Case Acquisition from Text: Ontology-based Information Extraction with SCOOBIE for myCBR

Thomas Roth-Berghofer[1,2], Benjamin Adrian[1,2], and Andreas Dengel[1,2]

[1] Knowledge Management Department,
German Research Center for Artificial Intelligence (DFKI) GmbH
Trippstadter Straße 122, 67663 Kaiserslautern, Germany

[2] Knowledge-Based Systems Group, Department of Computer Science,
University of Kaiserslautern, P.O. Box 3049, 67653 Kaiserslautern

{firstname.lastname}@dfki.de

Abstract. *myCBR* is a freely available tool for rapid prototyping of similarity-based retrieval applications such as case-based product recommender systems. It provides easy-to-use model generation, data import, similarity modelling, explanation, and testing functionality together with comfortable graphical user interfaces. SCOOBIE is an ontology-based information extraction system, which uses symbolic background knowledge for extracting information from text. Extraction results depend on existing knowledge fragments. In this paper we show how to use SCOOBIE for generating cases from texts. More concrete we use ontologies of the Web of Data, published as so called Linked Data interlinked with *myCBR*'s case model. We present a way of formalising a case model as Linked Data ready ontology and connect it with other ontologies of the Web of Data in order to get richer cases.

Key words: Textual Case-Based Reasoning, Ontology-based Information Extraction, Linked Open Data, Web of Data

1 Introduction

Structural Case-Based Reasoning relies on cases described by attributes and corresponding values. Structural CBR systems organise attributes in various ways, e.g., as flat attribute lists or in an object-oriented way. This organisation is often called domain ontology. The structural approach is useful in domains where additional knowledge such as complex similarity measures must be used in order to produce good results, and where a case model is easy to acquire.

myCBR[3] [22] is an Open Source (Structural) CBR tool developed at the German Research Center for Artificial Intelligence (DFKI). Key motivation for implementing *myCBR* was the need for a compact and easy-to-use tool for rapidly

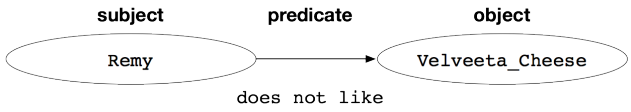
[3] http://www.mycbr-project.net

building prototype CBR applications in teaching, research, and small industrial projects with low effort. Moreover, the tool is easily extendable in order to facilitate the experimental evaluation of novel algorithms and research results.

In domains where large collections of semi-structured documents are readily available a different CBR approach is used: Textual CBR (TCBR). TCBR systems aim at managing information contained in semi-structured documents and providing means for content-oriented retrieval. The approach is especially useful in domains where the intended user is able to immediately make use of the knowledge contained in the respective documents.

There is no standard or consensus about the structure of a textual CBR system. One way of employing TCBR is to determine the structure of cases by a case model and use a pre-processor to fill case templates from semi-structured texts, thus providing a structured representation of the documents. The quality of the cases obviously depends on the abilities of the pre-processor.

In this paper, we describe how to combine the ontology-based information extraction tool SCOOBIE [3] as such a pre-processor with *myCBR*, where an ontology is an explicit, formal specification of a conceptualisation, i.e. a particular vocabulary that can be employed for describing aspects of real domains [16]. SCOOBIE strongly depends on the existence of an ontology to provide formal extraction results. In the presented approach, one such ontology is the case model of *myCBR*. As standard ontology representation formalism we chose to express the case model in the Resource Description Framework language (RDF) [23] where meaning is expressed by facts encoded in sets of triples [7].




```
<rdf:Description
  rdf:about="http://dbtropes.org/resource/Main/Ratatouille#Remy">
  <does-not-like
    rdf:resource=
      "http://mycbr-project.net/models/Recipe#velveeta_cheese"/>
</rdf:Description>
```

Fig. 1. RDF represents information as graph in form of triples. Triples consist of subject, predicate and object values. RDF can be serialised in XML.

Triples are like elementary sentences composed of subject, predicate, and object. Subjects, predicates, and objects are given as names for entities, also called resources or nodes. Entities represent something like a contact, an appointment, a website, etc. Names are either literals or Uniform Resource Identifiers (URI) [8], which are global in scope, always referring to the same entity in any RDF document in which they appear. Fig. 1 shows an example of an RDF triple describing which cheese Remy does not like.

