jeffrey Bigham, Andrei Lifchits, and Alpa Jain. Organizing and searching the world wide web of facts - step one: the one-million fact extraction challenge. In *proceedings of the 21st national conference on Artificial intelligence - Volume 2*, pages 1400–1405. AAAI Press, 2006.
25. Lev Ratinov and Dan Roth. Design challenges and misconceptions in named entity recognition. In *CONLL*, pages 147–155, 2009.
26. Christine Thielen. An approach to proper name tagging for german. In *In Proceedings of the EACL-95 SIGDAT Workshop*, 1995.
27. Sebastian Tramp, Norman Heino, Sören Auer, and Philipp Frischmuth. RDFauthor: Employing RDFa for Collaborative Knowledge Engineering. In P. Cimiano and H.S. Pinto, editors, *Knowledge Engineering and Management by the Masses; 17th International Conference, EKAW 2010, Lisbon, Portugal*, October 2010.
28. Peter D. Turney. Coherent keyphrase extraction via web mining. In *IJCAI*, pages 434–439, San Francisco, CA, USA, 2003.
29. D. Walker and R. Amsler. The use of machine-readable dictionaries in sublanguage analysis. *Analysing Language in Restricted Domains*, 1986.
30. Gang Wang, Yong Yu, and Haiping Zhu. Pore: Positive-only relation extraction from wikipedia text. In *ISWC07*, volume 4825 of *LNCS*, pages 575–588, Berlin, Heidelberg, 2007. Springer Verlag.
31. Yulan Yan, Naoaki Okazaki, Yutaka Matsuo, Zhenglu Yang, and Mitsuru Ishizuka. Unsupervised relation extraction by mining wikipedia texts using information from the web. In *ACL*, ACL '09, pages 1021–1029, 2009.
32. GuoDong Zhou and Jian Su. Named entity recognition using an hmm-based chunk tagger. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 473–480, Morristown, NJ, USA, 2002. Association for Computational Linguistics.