symmetry

MDPI

Article

Handling Semantic Complexity of Big Data using Machine Learning and RDF Ontology Model

Rauf Sajjad *, Imran Sarwar Bajwa and Rafaqut Kazmi

Department of Computer Science & IT, The Islamia University Bahawalpur, Bahawalpur 63100, Pakistan; imran.sarwar@iub.edu.pk (I.S.B.); rafaqutkazmi@gmail.com (R.K.)

* Correspondence: raufsajjad786@gmail.com; Tel.: +92-62-925-5466

Received: 3 January 2019; Accepted: 14 February 2019; Published: 1 March 2019

Abstract: Business information required for applications and business processes is extracted using systems like business rule engines. Since the advent of Big Data, such rule engines are producing rules in a big quantity whereas more rules lead to more complexity in semantic analysis and understanding. This paper introduces a method to handle semantic complexity in rules and support automated generation of Resource Description Framework (RDF) metadata model of rules and such model is used to assist in querying and analysing Big Data. Practically, the dynamic changes in rules can be a source of conflict in rules stored in a repository. It is identified during the literature review that there is a need of a method that can semantically analyse rules and help business analysts in testing and validating the rules once a change is made in a rule. This paper presents a robust method that not only supports semantic analysis of rules but also generates RDF metadata model of rules and provide support of querying for the sake of semantic interpretation of the rules. The results of the experiments manifest that consistency checking of a set of big data rules is possible through automated tools.

Keywords: Semantic complexity; Big Data; machine learning; rules; ontology; RDF

1. Introduction

In a modern enterprise, the process of decision making is typically assisted by a knowledge-base and a system for reasoning. The rules are key source of information for business applications and business processes. Such information is generated from existing systems and data using business rule engines [1]. However, such business rules engines tend to generate a large volume of rules if applied on Big Data. The large volume of rules likewise increases semantic complexity and leads to difficult rule management and rule analysis due to possible inconsistencies looming after manual changes are made to a set of business rules. However, semantically consistent rules facilitate the process of decision making. Furthermore, the modern business rules are now expressed in SBVR (Semantic of Business vocabularies and Rules) [2] introduced by OMG (Object Management Group) that works in coordination with other OMG standards like BPMN (Business Process Model Notation), MDA (Model Driven Architecture), etc. [3]. There is a need of a platform that handles semantic complexity of such SBVR business rules extracted from Big Data.

With the advent of SBVR, it has become simple and efficient to describe a set of rules in SBVR to effectively run business operations [4]. A business rule management system (BRMS) is typically used in an enterprise to manage and deploy hundreds of such rules and smoothly running the operations of an enterprise. The implementation of such BRMS in an organization assists in verification and validation of rules [5]. Such consistency checking becomes more critical if the rules are extracted from Big Data. The consistency verification of a new rule or an existing rule in a rule repository may have conflict with some other rule in the repository. Such conflicting or inconsistent rules can hurdle operations of an enterprise and may bring a whole system into an inconsistent state. Considering

this challenge, it was necessary to design a framework to facilitate a business analyst in tool-based validation and consistency checking of rules by finding out that a rule is consistent by using the rules' background knowledge [6].

In modern information systems, knowledge management has emerged into a major research area due to significance of knowledge-based decision making. The field of knowledge management relies on ontologies to store and manage business knowledge. The design and development of a new ontology involves defining concepts, classes, their properties, their relationships, cardinalities, etc. However, using RDF (Resource Description Framework) [7,8] is a simple way of generating domain-specific meta-data. In a knowledge-based decision making frameworks, a set of domain ontologies along a new ontology are used. However, a semantic model is required for better interpretation of business vocabulary and business rules whereas ontologies are key source of semantic modelling using SWRL (Semantic Web Rule Language) [9], OWL (Ontology Web Language) [9], etc. The process of rule generation from Big Data using a business rule engine is shown in Figure 1. The typical steps are rule extraction, storing in rule repository, deployment, testing, validation, management, etc.

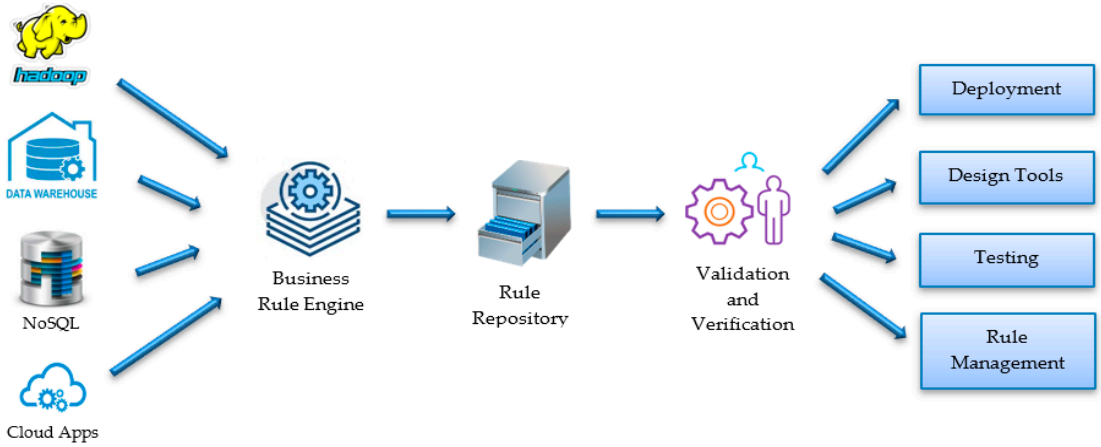

Figure 1. A typical business rule extraction system for Big Data.

During the literature survey, it was found that automated and intelligent verification of rules generated in rule harvesting (see Figure 1) process is not available theoretically or by tool. However, such automated support for rule verification provides the ability of consistency validation of business rules stored in a rule repository. Similarly, when dynamic changes are made in a rule in the repository, it tends to semantically inconsistent with respect to the other rules in the repository [10].

In this paper, a novel approach is presented that uses machine learning technique to classify the SBVR rule meta-data and to map it to an RDF model. Such RDF representation of a rule can be easily verified using online web-services such as W3C validator [11]. This approach can simplify the consistency checking of two business rules by generating their RDF graphs. Afterwards, SPARQL (a RDF query language) based queries can help in validating the rules. Once the semantic validation is confirmed of a new rule, it can be stored in the rule repository and can be deployed in the system. A novel contribution in this paper is revised version of EM (Expectation-Maximisation) algorithm for knowledge representation and an approach to semantically annotate rules metadata using RDF schema [12]. Once an RDF schema is generated, the rules are easy to validate for its consistence using W3C validator and other similar web services.

In automation of generating process of rule metadata model, a major challenge was automatic parsing of the rules and semantic interpretation of natural language rules for extraction of vocabulary and finally mapping the extracted vocabulary to RDF syntax [13] and finally verification of the resultant RDF schema [14]. A framework (Figure 2) is presented in Section 4 that is a novel contribution and it uses algorithm 1 to parse the rules and extract vocabularies and finally, Jena library is used to generate RDF schema of the extracted vocabulary [15].

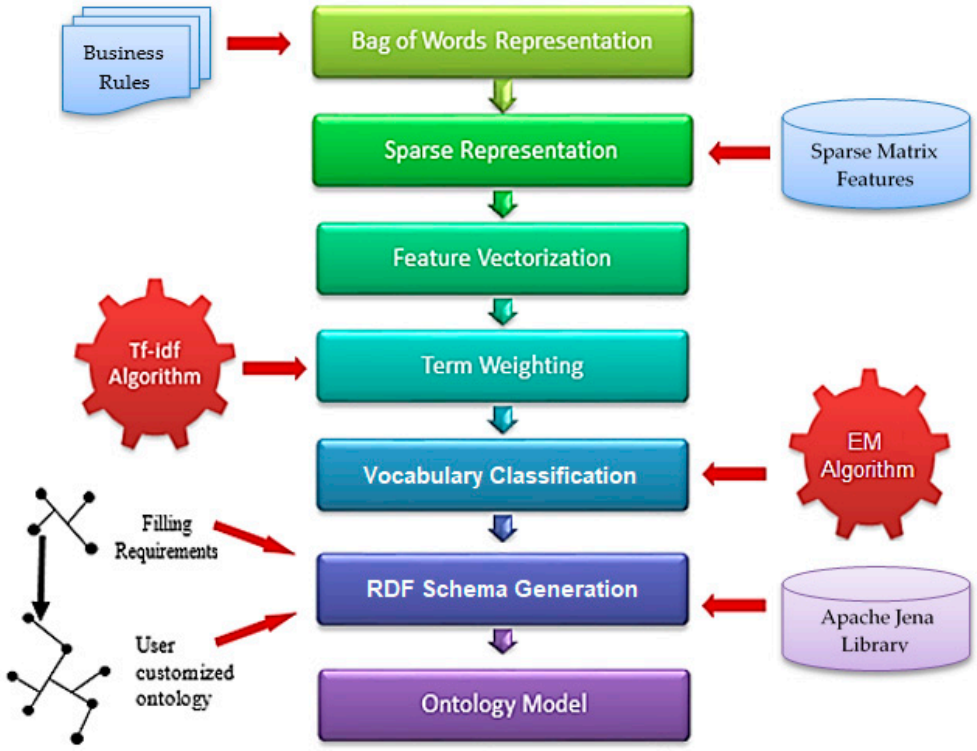

Figure 2. Used Framework for rule interpretation. EM: Expectation-Maximisation; RDF: resource description framework; tf-idf: Term Frequency–Inverse Document Frequency.

The rest of this paper is structured into a set of sections: Section 2 describes related work to business rules and business rule management systems, Section 3 discusses the EM-based algorithm and Section 4 describes the used framework for automated generation of rule metadata model. Section 5 discusses the experiments and results and related wok is discussed in Section 6 and finally paper is concluded along the future work.

2. Related Work

Ontologies are used for the support of knowledge extraction but still this approach is not fully utilised [14]. To remove ambiguity in the queries query expansion techniques are used which help to avoid ambiguity in the queries. There are a few ontology-based query expansion techniques whose purpose is to avoid ambiguities [15]. Ontologies are also base of semantic web. The semantic web is considered "web of linked data" [16] and in semantic web there are groups of methods and techniques that are used by computers to understand the manning of the information that is available on the web. Formal ontologies are very important for the Semantic Web. The existing ontologies present in literature have intersection and union of anonymous classes. The anonymous classes are those whose scope is limited, and these classes may not be a part of a whole process. Ontologies can be used within bioinformatics. Biologists performs their work with the help of pre-existing knowledge and also make references about the item that has been investigated [17,18]. In the literature many methodologies, tools and languages have been mentioned to build ontologies. Ontology technology provides us with many tools and methodologies. To integrate information in IT systems, the reference ontologies are used [18]. To specify syntax semantic interface for an ontology-based NLP system, both lexicon and ontology can be used, and ontology-based NLP systems produce output that is according to specific domain ontology [19,20]. Typically, in text, manual annotations are not practical and scalable. On the other hand, automatic annotation tools are still not matured enough to be used. Hence for special knowledge services some tools are required to directly search and extract knowledge [21]. The discussed work shows that there is need for an approach to assist in querying and interpreting the rules. As a dynamic

alteration in a typical rule can lead into a conflict in rules stored in a repository, the semantic analysis of a rule can help an analyst in testing and validating it. In this paper, we address this problem by using an automated approach to support semantic analysis of rules and generate RDF metadata model of rules and provide support of querying for the sake of interpretation of the rules.

The process of identification and documentation of business rules is called rule harvesting [22]. In a typical rule harvesting process, the rules are identified from candidate rules and true business rules are extracted and are stored in a rule repository. Since, the manual extraction is not only difficult but also infeasible, a tool support for rule harvesting can be quite helpful. In the recent past, a few tools are presented for rule harvesting however these tools lack support for finding and measuring rule relevance, rule completeness, rule consistency etc. The majority of these rule harvesting methods do not have support for finding and measuring semantic complexity and consistency. Moreover, the combination of manual and automated efforts gives the better result of rule harvesting [23]. With the lack of adherence to development guidelines, it is difficult for the application developer to navigate through the code and naming conventions. The use of conventions can make understanding the logic in business difficult.

In a traditional legacy application, it is difficult to negotiate the technical requirements for the implantation of rules in terms of code. Typically, a business rule assists both the person running the application and the end user of the application. The bottom-up rules approach causes problems in classification of the concepts [24]. Here, the context approach is use for collecting the rules. In an automated rule harvesting software, it is a tricky process to retrieve and collect business rules. Manual rules are clear, but rules extracted from code may be inconsistent or ambiguous. However, in manual collection, there is no information about the actual location of a rule implementation that leads to rule harvesting from code. On the other hand, automated rule extraction is fast and efficient, but search criteria is not sufficiently defined. Rule harvesting from a existing code may require a project team to collect rules [25] and document rules with the help of deep knowledge of the business. Similarly, a team may need to collect the processes which are used for express use-case diagrams, activity diagrams, and then store the rules into a database repository where they are connected with each other. Finally, rules are handed to the developer who implements the processes using the rules with the new, detailed knowledge.

Business Rules Management System (BRMS) [26] is software that is used to implement a set of business rules. In a typical BRMS, functionalities are provided such as defining new business rules, deploying the business rules, executing the business rules and monitoring and maintaining the rules.

A BRMS is quite helpful in better management of business rules and finding ways of decision making [27]. Additionally, a BRMS can help in identifying inconsistent and ambiguous rules. A BRMS is a common solution that is used to define, deploy and maintain operation of a system in organisation and logic, also referred to as business rules, policies, or the condition that take action in system. In a business management system, the business rules engine plays a vital part in the execution of one or more rules at run time in the system [28]. Business rule engine is software that provides the ability to register, define, classify and manage all the business rules. In the collection of business rules, one or more rules are in a group or cluster on a server that is ready to use import and export. