Iran J Sci Technol Trans Electr Eng (2016) 40:103–115
DOI 10.1007/s40998-016-0011-3

RESEARCH PAPER

# Semi-Automatic Rule Learning Method Enabling Information Extraction for Ontology Population

Girish R. Ranganathan[1] · Yevgen Biletskiy[2] · Ismail Akbari[1]

Received: 6 October 2015 / Accepted: 9 August 2016 / Published online: 7 November 2016

**Abstract** Business firms around the world have been generating enormous amounts of domain-related documents. Most of these firms are adapting semantic Web-based techniques into their software systems. Hence, they want to semantically enrich their documents to enable more meaningful querying or processing of the information in the documents. To impart semantics into these documents, ontologies relevant to the business domain should be used. In this context, to populate the domain ontology with the information from the source documents, a method for semi-automatic learning of extraction rules for populating the ontology is presented and implemented in the rule learning system. In addition to the rule learning system, a framework for separating the business logic from application logic and storing the business rules and extraction rules in external user-friendly format is presented in brief. The rule learning system is mainly developed to be a part of the presented framework, but it can be used as a standalone system to learn any decision or association rules too. The framework uses the rule learning system for learning extraction rules. The main idea behind the work presented is to learn extraction rules to be used by an information extraction system (part of the framework) to populate the domain ontology. The extraction rules learned by the rule learning system can be used with any business rules management system (BRMS) with appropriate wrappers to populate the domain ontology.

**Keywords** Business applications · Business rules · Extraction rules · Rule learning · Ontology · Semantic Web

✉ Ismail Akbari
iakbari@unb.ca

Girish R. Ranganathan
girish.ranganathan@unb.ca

Yevgen Biletskiy
biletski@unb.ca

1 Faculty of Computer Science, University of New Brunswick, PO Box 4400, C314, 550 Windsor Street, Fredericton, NB E3B 5A3, Canada

2 Department of Electrical and Computer Engineering, University of New Brunswick, PO Box 4400, D36, 15 Dineen Drive, Fredericton, NB E3B 6B9, Canada

## 1 Introduction

The work presented in this paper incorporates features of the semantic Web (Alpaydin 2004; Apache POI HSSF 2002) by using ontologies to impart semantics to the business documents. Around the world, most of the business documents with information about their consumers, working staff, procurement details, stock information, and other business data are mostly stored in organized documents such as tables. Hence, a very commonly used document format in this direction is Microsoft Excel, which is very user and business friendly. To impart semantics into the data in the business documents, ontologies relevant to the domain of business are used. These ontologies specific to a business domain are called domain ontologies (Bach et al. 2008; Ball et al. 2005). Some common enterprise ontologies are: enterprise ontology (Behkamal et al. 2012), ontology for enterprise modeling (Berners-Lee et al. 2001), TOVE Ontology Project for enterprise modeling (Biletskiy and Ranganathan 2008), and many others. The ontology can be represented in any ontology language like Web Ontology Language (OWL) (Boley 2004).

To enable easy creation of domain ontologies, the knowledge engineers can use ontology editors such as

Protégé (Buitelaar et al. 2008). The domain ontology can be built using ontology building methods. Among the common methodologies to build ontologies (Bach et al. 2008), the skeletal methodology proposed by Uschold and King (Ball et al. 2005) is very widely used. The created domain ontology should be populated with instances. To populate the ontology with information from source documents, there is a need for an information extractor. The information extractor used as a part of the presented framework (Celjuska and Vargas-Vera 2004) is the rule-based BRMS engine like OpenL tablets (Cimiano 2006). The extraction rules needed by the BRMS engine are learned by the rule learning system. The described approach for rule learning enables us to populate business domain ontologies semi-automatically and semantically to enrich business documents. The ontology instances reflect the content of semantically processed business documents. If desired, the populated domain ontology can be further compared (matched) with methods such as the ones introduced in Cohen (1995) to previous versions of the ontology (if available) or other similar ontologies in the domain for populating or finding inconsistencies between two ontologies.

To perform the necessary learning semi-automatically, the method described in this paper uses an annotation system which enables the knowledge engineer to create a training dataset. This training dataset is then used by the rule learning system to learn extraction rules required for populating the domain ontology. For populating the ontology with instances from the source documents, ontology capture methods (Bach et al. 2008; Ferrer-Troyano et al. 2005; Fox and Gruninger 1998), which form a part of the ontology building process, can be used. Ontology is normally captured from sources containing information about the domain of interest. There are two general methods of ontology capture: manual capture and, in general, information extraction-based capture. In the presented framework, the focus is only on populating the created domain ontology using an information extraction and document classification method.

Manual methods are not interesting in the context of this paper. Information extraction methods (Fox et al. 1996; Gomez-Perez et al. 2004) deal with semi-automatic or automatic information extraction of necessary information from structured, semi-structured, and unstructured documents. The extracted information is stored in an appropriate format which can be used for further processing. Some useful methods of information extraction related to the present work are described in Holmes et al. (1994). The work Holmes et al. (1994) presents a system engineered to perform Web-based information gathering, filtering, and monitoring using ontology and agents. The work Jena (2011) extracts relevant information from source documents based on a measure it proposes called query-sensitive similarity measure (QSSM).

The most popular information extraction methods are based on a smaller subset of information extraction called machine learning. Information extraction also includes rule extraction or rule induction, or rule learning, which is the focus of this paper. The work Language and org (2004) describes a semi-automatic system, so-called “Ontosophie”, for ontology population from unstructured text. The system is based on supervised learning of extraction rules from annotated text. The extracted rules are applied to new texts to populate the ontology. The “Ontosophie” system uses natural language processing (NLP) to recognize syntactic constructs. The system described in the present work focuses on the position-based information extrications, but can be empowered by NLP in future. The main focus of the present work is on the position-based rule learning using advantages of structure in the tabular formatted documents. Another challenge of the present work is that the information extraction rules are learned from human created documents rather than Web pages generated from a database in the background.

Rule learning (Li et al. 2007; Maedche and Staab 2001) is an area of information extraction in which formal rules are extracted from sets of observations. In the context of this paper, the set of observations is created by the annotation system. The rules extracted may represent a full scientific model of the data, or merely represent local patterns in the data. The rules extracted can be represented in any of these languages or others, not mentioned here, for further processing. Some useful methods of rule learning and ontology population have been described (Manine et al. 2008; Mierswa et al. 2006; Nederstigt et al. 2014a, b; Oberle et al. 2005; OpenL Tablets and net 2006; Protégé ontology editor and edu 2000). The paper by Manine et al. (2008) presents a system which performs information extraction for ontology population. In Mierswa et al. (2006), a model to learn transformation rules generated from a set of features for part of speech tagging has been presented. In Nederstigt et al. (2014a), an adaptation rule learning system which learns the rules and the domain knowledge from a case base has been presented. The work by Nederstigt et al. (2014b) presents a system based on ViPER for extracting information in Web tables using a type of rules called Florid rules. The work by Oberle et al. (2005) presents a hybrid method which combines automatic ontology construction with human intervention to increase the effectiveness of ontology learning. The framework FLOPPIES for ontology population from documents in tabular format has been presented in OpenL Tablets and net (2006). This work is very relevant to the present work, but applied to Web-based e-commerce storage. A comprehensive survey of ontology-based information extractions has been presented in (Protégé ontology editor and edu 2000). Some methods of information extraction from semi-structured documents for ontology population are relevant to the present work.

To enable the use of the rule learning system, the knowledge engineer has to create a training dataset from the source documents and annotate this set of documents with respect to the ontology. The resulting annotation information is pre-processed to create a training dataset for the rule learning system. The rule learning system is then enabled to learn a model from the training dataset. Once the rule learning system learned a model, it can be used to create a set of extraction rules, which can be used by BRMS for populating the ontology with instances from the source documents. The rule learning system is based on an implementation of propositional rule learner named Repeated Incremental Pruning to Produce Error Reduction (RIPPER) (Ren 2014).

In the presented framework, all the information extracted (ontology instances or facts) from the source documents are stored in Positional Slotted Language (POSL) (Schapire and Singer 2014) syntax. Then the POSL facts and rules can be used with a reasoning engine like OO jDREW (Sim and Wong 2004), for further processing.

The remaining sections of the paper are organized as follows: Sect. 2 presents related works. The framework is given in Sect. 3. The details of the annotation system and rule learning systems are presented in Sects. 4 and 5, respectively. An example system is given in Sect. 6 and evaluation of the system is described in Sect. 7. Finally, Sect. 8 concludes this work.

## 2 Related Work

The rule learning system presented in this work mainly focuses on learning classification rules. The system presented can be used to learn robust association rules similar to the rules presented in Simon et al. (2006) if data to be learned is presented in an appropriate format. The rule learning system implicitly uses ontologies in the background, which is equivalent to those of (Simon et al. 2006) to some extent. The system can be considered to be equivalent to that of Uschold and King (1995) in the sense that it can be made to learn decision and association rules. The presented rule learning system uses classification rules for learning ontology instances. Hence, it also can be compared with TRIPPER (Uschold et al. 1998). Since the system uses contextual information to improve the learning of rules, it does incorporate the idea presented in Vasile et al. (2006). The system also has some features of the work by Simon et al. (2006), but the work in Simon et al. (2006) cannot be directly compared with the work presented in this paper, because the work in Simon et al. (2006) presents numerical data streams alone. Annotated documents are similar to example documents; hence, the work in this paper covers the information presented in Uschold and King (1995). The work in this paper uses an annotation system with which the knowledge engineer can annotate documents to learn rules from. Then the rules learned are applied to similar documents to populate the domain ontology. This is equivalent to the concept of learning from examples (Uschold and King 1995).

The proposed rule learning system has a wide variety of applications. In the context of this paper, the rule learning system is used as a part of a framework (Celjuska and Vargas-Vera 2004). The knowledge (ontology and rule)-based framework for the development of business domain applications is presented in Fig. 1.

## 3 The Framework

The components of the presented framework in Fig. 1 are presented in Table 1. The following sections after this section each describe the components from Table 1 in detail.

## 4 Annotation System

The annotation system (developed in Java) enables the knowledge engineer to load the appropriate domain ontology into the system. In addition to the ontology, the system can load source documents, one at a time. Once the ontology and the source documents are loaded into the system, the knowledge engineer can annotate the information in the source documents with the ontology. Each of the selected source documents can be annotated on a row-cell basis by selecting the entities and properties from the ontology. The result of this annotation will be an XML file which contains all the necessary information. This XML file with annotation information is then further used by the rule learning system. For now, it is implemented to work with MS Excel documents. The annotation system also extends support to annotate relationships between entities in the ontology. The system and its graphical user interface (GUI) are implemented using Java. To develop the annotation system, JENA API (Wimalasuriya and Dejing 2010) and Apache POI API (Witten and Frank 2005) were used to load the domain ontology into the system and to load the MS Excel file into the system respectively.

To annotate the necessary cells with the information from the ontology, the user can navigate to the appropriate row using the next and previous buttons, select the appropriate class in the main combo box, and then select

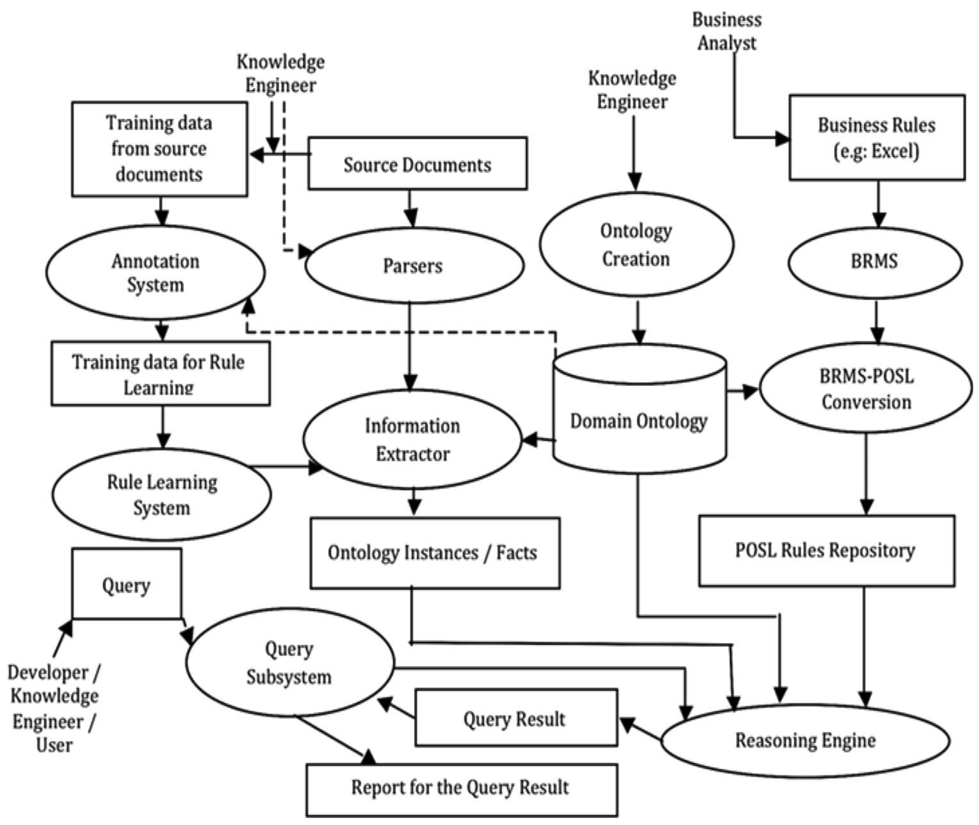


**Fig. 1** The framework for development of business domain applications (Celjuska and Vargas-Vera 2004)

the appropriate property in the combo box above the column to be annotated. The annotation information for the row is captured when the user presses the next or previous button. The user can thus annotate the entire sheet or even the workbook in this manner. After all the necessary cells have been annotated, the annotation information can be stored in an XML file by pressing the save button. A part of the XML file generated by the annotation system is shown in Fig. 9 in "Appendix".

## 5 Rule learning system

The rule learning system (Fig. 2) uses an implementation of a fast rule induction system called Repeated Incremental Pruning to Produce Error Reduction (RIPPER) (Ren 2014). The rule learning system uses the RapidMiner (Zachman 1987) tool's Java API for its implementation. RapidMiner is a tool which includes an implementation of the most successful machine learning and data mining algorithms. It is a tool developed using Java programming language which offers an environment for performing machine learning and data mining experiments. It was formerly known as YALE (Yet Another Learning Environment). It allows experiments to be made up of a large number of arbitrarily nestable operators described in XML files, which can easily be created manually or with its graphical user interface. It also integrates all learning schemes and attribute evaluators of the Weka learning environment (Li et al. 2007), which is another successful environment for machine learning experiments. RapidMiner is a very flexible tool which can be used to implement, test, experiment, and evaluate different machine learning and data mining algorithms on a variety of datasets from application code by just modifying the necessary configuration files.

For now, the rule learning system presented uses the RuleLearner operator of the RapidMiner to perform the rule induction. The reason for choosing RIPPER over other existing rule learning systems like PRISM, single attribute learning, Perceptron, Apriori-type algorithm, Conjunctive rule learner, naïve Bayes, etc. is that the efficiency, precision and recall of the RIPPER over the type of data used as part of this paper are much higher than the others.

The rule learning system starts its role after the annotation system is done with generating the XML files for each of the documents annotated. The XML files with annotation information (a part of a sample XML file is shown in Fig. 9 in "Appendix") are parsed to create the training dataset for the rule learning system. The parsed data are then mapped back to the source documents to extract the appropriate structural information, which would help the rule learning system to extract rules with relatively

**Table 1** Components of the framework (Celjuska and Vargas-Vera 2004)

| Component | Description |
|---|---|
| Source documents | Source documents are the main source of information to the system. They can be documents in user-friendly formats (i.e., Microsoft Word, Excel, etc.) or XML-based formats. The information from the source documents is loaded into the system memory and used for further processing |
| Appropriate parser | The proposed framework assumes the need for appropriate parsers for source documents, which can parse the document enabling the information extractor to retrieve the necessary information needed to populate the ontology |
| Information extractor | An information extractor is needed to extract the facts or ontology instances from the parsed source documents. This information extractor is implemented by using extraction rules created by the rule learner in BRMS format such as OpenL tablets stored in user-friendly document formats such as MS Excel, thereby making the extraction rules user friendly and external to the application. These rules are then used by the OpenL tablets engine to extract the facts or ontology instances from the source documents. In the presented framework, there is also an alternative way to create rules for information extraction using the semi-automatic rule learning subsystem |
| Domain ontology | Domain ontology is one of the main components of the proposed framework. The role of the domain ontology is to bring semantics into the system. The information sources and users may follow different schemes or terminologies. Hence, at least lightweight background ontology must be developed by a knowledge engineer to interoperate between the different schemes. In the presented framework, the domain ontology is supposed to be stored in Resource Description Framework Schema (RDFS) syntax or Ontology Web Language (OWL) syntax, and populated by ontology instances extracted from source documents. These instances are stored in Positional Slotted Language (POSL) syntax (which is another representation of RuleML) because of relative human readability (so, the knowledge engineer can manually intervene with the knowledge/rule-base without difficulties) |
| Business rules | Business rules for the application logic, which may be used for the delivery of appropriate information from source documents to users, can be created and stored in user-friendly formats like Microsoft Excel due to the convenient tabular format, which enables easy human reading and editing. On the other hand, formats like MS Excel are not machine interpretable due to the lack of semantics in documents. The presented framework considers the use of rules in Excel documents to externalize them from specific applications |
| Application logic | A part of the application logic (is a distributed component in the framework) can also be externalized in Excel tables, with the help of BRMS like OpenL tablets, in the proposed framework. As a result, the externalized part of the application logic can be modified by a software developer or knowledge engineer without affecting the other components of the system or even the main system |
| Reasoning engine | The reasoning engine used in the framework loads the domain ontology, converts business rules from the rules repository and the extracted facts from the source documents. Based on the user's choices, queries can be generated from the application or the users can create the queries themselves. These queries are then given to the reasoning engine. The reasoning engine can then run the query and return the results to the application for further processing or for creating a report for the user. Some transformations can also be used to present the results in other user-friendly formats. Although any reasoning engine can be used, the OO jDREW is proposed in the present work because of its capabilities to handle the POSL syntax |
| Annotation system | Annotation system plays a major role in the presented framework. The knowledge engineer can create training dataset from large amounts of source documents. This training dataset is then annotated by the knowledge engineer with respect to the domain ontology. The annotation system is implemented such that the knowledge engineer can load the semi-structured source documents and the domain ontology and can annotate the parts of the documents with respect to the ontology. The annotated information is captured in XML format |
| Rule learning system | The rule learning system in the framework is for learning extraction rules to be used by the information extractor component for populating the ontology. It uses the annotated information to create a training set. The annotated information in XML is mapped back to the source documents to extract some structural information. This extracted structural information along with the annotation information is used to create an organized training set to be used by the rule learning system. The rule learning system then uses the created training set to learn a model for the presented information. The learned model is then used for creating extraction rules. The created extraction rules are then converted to appropriate BRMS format such as OpenL and are given to the information extractor for further use |

higher accuracy. These steps are visualized using the example system in Sect. 6.

The annotation information is given for the rule learning system in an XML file. The XML file contains details from the source document such as the column number or row number, or cell's column and row numbers along with the annotation information: the ontology name, class name, instance and property names and the cell content (a part of a sample XML file is shown in Fig. 9 in "Appendix"). This information is stored in predefined XML tags. The XML tag structure is used to parse the information stored in between the tags. The parsed information is stored in the memory initially and then organized into a tabular Excel format (training data) making it easy for the knowledge engineer to

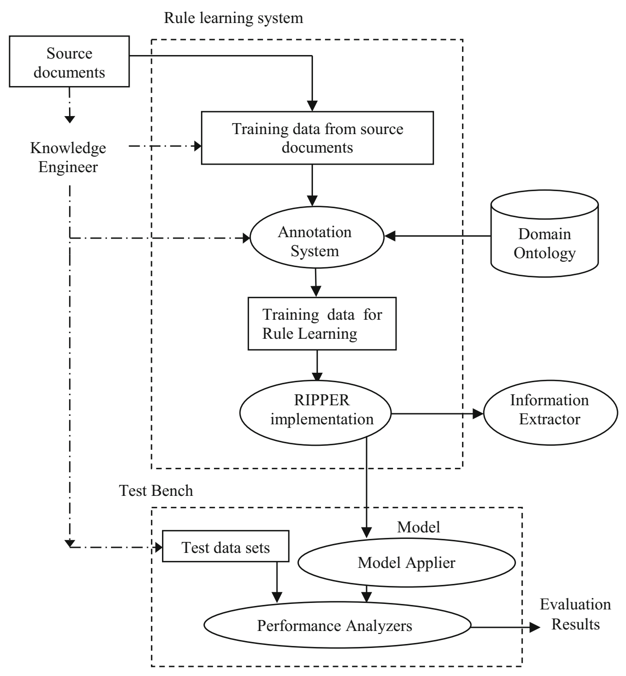


**Fig. 2** The semi-automaticrule learning system with test bench

edit and troubleshoot (a snapshot of a part of the annotation information in tabular format is shown in Sect. 6).

The rule learning is performed incrementally in four stages (depending on the number of attributes to be learned). At the first stage, the remaining attributes are hidden, facilitating the learning of the first attribute. At the second stage of learning, the first unknown attribute column (part of annotation information) is included. At the third stage of learning, the first and second unknown attribute columns (part of annotation information) are included. In the fourth stage, the first, second and third unknown attribute columns (part of annotation information) are included. In each stage, rules for learning the appropriate unknown attributes are created. This type of learning can be called incremental rule learning.

In the presented example, there are four unknown attributes. The order of the attributes plays a very important role. To select the order of the attributes, there is a need to consider the fact that the OpenL tablets rules engine (any other BRMS can be used) overrides the first set of rules with a later set of rules if they address the same data or dataset. Hence, the attribute columns are ordered based on the diversity of the data in the column from least diverse to the most diverse. This is because when there is less diverse data, the accuracy of the created rules is low, and the use of this less diverse data is low. On the other hand, the set of rules created using a high diverse data is much more accurate than the set of rules created using less diverse data. Hence, it is important to give higher priority to the set of rules created using more diverse data.

The learning is performed by the rule learner operator of the RapidMiner tool. The result of the incremental rule learning process is a set of rules in human readable format represented using a group of if–then–else statements. The set of output rules by the RapidMiner tool is then processed to convert them into appropriate syntax to be used by the information extraction system. In the presented example, the rules created by the RapidMiner tool are converted to Java syntax to be used by the OpenL tablets BRMS. Then the new rules are stored in the appropriate rules file to be used by the BRMS. To use the learned rules with OpenL tablets, the attributes (both known and unknown) used for learning the rules can be used to create data types in OpenL. Once OpenL data types are created from the attributes, the converted rules can be used directly with the OpenL engine because the rules work by assigning a value or a range of values for different attributes. As a part of the future work, methods for directly learning the decision table from the training dataset is under progress.

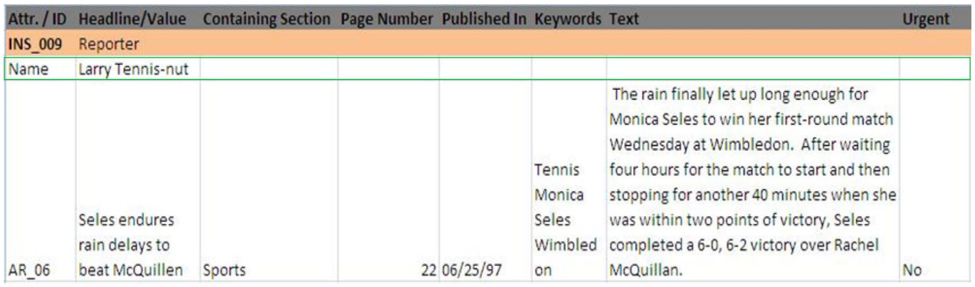
## 6 An Example System

An application of the framework is considered in this section (Fig. 3). A newspaper publishing company is considered as a consumer of the framework. In this example, OpenL tablets is used as the BRMS. Also, to utilize the framework in this company, an ontology belonging to a newspaper domain is needed. A simple lightweight ontology is considered to be used as the background ontology. The ontology used as a part of this work is the newspaper ontology which is available as a part of Protégé (version: 3.3). The rules applicable to this domain, the information about the employees, parsers, and partly the application logic are stored in MS Excel and will be utilized by OpenL tablets.

To explain the developed application in relative detail, a table with information about the employees of the company is considered (Fig. 4). The data in the form of Excel table are parsed by the developed parser.

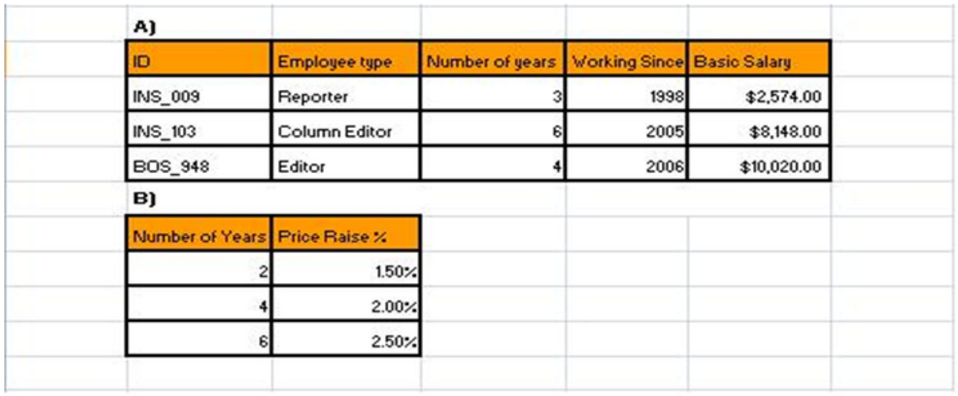
The knowledge engineer can now load the source document (Fig. 3) along with the newspaper ontology into the annotation system and annotate the source document. The annotation information produced by the annotation system is stored in XML file in a predefined format. The information in the XML file is parsed and stored in the Excel file for further processing. A snapshot of the annotation information in the Excel file is presented in Fig. 5.

The information in this Excel file is used to extract contextual information from the source document. A snapshot of some contextual information that has been extracted is shown in Fig. 6.



| Attr. / ID | Headline/Value | Containing Section | Page Number | Published In | Keywords | Text | Urgent |
|---|---|---|---|---|---|---|---|
| INS_009 | Reporter | | | | | | |
| Name | Larry Tennis-nut | | | | | | |
| AR_06 | Seles endures rain delays to beat McQuillen | Sports | 22 | 06/25/97 | Tennis Monica Seles Wimbledon | The rain finally let up long enough for Monica Seles to win her first-round match Wednesday at Wimbledon. After waiting four hours for the match to start and then stopping for another 40 minutes when she was within two points of victory, Seles completed a 6-0, 6-2 victory over Rachel McQuillan. | No |

**Fig. 3** Source data fragment in Excel table format



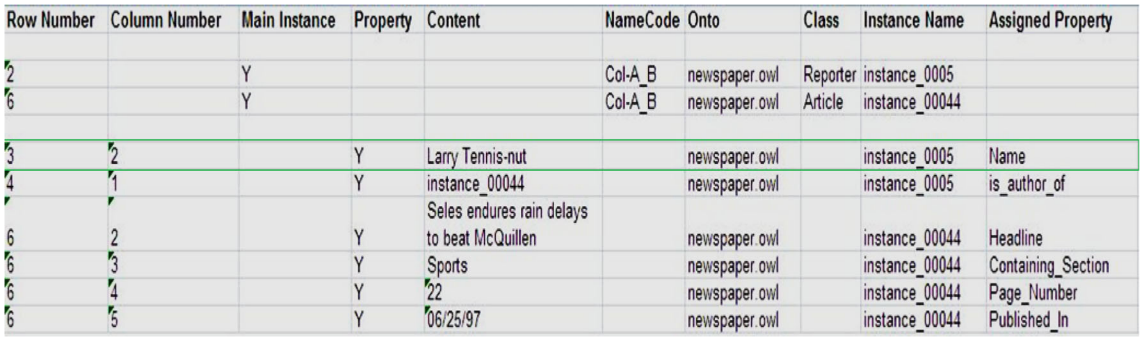
A)

| ID | Employee type | Number of years | Working Since | Basic Salary |
|---|---|---|---|---|
| INS_009 | Reporter | 3 | 1998 | $2,574.00 |
| INS_103 | Column Editor | 6 | 2005 | $8,148.00 |
| BOS_948 | Editor | 4 | 2006 | $10,020.00 |

B)

| Number of Years | Price Raise % |
|---|---|
| 2 | 1.50% |
| 4 | 2.00% |
| 6 | 2.50% |

**Fig. 4** An example of data and rules of a Newspaper publishing company in Excel format



| Row Number | Column Number | Main Instance | Property | Content | NameCode | Onto | Class | Instance Name | Assigned Property |
|---|---|---|---|---|---|---|---|---|---|
| 2 | | Y | | | Col-A_B | newspaper.owl | Reporter | instance_0005 | |
| 6 | | Y | | | Col-A_B | newspaper.owl | Article | instance_00044 | |
| | | | | | | | | | |
| 3 | 2 | | Y | Larry Tennis-nut | | newspaper.owl | | instance_0005 | Name |
| 4 | 1 | | Y | instance_00044 | | newspaper.owl | | instance_0005 | is_author_of |
| 6 | 2 | | Y | Seles endures rain delays to beat McQuillen | | newspaper.owl | | instance_00044 | Headline |
| 6 | 3 | | Y | Sports | | newspaper.owl | | instance_00044 | Containing_Section |
| 6 | 4 | | Y | 22 | | newspaper.owl | | instance_00044 | Page_Number |
| 6 | 5 | | Y | 06/25/97 | | newspaper.owl | | instance_00044 | Published_In |

**Fig. 5** Transformed annotation information

Now, the contextual information extracted (Fig. 6) from the source document is appended to the annotation information (Fig. 5) to create a final training set for the rule learning system. This final training set is further parsed to add some filler into empty cells, because the rule learning system performs better with non-empty cells in the training set. The filler added into the empty cells depends on the contents of the column predominantly. If the column mostly consists of string values, then the filler used to test the system is a string "N/A". If, on the other hand, the column contains mostly integer or float values, negative infinity can be used. This is just for testing. In real world, the knowledge engineer can determine the fillers that will work best for their application and data needs. This final training set is given to the rule learning system incrementally to learn the set of extraction rules for populating the domain ontology.

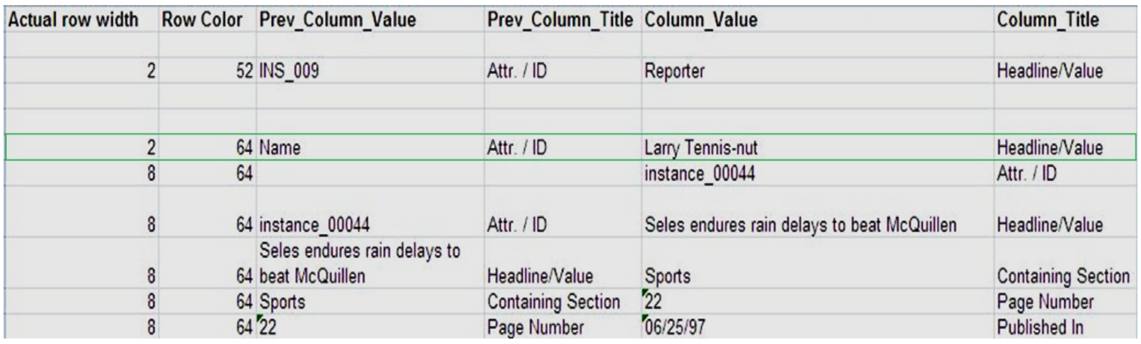
The rule learning system produces a final set of rules after all the stages of the incremental learning process are finished. The rules generated are then processed and converted to appropriate syntax based on the BRMS or other extraction system used. In the presented example, the rules are converted to the Java syntax in a manner to be used with the OpenL tablets. A snapshot of the final rules (user readable) generated is shown in Fig. 7.

The parsed information is then converted into POSL facts by the converter with the consideration of the newspaper ontology in the background. The reason is POSL facts and rules can later be queried or new information can be inferred:

```
Name(name0).
name_val(name0, "Larry Tennis-nut" : String).

Employee(emp0).
hasID(emp0, "INS_009" : String).
hasName(emp0, name0).
isType(emp0, "Reporter" : String).
hasArticle(emp0, art01).
Article(art01).

hasID(art01, "AR_06" : String).
hasHeading(art01, "Seles endures rain delays to beat McQuillen"
: String).
containing_section(art01, "Sports" : String).
page_number(art01, 22 : Integer).

hasKeywords(art01, "Tennis Monica Seles Wimbledon" : String).
text(art01, "The rain finally let up long enough for Monica
Seles to win her first-round match Wednesday at Wimbledon.
After waiting four hours for the match to start and then
stopping for another 40 minutes when she was within two points
of victory, Seles completed a 6-0, 6-2 victory over Rachel
McQuillan. " : String).
isUrgent(art01, "No" : String).
```

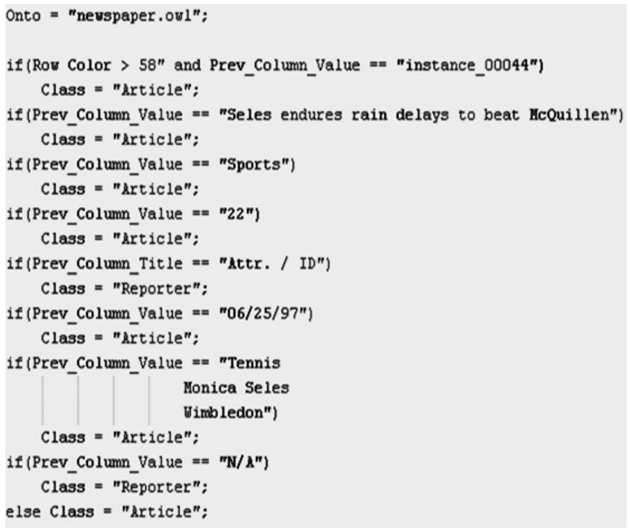
The POSL facts, which are partly an instantiation of the ontology, can be created in two ways in the presented framework: manually using OpenL tablet for rules and schema for facts, and another way is through the semi-automatic rule learning system by manual annotations. For creating POSL facts manually, the knowledge engineer creates conversion rules or OpenL methods in OpenL tablets which convert the data in the table to POSL syntax. This task becomes very cumbersome if the domain has voluminous amount of data in different tables. In this context, the rule learning system becomes very handy, where the knowledge engineer can create a small dataset and annotate them with respect to the domain ontology. The rule learning system now takes the annotated information and creates the necessary extraction rules for populating the ontology. The populated ontology can then be expressed in POSL format. This mechanism can be used to convert the business rules in tables to ontology-based POSL format too.

The POSL syntax of the rules is not shown to the business analyst, making his/her work simple. The business



| Actual row width | Row Color | Prev_Column_Value | Prev_Column_Title | Column_Value | Column_Title |
|---|---|---|---|---|---|
| | | | | | |
| 2 | 52 | INS_009 | Attr. / ID | Reporter | Headline/Value |
| | | | | | |
| | | | | | |
| 2 | 64 | Name | Attr. / ID | Larry Tennis-nut | Headline/Value |
| 8 | 64 | | | instance_00044 | Attr. / ID |
| 8 | 64 | instance_00044 | Attr. / ID | Seles endures rain delays to beat McQuillen | Headline/Value |
| 8 | 64 | Seles endures rain delays to beat McQuillen | Headline/Value | Sports | Containing Section |
| 8 | 64 | Sports | Containing Section | 22 | Page Number |
| 8 | 64 | 22 | Page Number | 06/25/97 | Published In |

**Fig. 6** Information obtained from source document



```
Onto = "newspaper.owl";

if(Row Color > 58" and Prev_Column_Value == "instance_00044")
    Class = "Article";
if(Prev_Column_Value == "Seles endures rain delays to beat McQuillen")
    Class = "Article";
if(Prev_Column_Value == "Sports")
    Class = "Article";
if(Prev_Column_Value == "22")
    Class = "Article";
if(Prev_Column_Title == "Attr. / ID")
    Class = "Reporter";
if(Prev_Column_Value == "06/25/97")
    Class = "Article";
if(Prev_Column_Value == "Tennis
                        Monica Seles
                        Wimbledon")
    Class = "Article";
if(Prev_Column_Value == "N/A")
    Class = "Reporter";
else Class = "Article";
```

**Fig. 7** A snap shot of classification or extraction rules generated by the rule learning system

analyst or the respective official would like to maintain the information about the employees and at the same time keep track of, for example, their salaries to different employees. For this, he/she can develop a set of rule tables where the rules applied to the different types of employees are stored (Fig. 4 part A). The system takes these rules and converts them into appropriate POSL facts and rules. Once these rules are in POSL format, the business analyst or the respective official can start querying the knowledge base to get necessary information about employees. Some of the POSL facts created from the rules in Excel table are:

```
number_years("INS_009" : String, 3 : Integer).
year"INS_009" : String, 1998 : Integer).
basic_sal"INS_009" : String, 2574.00 : Real).
```

Using the above-mentioned facts, the respective official can query the knowledge base for obtaining information about employees. To use these facts to get information about a particular employee, the system also generates rules which can be used to receive information on new salary for any employee. When the respective official wants to receive information from the knowledge base, the system calls a method generating an internal query to the knowledge base using the OO jDREW (Sim and Wong 2004) engine in the backend. These queries get automatically called based on the selections made by the respective official. In the case with a knowledge engineer, he/she can query the knowledge base directly in the GUI of the OO jDREW engine, because he/she will probably be familiar with the POSL syntax. The system delivers the results of the query to the person querying when the queries are issued using the system. There might be situations when the user might have been instructed by the organization, for instance, to raise the salary of an employee or a set of employees on a specific occasion like a bonus for many or promotion for a group. In this case, the respective official just has to create another table which specifies the percentage raise information on the salary for each class of employee and the knowledge engineer has to create rule schema or annotation information for this new table. A similar scenario is shown in Fig. 4 part B, where the respective official can create rules to apply a uniform percentage raise to employees of a class. From the table presented in Fig. 4 part B, the rules for inferring the new salary raise will look as follows:

```
% New salary raise for a class of employees

percentage_raise?b : Integer, 0.015 : Real) :-
greaterThanOrEqual(?b, 2 : Integer), lessThan(?b, 4 : Integer).


% New Salary based on years of service

new_basic_sal?a : String, ?c : Real) :- number_years(?a :
String, ?b : Integer), basic_sal(?a : String, ?x : Real),
percentage_raise(?b : Integer, ?d : Real), add(?y, 1 : Integer,
?d : Real), multiply(?c, ?x : Real, ?y).
```

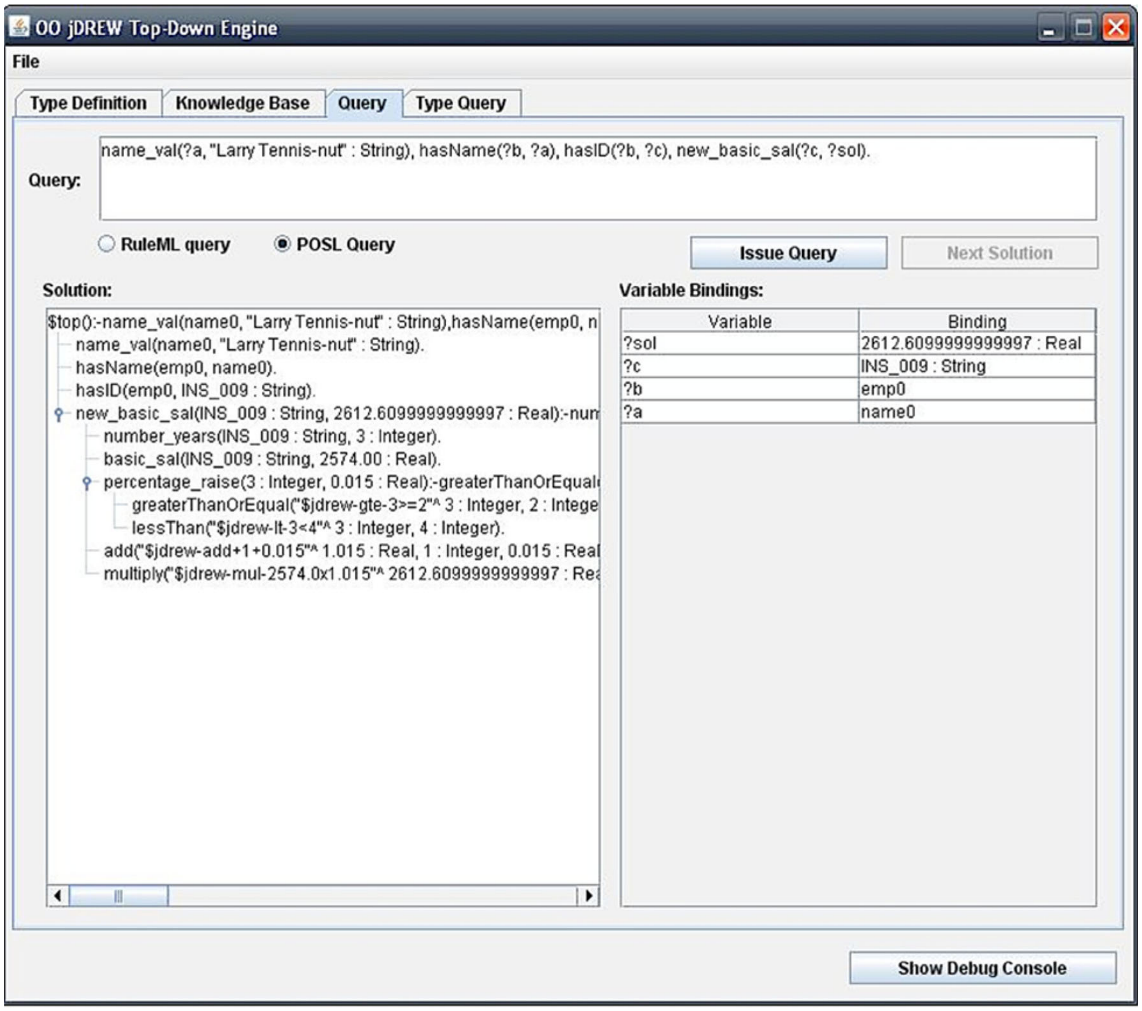
Once these rules are created, the system can automatically recognize the new rules and incorporate them. Now, if the knowledge engineer or the respective official queries the system, they will be getting the updated information for different employees based on the new percentage increase added into the system. The domain ontology is loaded into OO jDREW reasoning engine through the Java API and parsed. Then the created facts and rules of the system are loaded into the reasoning engine and parsed. To test the system, the following query is issued to the reasoning engine:

```
%The query asks, Give the new basic salary for a person named
"Larry Tennis-nut"

name_val ?a, "Larry Tennis-nut" : String), hasName(?b, ?a),
hasID(?b, ?c), new_basic_sal(?c, ?sol).
```

The query result is stored in the variable—?c as ?c = 2612.609. A snapshot of OO jDREW reasoning engine running the query is shown in Fig. 8.

## 7 Evaluation

In the context of the present paper, a rule learning system and a framework are presented. The rule learning system is the focus of the work presented. A test bench was created to test the rule learning system incrementally. The test bench evaluates the rule learning system during each incremental phase and gives results in terms of precision and recall. The test bench was created using the testing operators available in RapidMiner. As explained earlier, the rule learning system learns the rules for the unknown attributes from the known attributes incrementally. After the model for each unknown attribute is learned, the model learned for that unknown attribute is evaluated immediately. Testing was performed on 1:3 training to test the set basis (for every model learned it has been tested on three test datasets). The knowledge engineer has to create the test datasets by giving the values for each of the unknown attributes. The created test datasets are then given to the test bench which then loads the learned model and predicts each of the unknown attributes in an incremental manner. Then the test bench compares the predicted values for the unknown attributes with values for this unknown attributes given by the knowledge engineer. Based on this



Query: name_val(?a, "Larry Tennis-nut" : String), hasName(?b, ?a), hasID(?b, ?c), new_basic_sal(?c, ?sol).

Solution:

```
$top():-name_val(name0, "Larry Tennis-nut" : String),hasName(emp0, n
  name_val(name0, "Larry Tennis-nut" : String).
  hasName(emp0, name0).
  hasID(emp0, INS_009 : String).
  new_basic_sal(INS_009 : String, 2612.6099999999997 : Real):-num
    number_years(INS_009 : String, 3 : Integer).
    basic_sal(INS_009 : String, 2574.00 : Real).
    percentage_raise(3 : Integer, 0.015 : Real):-greaterThanOrEqual
      greaterThanOrEqual("$jdrew-gte-3>=2"^ 3 : Integer, 2 : Intege
      lessThan("$jdrew-lt-3<4"^ 3 : Integer, 4 : Integer).
    add("$jdrew-add+1+0.015"^ 1.015 : Real, 1 : Integer, 0.015 : Real
    multiply("$jdrew-mul-2574.0x1.015"^ 2612.6099999999997 : Re
```

Variable Bindings:

| Variable | Binding |
|---|---|
| ?sol | 2612.6099999999997 : Real |
| ?c | INS_009 : String |
| ?b | emp0 |
| ?a | name0 |

**Fig. 8** A snap shot of query result in OO jDREW

**Table 2** Evaluation results

| Similarity between the training data and test data | Precision overall (%) | Recall overall | *F* measure overall |
|---|---|---|---|
| >80% | 100.00 | 100.00 | 100.00 |
| >60% but <=80% | 83.40 | 84.60 | 84.00 |
| > 40% but <=60% | 78.80 | 79.68 | 79.24 |

comparison, precision, recall and F-measure are calculated for each attribute for each test data. In the context of this paper, precision is defined as the amount of relevant information retrieved by a search divided by the total information retrieved by that search, and Recall is defined as the amount of relevant information retrieved by a search divided by the total existing relevant information. The overall evaluation results for the evaluation conducted are presented in Table 2. For the test data with equivalent similarity with the training data, the evaluation results are almost similar.

The performance of the system depends on a variety of factors like the quality and quantity of the training dataset, similarity between the test and training dataset, and the method used to generate the test data. The quality and quantity of the training dataset is dependent on the way the training set is created. The performance of the model learned will be much better if each unknown attribute has sufficient training values to learn the model from. The distribution or uniformity of the number of values used to train each unknown attribute again plays a very important role in the quality of the model learned.

The similarity between the training dataset and test dataset is a very common factor which plays a very important role in the result of evaluation. To evaluate the rule learning system, different test datasets have been created with different levels of similarity between the training dataset and the test dataset. Three cases have been considered, greater than 80% similarity, greater than 60% but less than or equal to 80% similarity, and greater than 40% but less than or equal to 60%.

The method used to generate the test dataset plays a very important role in the result of evaluation. The most general methods for generating test data are: manual creation and automatic creation of test dataset. As a part of this paper, the test dataset is created by the knowledge engineer manually. Hence, it is believed that the percentage of error is minimal. Also, the test dataset is created in a way that it is different from the training dataset beyond a certain percentage (80, 60, 40% similarity), etc. The similarity here mainly means the similarity in the format of the contents in the document and very minor repetition of content.

The evaluation of the framework itself is not very important in the context of the present paper, but an evaluation of the framework with the rule learning system might be of interest. The best way to evaluate the presented framework is to compare it with a successful framework in its cadre. There are four classes of enterprise architecture frameworks: open source or consortia-developed frameworks, commercial frameworks, defense industry frameworks, and government frameworks.

A very popular and successful framework for enterprise architecture is Zachman framework for enterprise architecture (Zolghadri-Jahromi and Valizadeh 2006). To compare the presented framework with the Zachman framework, a comparison is conducted between the components of the two frameworks to determine how well the presented framework fits into the Zachman framework. The presented framework for business domain applications covers the Zachman framework. All the data perspectives are covered by the domain ontology and its model presented as a part of the framework. All the functional perspectives are covered by the reasoning engine used and externalized application logic made possible by the use of an appropriate BRMS (like OpenL tablets). All the network perspectives are covered by the relationships between the components of the presented framework. All people perspectives are covered by the possibility of independent development and maintenance of the components of the framework. All time and motivation perspectives are covered by the design and implementation of rules, and their semantic processing, and the entire framework.

## 8 Conclusion

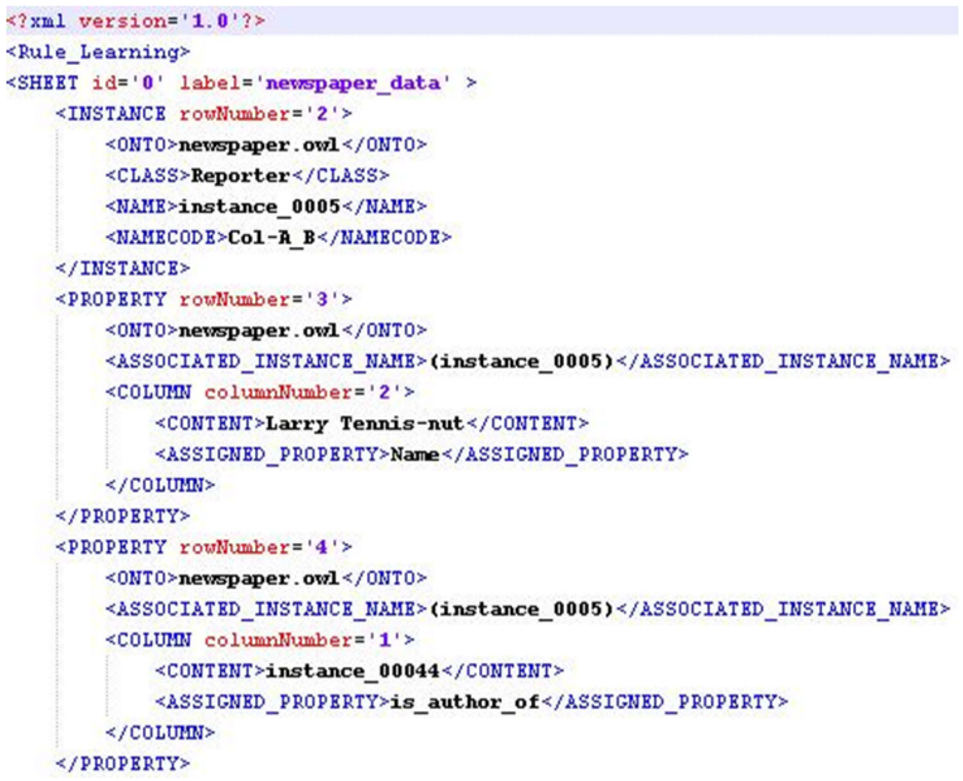
The present paper has described a semi-automatic rule learning system, which creates information extraction rules semi-automatically with the help of manual annotations. The paper describes a novel approach to the rule learning system for ontology population from business documents stored in tabular formats (e.g., MS Excel). The presented rule learning system utilizes fast rule induction algorithm (RIPPER) and the semantic Web techniques with the purpose of extracting ontology instances and imparting semantics to the business documents. The documents in tabular formats are first processed by the annotation system, which produces corresponding XML files with annotation information; this information along with some contextual information is processed by the rule learning system to learn the model of the system and create classification or extraction rules to populate the ontology. The fast rule induction algorithm (RIPPER) used by the rule learning system is implemented as part of the RapidMiner tool. A rule learning example is presented as a proof of concept. The reason for creating a semi-automatic rule learning system for learning classification or extraction rules is that in industry (since the data dealt with is sensitive) the knowledge engineers would need to have more control over the system, also they would like to have better accuracy, precision and recall which are higher in the case of semi-automatic systems than fully automatic systems.

A part of the paper presented a framework for building knowledge-based software applications and also presented the use of the rule learning system in such a framework. The presented framework works with domain ontology in the background. The framework assumes full independence of development of all its components (ontology, business logic/business rules, application logic, and reasoning) on each other. The presented framework combines the advantages of the semantic Web computing techniques, OpenL as the framework works toward the externalization and semantic enrichment of business rules for their machine interpretability, and semantic querying of business rules. The success of a business domain software application built using this framework depends on the quality of each single component. Overall, the presented framework has all the components and details necessary for performing successful business modeling.

Future research in the direction shown by the presented framework is multi-dimensional. An important further direction is an extension of the developed framework to support other types of information systems (different from business domain software applications). Future research will work toward providing support for multiple ontologies in the presented framework. The presented framework has an information extraction component which, for now, only populates the domain ontology. This direction will be to include information extraction methods for extracting complete ontologies from source documents. Future research will also include providing support for a large variety of document formats, including other semi-structured formats to unstructured formats, as source documents. Another area of future research, even though challenging, will be to extract business rules automatically from business documents, which will lead to further software requirements elicitation from business documents.

## Appendix

See Fig. 9.



```
<?xml version='1.0'?>
<Rule_Learning>
<SHEET id='0' label='newspaper_data' >
    <INSTANCE rowNumber='2'>
        <ONTO>newspaper.owl</ONTO>
        <CLASS>Reporter</CLASS>
        <NAME>instance_0005</NAME>
        <NAMECODE>Col-A_B</NAMECODE>
    </INSTANCE>
    <PROPERTY rowNumber='3'>
        <ONTO>newspaper.owl</ONTO>
        <ASSOCIATED_INSTANCE_NAME>(instance_0005)</ASSOCIATED_INSTANCE_NAME>
        <COLUMN columnNumber='2'>
            <CONTENT>Larry Tennis-nut</CONTENT>
            <ASSIGNED_PROPERTY>Name</ASSIGNED_PROPERTY>
        </COLUMN>
    </PROPERTY>
    <PROPERTY rowNumber='4'>
        <ONTO>newspaper.owl</ONTO>
        <ASSOCIATED_INSTANCE_NAME>(instance_0005)</ASSOCIATED_INSTANCE_NAME>
        <COLUMN columnNumber='1'>
            <CONTENT>instance_00044</CONTENT>
            <ASSIGNED_PROPERTY>is_author_of</ASSIGNED_PROPERTY>
        </COLUMN>
    </PROPERTY>
```

**Fig. 9** A part of the XML file with annotation information

## References

Alpaydin E (2004) Introduction to machine learning, MIT Press

Apache POI HSSF API (2002) http://poi.apache.org/spreadsheet/index.html (Accessed 2014)

Bach NX, Cuong LA, Ha NV, Binh NN (2008) Transformation rule learning without rule templates: a case study in part of speech tagging. In: Proceedings of the. International Conference on Advanced Language Processing and Web Information Technology, pp. 9–14

Ball M, Boley H, Hirtle D, Mei J, Spencer B (2005) The OO jDREW reference implementation of ruleml. In: Proceedings of the rules and rule markup languages for the semantic web (RuleML-2005), Springer LNCS 3791, pp. 218–223

Behkamal B, Naghibzadeh M, Askari Moghadam R (2012) Pre-processing ontologies to improve the results of matchers. Iran J Sci Technol Transact Elect Eng 36(E2):95

Berners-Lee T, Hendler J, Lassila O (2001) The semantic web. Sci Am 5:34–43

Biletskiy Y, Ranganathan GR (2008) An invertebrate semantic/software application development framework for knowledge-based systems. Knowl-Based Syst 21(5):371–376

Boley H (2004) POSL: An Integrated Positional-Slotted Language for Semantic Web Knowledge, http://www.ruleml.org/submission/ruleml-shortation.html (accessed 2014)

Buitelaar P, Cimiano P, Frank A, Hartung M, Racioppa S (2008) Ontology-based information extraction and integration from heterogeneous data sources. Int J Hum Comput Stud 66:759–788

Celjuska D, Vargas-Vera M (2004) Ontosophie: A semi-automatic system for ontology population from text. In: Proceedings of the 3D International Conference on Natural Language Processing (ICON)

Cimiano P (2006) Ontology learning and population from text: algorithms, evaluation and applications. Springer-Verlag, New York

Cohen WW (1995) Fast effective rule induction (RIPPER). In: Proceedings of the 12th International Conference on Machine Learning, pp. 115–123

Ferrer-Troyano F, Aguilar-Ruiz JS, Riquelme JC (2005) Incremental rule learning based on example nearness from numerical data streams, Symposium on Applied Computing. In: Proceedings of the ACM symposium on Applied computing, pp. 568–572

Fox MS, Barbuceanu M, Gruninger M (1996) An organization ontology for enterprise modelling: preliminary concepts for linking structure and behaviour. Comput Ind 29:123–134

Fox MS, Gruninger M (1998) Enterprise modelling. AI Magazine 19(3):109–121

Gomez-Perez A, Corcho O, Fernandez-Lopez M (2004) Ontological Engineering: with examples from the areas of Knowledge Management, e-Commerce and the Semantic Web. Advanced information and knowledge processing. Springer, London

Holmes G, Donkin A, Witten H (1994) Weka: a machine learning workbench. In: Proceedings of the Second Australia and New Zealand Conference on Intelligent Information Systems, pp. 357–361

Jena API (2011) https://jena.apache.org/, accessed 2014)

Li H, Hu D, Hao T, Wenyin L, Chen X (2007) Adaptation rule learning for case-based reasoning. In: Proceedings of the Third International Conference on Semantics, Knowledge and Grid, pp. 44–49

Maedche S, Staab S (2001) Ontology learning for the semantic web. IEEE Intell Syst Archive 16(2):72–79

Manine P, Alphonse E, Bessieres P (2008) Information extraction as an ontology population task and its application to genic interactions. In: Proceedings of the 20th IEEE International Conference Tools with Artificial Intelligence. 2:74–81

Mierswa I, Wurst M, Klikenberg R, Scholz M, Euler T (2006) YALE: rapid prototyping for complex data mining tasks. In: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-06), pp. 935–940

Nederstigt LJ, Aanen SS, Vandic D, Frasincar F (2014a) FLOPPIES: a framework for large-scale ontology population of product information from tabular data in E-commerce stores. Decis Support Syst 59:296–311

Nederstigt LJ, Aanen SS, Vandic D, Frasincar F (2014b) FLOPPIES: a framework for large-scale ontology population of product information from tabular data in e-commerce stores. Decis Support Syst 59:296–311

OWL: Web Ontology Language, http://www.w3.org/2004/OWL (2004, accessed 2014)

Oberle D, Staab S, Studer R (2005) Supporting application development in the Semantic Web. ACM Trans Internet Technol 5(2):329–358

OpenL Tablets, http://openl-tablets.sourceforge.net/(2006, accessed 2014)

Protégé ontology editor, http://protege.stanford.edu/(2000, accessed 2014)

Ren F (2014) Learning time-sensitive domain ontology from scientific papers with a hybrid learning method. J Info Sci 40(3):329–345

Schapire R, Singer Y (2014) SLIPPER, http://www.cs.cmu.edu/~wcohen/slipper/(1999, accessed 2014)

Sim KM, Wong PT (2004) Towards agency and ontology for web-based information retrieval. IEEE Trans Syst Man Cybern C Appl Rev 34(3):257–269

Simon K, Hornung T, Lausen G (2006) Learning rules to pre-process web data for automatic integration. In: Proceedings of the. Second International Conference on Rules and Rule Markup Languages for the Semantic Web, pp. 107–116

Uschold M, King M (1995) Towards a methodology for building ontologies. The IJCAI-95 Workshop on Basic Ontological Issues in Knowledge Sharing. pp. 15–30

Uschold M, King M, Moralee S, Zorgios Y (1998) The Enterprise Ontology. The Knowledge Engineering Review, Special Issue on Putting Ontologies to Use. pp. 31–89

Vasile F, Silvescu A, Kang DK, Honavar V (2006) TRIPPER: rule learning using taxonomies. In: Advances in knowledge discovery and data mining, Springer Berlin Heidelberg, pp. 55–59, 2006

Wimalasuriya DC, Dejing D (2010) Ontology-based information extraction: an introduction and a survey of current approaches. J Info Sci 36(3):306–323

Witten H, Frank E (2005) Data mining: practical machine learning tools and techniques (2nd ed). Morgan Kaufmann series in data management systems

Zachman A (1987) A framework for information systems architecture. IBM Syst J 26(3):276–292

Zolghadri-Jahromi M, Valizadeh MR (2006) A proposed query-sensitive similarity measure for information retrieval. Iran J Sci Technol Trans B Eng 30(B2):171–180