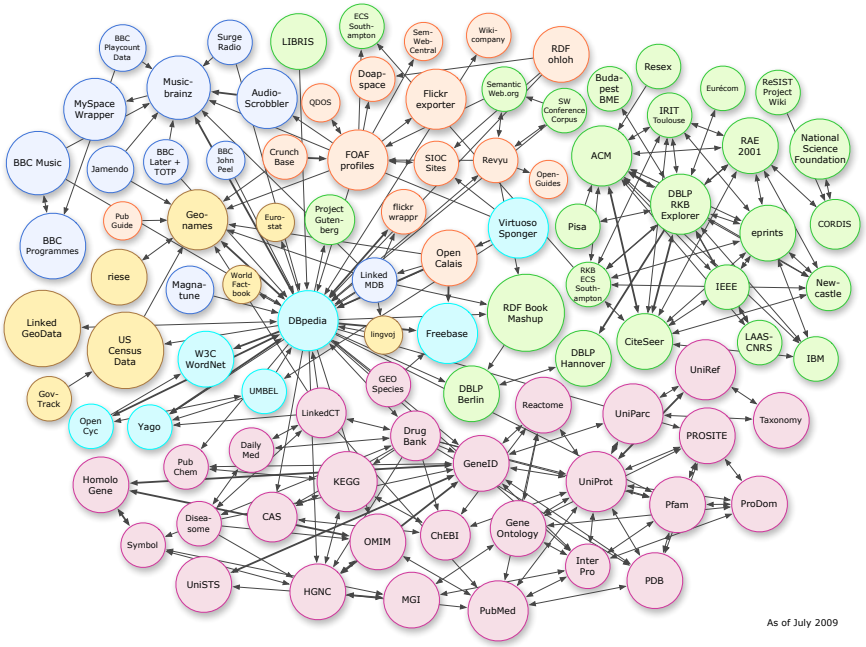


Fig. 2. Existing linked open data sets (as of July 2009) [9]

Representing a case model in RDF results in URIs for each attribute and possible values. If an HTTP request of these URIs delivers RDF triples with additional information about the resource identified with this URI and if several URIs in multiple web domains are interlinked with special types of predicates (e.g., `owl:sameAs`) whether they represent a similar resource, we talk about Linked (Open) Data (Fig. 2)[4] or the Web of Data [9].

We show that case models published as Linked Open Data can be interconnected with information from various data sources. Thus, they can be enriched with additional knowledge from other ontologies. Combining SCOOBIE and *myCBR* allows leveraging available knowledge of the Web of Data for acquiring cases from texts.

The rest of the paper is structured as follows: The next section provides an overview of previous and related work regarding ontology-based information extraction and Textual CBR. Section 3 briefly recapitulates relevant features of *myCBR*. In Sect. 4 we describe the ontology-based information extraction tool SCOOBIE before we put SCOOBIE to use for *myCBR* (Sect. 5). Finally, Sect. 6 summarises the combination of the two tools and concludes the paper with an outlook on further work.

[4] Graphic source URL of the LOD cloud: `http://www4.wiwiss.fu-berlin.de/bizer/pub/lod-datasets_2009-07-14_colored.png`

2 Related Work

In their survey article on Textual CBR, Weber et al. [24] formulate four research questions that TCBR addresses. TCBR looks at how to *assess similarity between textually represented cases*, *map from texts to structured case representations*, *adapt textual cases*, and *automatically generate representations for TCBR*.

In this classification schema our approach clearly addresses the second question, of how to map texts to structured case representations and work along the line of Lenz's framework, which targets semi-structured documents [18]. In this our approach is similar to what empolis' information access suite [14] and jColibri[5] [6, 19] are offering: intelligent access to all sorts of documents.

In [17], Lenz describes a layered model that divides text processing into several stages, i.e., Keyword Layer, Phrase Layer, Thesaurus Layer, Glossary Layer, Feature Value Layer, Domain Structure Layer, and Information Extraction Layer. SCOOBIE is built upon a standard information extraction pipeline architecture [2]. In general, Lenz' layered model of textual CBR subsumes SCOOBIE's pipeline architecture, moreover SCOOBIE can be incorporated into parts of the layer model, i.e., Keyword Layer, Phrase Layer, Thesaurus Layer, and the Information Extraction Layer.

SCOOBIE is based on information extraction (IE) principles that are well described in [5]. Comparable and approved OBIE systems are the General Architecture for Text Engineering (GATE)[6] [10] and the SmartWeb Ontology-Based Annotation component (SOBA) [12]. In contrast to these, SCOOBIE does not use the ontology as input gazetteer, i.e., a plain list of relevant labels, but as a model for semantic analyses such as instance disambiguation and discourse analysis. This technique of using existing domain ontologies as input for information extraction tasks and extraction results for ontology population and therefore knowledge acquisition was presented already in [20].

SCOOBIE also supports the population of extraction templates. The concept of querying ontologies with IE templates in order to extract information from text is completely missing in existing OBIE systems. Other OBIE approaches can be found in the proceedings of the OBIES workshop 2008 [4].

3 Similarity-based Retrieval with myCBR

myCBR, in its current version[7], focuses on the similarity-based retrieval step of the CBR cycle [1], as this is still the core functionality of most CBR applications. A popular example of such retrieval-only systems are case-based product recommender systems [11]. While the first CBR systems were often based on simple distance metrics, today many CBR applications make use of highly

[5] `http://gaia.fdi.ucm.es/projects/jcolibri`
[6] `http://gate.ac.uk`
[7] *myCBR*, Version 2.6

sophisticated, knowledge-intensive similarity measures [21]. Such extremely domain specific similarity measures enable the improvement of the retrieval quality substantially. However, they increase the development effort significantly.

myCBR is a plug-in for the Java-based Open Source ontology editor Protégé[8] [15], which was chosen as the modelling platform for *myCBR* and which existing functionality for creating case models and instances was extended for modelling similarity measures and testing similarity-based retrieval.

The major goal of *myCBR* was to minimise the effort for building CBR applications that require knowledge-intensive similarity measures. Therefore, it provides comfortable graphical user interfaces for modelling various kinds of attribute-specific similarity measures and for evaluating the resulting retrieval quality. In order to reduce also the effort of the preceding step of defining an appropriate case representation, it includes tools for generating the case representation automatically from existing raw data.

myCBR is intended for structural CBR applications that make use of rich attribute-value based or object-oriented case representations. Although Protégé provides powerful graphical user interfaces for modelling attribute-value based and object-oriented representations, their manual definition remains a laborous task. It includes the definition of classes and attributes and the specification of accurate value ranges required for a meaningful similarity assessment.

3.1 CSV Import

In order to ease the definition of case representations, *myCBR* provides a powerful CSV[9] data import module. CSV files are widely used to store attribute-value based raw data in pure ASCII format. For example, in the Machine Learning community example data sets are usually exchanged by using CSV files[10]. Using the CSV importer, the user has the choice to import data instances into an existing Protégé data model, or to create a new model automatically based on the raw data. In the latter case, *myCBR* generates a Protégé slot[11] for each data column of the CSV file automatically. After the CSV data has been imported, the user may further modify the generated case model (e.g., extend it to an object-oriented representation) to meet the application specific needs. The final case model together with the case base is stored by *myCBR* in XML files.

3.2 Cases from Texts

For retrieving texts with *myCBR* a similar approach to CSV import is used. But where the entries in the CSV file are used at face value without interpreting

[8] http://protege.stanford.edu/
[9] Comma Separated Values
[10] See, for example, `http://archive.ics.uci.edu/ml/`
[11] In Protégé, attributes are called *slots*.

them[12], texts as raw material for cases need to be searched for the occurrence of certain concepts and phrases, i.e., attribute values.

Concepts may be expressed in different ways. They may vary grammatically (singular, plural, case) or semantically (synonyms, hyponyms, hypernyms). The different representations can be seen as triggers for concepts. Concepts can also be measured values, i.e., numbers with a unit.

Another difference to the CSV import is that a case model cannot be derived automatically from texts. At least a basic case model is required to start building cases from texts. Section 5 describes the process in detail. First we will have a closer look at SCOOBIE.

4 Ontology-based Information Extraction with SCOOBIE

Ontology-based information extraction (OBIE) extracts formal facts from text resources. In terms of RDF, facts may be triples representing attribute knowledge about a resource (e.g., *:Cheddar_ Cheese skos:prefLabel "Cheddar Cheese"*)[13] or relations between resources (e.g., *:Remy :does-not-like :velveeta_ cheese*). OBIE algorithms incorporate those relevant bits of knowledge from an input ontology that support information extraction inside a pipeline of cascading extraction tasks [2]. Conceiving the extraction pipeline of the SCOOBIE system [3] as black box, mandatory input parameters are:

1. The *ontology*, which comprises vocabularies and schemes, used to represent entities of the SCOOBIE domain model. The classes (e.g., ingredients, or cooking steps), datatype properties of these classes (e.g., ingredient label or recipe name) and object properties between instances of these classes (e.g., persons not liking ingredients) define a search space of possible instances and facts that may be extracted from text.
2. *Entities* of SCOOBIE's domain model, which are represented as instances of the ontology. The given datatype property values of these instances (e.g., the symbolic slot value of a recipe's ingredient called *"cheese"*) are used for extracting instances from text. Object properties between instances are used for disambiguating instances with similar datatype property values or ranking the relevance of extracted instances.

As shown in Fig. 3, the ontology and its instances are analysed during an offline pre-processing and training phase. Results are index structures (e.g., suffix arrays, B*-trees) and learning models (e.g., Conditional Random Fields, K-Nearest Neighbour Classifiers) that can now be used by efficient extraction tasks inside the extraction pipeline:

[12] The importer analyses each column to determine the value ranges for slots. For text data, symbol or string type slots are created depending on a user defined threshold.

[13] In order to preserve readable inline examples we prefered writing RDF triples in Turtle syntax (please refer to `http://www.w3.org/2007/02/turtle/primer/`).

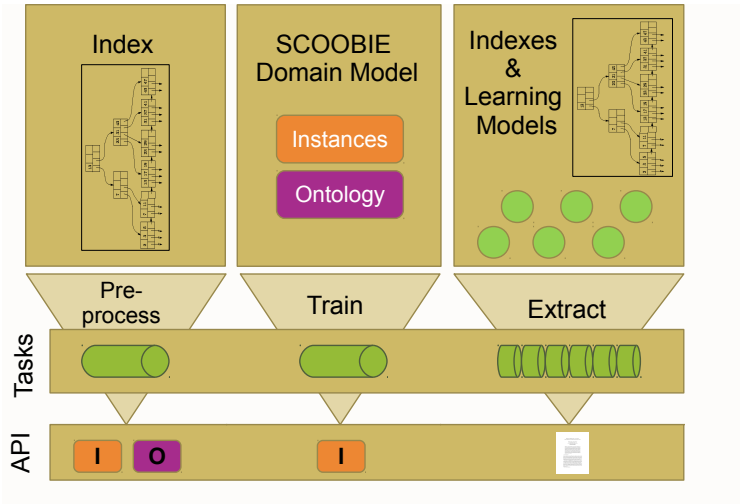


Fig. 3. Architecture of the ontology-based information extraction system SCOOBIE

1. *Normalisation*: Extracts document metadata and plain text data from textual or binary file formats. The language of the plain text is detected by applying statistics about n-gram distributions of letters in different languages.
2. *Segmentation*: Partitions plain text into segments: paragraphs, sentences, and tokens. With respect to the detected language, a POS tagger tags each token with its part of speech (POS).
3. *Symbolisation*: Recognises datatype property values in text by matching phrases in text and values of datatype properties of the domain model.
   For example, assume having the triple *:Cheddar_ Cheese skos:prefLabel "Cheddar Cheese"*, then *"Cheddar Cheese"* may be recognised as content symbol.
   By performing noun phrase chunking, noun phrases expressing candidates for names without any structure in syntax (e.g., names) are detected.
4. *Instantiation*: For each recognised datatype property value the Instantiation resolves instance candidates.
   For example, assume having the triple *:Cheddar_ Cheese skos:prefLabel "Cheddar Cheese"*, then *"Cheddar Cheese"* is resolved as *skos:prefLabel* of instance *:Cheddar_ Cheese*. An instance candidate recognition resolves possible candidates for recognised datatype property values. Here, ambiguities may occur if more than one instance possesses the same datatype property values (e.g., determining wether "onions" symbolise "red onions" or "green onions").
   Candidates are disambiguated by counting resolved instances in the domain model that are related directly with an object property or indirectly via another instance of the domain model. As result, the ambiguous instance with a higher count of related and recognised instances is taken.
5. *Contextualisation*: Extracts facts (RDF triples) about resolved instances. At first, a fact candidate extraction computes all possible facts between

resolved instances. Then, a set of fact selectors rates these facts according to heuristics. A known fact selector heightens rates of extracted facts that exist as triples inside the domain model.

6. *Population*: Creates scenario graphs in RDF format. They contain extracted values, i.e., HTTP URIs of resolved instances with those datatype property values that match with text sequences and RDF triples about object properties between these resolved instances. Scenario graphs can be filtered and ordered by confidence values in range between zero and one.

5 Generating Structural Cases from Texts

Using SCOOBIE as pre-processor for *myCBR* comprises four steps as illustrated by Fig. 4: Based on the *case model* the *information extraction model* is generated (1). This is then fed into SCOOBIE *for* myCBR together with *linked open data* and the text files from which the cases are to be extracted (2). The extracted concepts (expressed in RDF format) are then combined into cases (3), which can eventually be used for similarity-based retrieval (4). We will detail and exemplify each step in the following by providing our fictive user Remy with a similarity-based search engine for recipes.

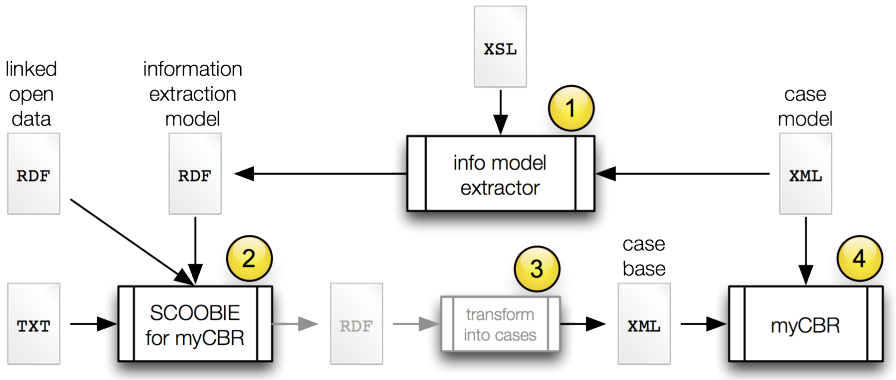


Fig. 4. Process overview: generate information extraction model from case model (1), extract case information from texts (2), transform result into cases (3), and use generated cases for retrieval (4)

5.1 Generating the Information Extraction Model

Cases in *myCBR* are class instances modelled with Protégé. In this first version only flat attribute-value lists are considered, but the approach can be easily

extended to more complex class structures by adapting the application logic in SCOOBIE for *myCBR* accordingly.

Our guiding example is recipe search. Remy likes to cook but sometimes forgets to buy ingredients and needs to deal with what is left in the fridge and in his storage room. Remy needs a system for retrieving recipes that he can cook with the ingredients at hand. As recipe collection we used recipes of the Second Computer Cooking Contest[14].

For the similarity-based search we modelled a class *Recipe* with such attributes as *title, ingredients_ meat, ingredients_ vegetables, ingredients_ fish, ingredients_ pasta, ingredients_ cheese, ingredients_ spices, preparation_ steps*. The title is not used for retrieval, but for identifying the recipe. All other attributes are of type *Symbol* and can hold multiple values. For example, allowed values for *ingredients_ meat* are *chicken, pork,* and *bacon* which can occur together in one recipe. *myCBR* provides various options to configure similarity measures for set type attributes [22]. Depending on the chosen settings, the mapping between query values and case values is calculated differently. For example, a set of values may have an "and" or an "or" semantic. The size of the query and case sets also may have different impact on the similarity.

The *information model extractor* (circle (1) in Fig. 4) takes the case model and transforms it into an information extraction model with the help of XSL transformations[15]. The Extensible Stylesheet Language (XSL)[16] is used for describing how to transform and render XML documents.

```
<SMFunction smfname="default" model_instname="ingredients_cheese"
    type="Symbol" active="true" simMode="Table">
  <QuerySymbol symbol="blue_cheese">
    <CBSymbol sim="1.0" symbol="blue_cheese" />
  </QuerySymbol>
  <QuerySymbol symbol="velveeta_cheese">
    <CBSymbol sim="1.0" symbol="velveeta_cheese" />
  </QuerySymbol>[...]
</SMFunction>
```

Fig. 5. Section of *myCBR* case model

Figure 5 shows a few lines of the case model.[17] Two of the allowed values for the attribute *ingredients_ cheese* are shown: *blue_ cheese* and *velveeta_ cheese*.

SCOOBIE needs its input ontologies represented in RDF format. For transforming the case model into RDF, using XSL was an obvious choice. We use SKOS, the Simple Knowledge Organisation Systems family of formal languages

[14] CCC at ICCBR 2009: `http://www.wi2.uni-trier.de/ccc09/index.php`

[15] `http://www.w3.org/TR/xslt`

[16] `http://www.w3.org/Style/XSL/`

[17] Actually, the XML snippet is part of the similarity measure functions file, which duplicates the information contained in the proprietary, LISP/Protégé file format.

```
<xsl:for-each select="QuerySymbol">
  &lt;skos:Concept
      rdf:about="<xsl:value-of
        select='concat("http://mycbr-project.net/models/Recipe#",
          @symbol)'/>"&gt;
    &lt;skos:prefLabel&gt;
      <xsl:value-of select='@symbol'/>
    &lt;/skos:prefLabel&gt;
    &lt;rdf:type
      rdf:resource=
        &quot;<xsl:copy-of select="$slot_name_for_type"/>&quot;/&gt;
  &lt;/skos:Concept&gt;
</xsl:for-each>
```

Fig. 6. Section of XSL transformations

as representation formalism. SKOS is designed for exactly our purpose of representing a structured controlled vocabulary.[18] SKOS is built upon RDF. Its main objective is to enable easy publication of controlled structured vocabularies for the Semantic Web.

Figure 6 shows the main section of the XSL file. The XSL transformations are nearly domain independent. Following the Linked Open Data principles with HTTP accessable URIs the RDF ontology file needs to exist at the given URL (e.g., `http://mycbr-project.net/models/Recipe`) and needs to be provided in the XSL stylesheet. And there is the basic requirement that just one class exists (which, in turn, implies that the case model is flat). Then, for each *QuerySymbol*, i.e., allowed value, one SKOS concept is created.

```
<skos:Concept
    rdf:about="http://mycbr-project.net/models/Recipe#blue_cheese">
  <skos:prefLabel>blue cheese</skos:prefLabel>
  <rdf:type rdf:resource="ingredients_cheese"/>
</skos:Concept>
```

Fig. 7. Section of transformation result: initial information extraction model

Each attribute value (of type Symbol) becomes a SKOS concept. In order to keep the relation between attribute and value the attribute name is treated as a SKOS concept's RDF type. The allowed value becomes the preferred SKOS label (and is used for the information extraction task in the next step). Figure 7 shows the result of the transformation process. The concept with URI `http://mycbr-project.net/models/Recipe#blue\_cheese` is of RDF type *ingredients_ cheese* and has the preferred label *blue cheese*.

[18] `http://www.w3.org/TR/2009/REC-skos-reference-20090818/`

5.2 Extract Case Information from Texts

The resulting information extraction model is the core ontology. It is used to extract concepts from input texts, but it would perform very poorly if used alone. SCOOBIE would only be able to find exact matches to preferred labels. In order to find more concepts additional ontologies are needed.

```
<http://mycbr-project.net/models/Recipe#red_pepper">
  owl:sameas <http://dbpedia.org/resource/Cayenne_pepper>
<http://mycbr-project.net/models/Recipe#blue_cheese">
  owl:sameas <http://dbpedia.org/resource/Blue_cheese>
```

Fig. 8. Example expressions connecting the case model to DBpedia

Further intelligence is brought into the system by, first, extending the number of labels (alternative labels), which act as triggers for the associated symbols and, second, by linking SKOS concepts, and thus symbols, to Linked Open Data (e.g., DBpedia[19]). This last step then enhances the case generation step drastically. Figure 8 shows a sample snippet of the respective expressions.

SCOOBIE then extracts attribute values according to the information extraction model and all additional ontologies linked to the information extraction model. A configuration model determines which portion of the texts are analysed into which attribute.

```
<RECIPE>
  <TI>"Blue" Fettuccine</TI>
  <IN>4 oz Danish blue cheese or 8 oz. Danish blue
    Castello cheese, chilled</IN>
  <IN>1/4 c Marinated, dried tomatoes</IN>
  <IN>8 oz Green fettuccine or spinach egg noodles</IN> [...]
  <IN>1/4 c Chopped fresh parsley</IN>
  <PR>[...]
    <STEP>Meanwhile, in small skillet over medium heat, heat
    reserved oil; add shallots and garlic. Saute until shallots
    are limp but not brown.</STEP>
    <STEP>Add wine, basil and reserved tomatoes. Heat through
    and keep hot.</STEP> [...]
    </PR>
</RECIPE>
```

Fig. 9. Example recipe text

[19] "DBpedia is a community effort to extract structured information from Wikipedia and to make this information available on the Web." `http://dbpedia.org/About`

The recipes of the Computer Cooking Contest are given in XML format. Figure 9 shows a section of a recipe. Each recipe is divided in title <TI>...</TI>, ingredients <IN>...</IN> and preparation steps <STEP>...</STEP>. The title is just copied into the attribute *title* and used as an identifier for a case. The ingredients are analysed into the attributes *title, ingredients_meat, ingredients_vegetables, ingredients_fish, ingredients_pasta, ingredients_cheese, ingredients_spices*. The preparation steps go into the attribute *preparation_steps*. All attributes are multi-valued, as already said above. The order of steps cannot be taken into account. This is due to the set semantics for multi-valued attributes.

5.3 Building the Case Base

The resulting RDF extracts are transformed into *myCBR*'s case base format (illustrated in Fig. 10). An instance is created, and the title is copied from the recipe text into the respective slot.

```
<Instance model_instname="recipes_Class10000">
  <slotvalue slot="title" value="&quot;blue&quot; fettuccine" />
  <slotvalue slot="ingredients_cheese" value="blue_cheese" />
  <slotvalue slot="ingredients_pasta" value="egg_noodles" />
  <slotvalue slot="ingredients_pasta" value="noodles" />
  <slotvalue slot="ingredients_pasta" value="spinach_noodles" />
  <slotvalue slot="preparation_steps" value="heat" />
  <slotvalue slot="preparation_steps" value="saute" /> [...]
</Instance>
```

Fig. 10. Snippet of an example (recipe) case

From each extracted SKOS concept the slot name, encoded in each of the concept's RDF type, is taken (e.g., *ingredients_cheese* in Fig. 7) and filled with the respective concept's name (e.g., *blue_cheese* from the concept's URI *http://mycbr-project.net/models/Recipe#blue_cheese*).

Finally, the newly generated case base can be used for searching recipes that match a given set of available ingredients and preferred way of preparation. Using the title of the case the original recipe text can be retrieved for Remy.

6 Conclusion and Outlook

Weber et al. [13] describe the combination of knowledge elicitation for a case model and information extraction techniques for case generation in a knowledge management scenario. SCOOBIE and *myCBR* very well fit into this scenario.

In this paper we described how to combine the ontology-based information extraction tool SCOOBIE with the similarity-based, structural CBR system *myCBR*. Ontology-based information extraction systems strongly depend on the existence of symbolic background knowledge for generating relevant results. Such

knowledge is available in *myCBR* and can be used as seeding knowledge for feeding the powerful information extraction system SCOOBIE. Linking this core knowledge to available linked open data on the Web of Data provides new opportunities for knowledge modelling and retrieval.

A case model usually is constructed top down after analysing a domain and the respective documents. From the case model we automatically generated an information extraction model consisting of SKOS concepts. For each allowed value of symbol type attributes we constructed a unique SKOS concept and used the allowed value as preferred label. This IE model already allows for simple text extraction. Linking the IE model to the Web of Data makes the difference.

As we have shown, SCOOBIE has a lot more to offer (e.g., extracting facts, or even new instances), but the current implementation only uses a fraction of SCOOBIE's abilities. A GUI, based on the OSGi[20] platform, is under development to ease the use of developing the case model in combination with case generation from texts. This close interaction will, on one hand, help to extend the case model with symbols previously not recognised (i.e., extend the range of allowed values), and, on the other hand, help to extend the information extraction model by linking allowed values to concepts of other ontologies on the Web of Data.

myCBR is an ongoing project. We encourage others to try out *myCBR* in their own research and teaching projects and to contribute to the further development by implementing their own extensions and experimental modules.

Acknowledgments

This work was funded by the BMBF project Perspecting (Grant 01IW08002).

References

1. Aamodt, A.: Explanation-driven case-based reasoning. In: Stefan Wess, K.D.A., Richter, M. (eds.) Topics in Case-Based Reasoning. Springer-Verlag, Berlin (1994)
2. Adrian, B., Dengel, A.: Believing finite-state cascades in knowledge-based information extraction. In: Dengel, A., Berns, K., Breuel, T., Bomarius, F., Roth-Berghofer, T.R. (eds.) KI. LNAI, vol. 5243, pp. 152–159. Springer, Berlin
3. Adrian, B., Hees, J., van Elst, L., Dengel, A.: iDocument: Using ontologies for extracting and annotating information from unstructured text. In: Mertsching, B., Hund, M., Aziz, Z. (eds.) KI 2009: Advances in Artificial Intelligence. Künstliche Intelligenz (KI-2009), September 15-18, Paderborn, Germany. Lecture Notes in Artificial Intelligence, LNAI, vol. 5803, pp. 249–256. Springer-Verlag, Heidelberg
4. Adrian, B., Neumann, G., Troussov, A., Popov, B. (eds.): Ontology-based Information Extraction Systems (OBIES 2008) (2008), `http://CEUR-WS.org/Vol-400/`
5. Appelt, D., Israel, D.: Introduction to information extraction technology: A tutorial prepared for ijcai-99. SRI International (1999)

[20] `http://en.wikipedia.org/wiki/OSGi`: [Last access: 2010-05-03]

6. Bello-Tomás, J., González-Calero, P.A., Díaz-Agudo, B.: JColibri: An Object-Oriented Framework for Building CBR Systems. In: Calero, P.A.G., Funk, P. (eds.) Proceedings of the 7th European Conference on Case-Based Reasoning. Lecture Notes in Artificial Intelligence LNAI, Springer (2004)
7. Bergmann, R., Schaaf, M.: Structural Case-Based Reasoning and Ontology-Based Knowledge Management: A Perfect Match? Journal of Universal Computer Science 9(7), 608–626 (2003), http://www.jucs.org/jucs_9_7/structural_case_based_reasoning [Last access: 2010-02-26]
8. Berners-Lee, T., Fielding, R.T., Masinter, L.: Rfc 3986: Uniform resource identifier (uri): Generic syntax (2005), http://www.ietf.org/rfc/rfc3986.txt [Last access: 2010-02-26]
9. Bizer, C., Heath, T., Berners-Lee, T.: Linked data - the story so far. International Journal on Semantic Web and Information Systems 5(3), 1–22 (2009), http://dblp.uni-trier.de/db/journals/ijswis/ijswis5.html#BizerHB09 [Last access: 2010-02-26]
10. Bontcheva, K., Tablan, V., Maynard, D., Cunningham, H.: Evolving GATE to meet new challenges in language engineering. Natural Language Engineering 10(3-4), 349–373 (2004), http://journals.cambridge.org/action/displayAbstract?fromPage=online&aid=252241&fulltextType=RA&fileId=S1351324904003468 [Last access: 2010-02-26]
11. Bridge, D., Göker, M.H., McGinty, L., Smyth, B.: Case-based recommender systems. Knowledge Engineering Review 20(3) (2006)
12. Buitelaar, P., Cimiano, P., Racioppa, S., Siegel, M.: Ontology-based Information Extraction with SOBA. In: Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC). pp. 2321–2324. ELRA (2006), http://www.aifb.uni-karlsruhe.de/WBS/pci/Publications/buitelaaretal_lrec06.pdf [Last access: 2010-02-26]
13. David, R.W., Aha, D.W., Sandhu, N., Munoz-Avila, H.: A textual case-based reasoning framework for knowledge management applications. In: Schnurr, H.P., Staab, S., Studer, R., Stumme, G., Sure, Y. (eds.) In Proceedings of the Ninth German Workshop on Case-Based Reasoning. pp. 244–253. Shaker Verlag, Aachen (2001)
14. empolis: Information Access Suite — e:IAS Text Mining Engine. Technical white paper (2008), empolis IAS 6.2, Publication Date: 8 Dec. 2008, Build: 5976
15. Gennari, J.H., Musen, M.A., Fergerson, R.W., Grosso, W.E., Crubézy, M., Eriksson, H., Noy, N.F., Tu, S.W.: The evolution of Protégé an environment for knowledge-based systems development. Int. J. Hum.-Comput. Stud. 58(1), 89–123 (2003)
16. Gruber, T.R.: Toward principles of the design of ontologies used for knowledge sharing. International Journal of Human and Computer Studies 43, 907–928 (1995)
17. Lenz, M.: Defining knowledge layers for textual case-based reasoning. In: Smyth, B., Cunningham, P. (eds.) EWCBR. Lecture Notes in Computer Science, vol. 1488, pp. 298–309. Springer (1998)
18. Lenz, M.: Knowledge sources for textual cbr applications. In: Lenz, M., Ashley, K. (eds.) Textual Case-Based Reasoning: Papers from the AAAI-98 Workshop. pp. 24–29. AAAI Press, Menlo Park, CA (1998), technical Report WS-98-12
19. Recio-García, J.A., Díaz-Agudo, B., Gómez-Martín, M.A., Wiratunga, N.: Extending jcolibri for textual cbr. In: Muñoz-Avila, H., Ricci, F. (eds.) ICCBR. Lecture Notes in Computer Science, vol. 3620, pp. 421–435. Springer (2005), http://dblp.uni-trier.de/db/conf/iccbr/iccbr2005.html#RecioDGW05

20. Sintek, M., Junker, M., Elst, L.V., Abecker, A.: Using information extraction rules for extending domain ontologies. In: Maedche, A., Staab, S., Nedellec, C., Hovy, E. (eds.) Position Statement for the IJCAI-2001 Workshop on Ontology Learning (2001), http://CEUR-WS.org/Vol-38/
21. Stahl, A.: Learning of Knowledge-Intensive Similarity Measures in Case-Based Reasoning. Ph.D. thesis, University of Kaiserslautern (2003)
22. Stahl, A., Roth-Berghofer, T.R.: Rapid prototyping of CBR applications with the open source tool myCBR. In: Bergmann, R., Althoff, K.D. (eds.) Advances in Case-Based Reasoning. Springer Verlag (2008)
23. W3C: Rdf primer (February 2004), http://www.w3.org/TR/2004/REC-rdf-primer-20040210/ [Last access: 2010-02-26]
24. Weber, R.O., Ashley, K.D., Brüninghaus, S.: Textual case-based reasoning. Knowledge Engineering Review 20(3), 255–260 (2005)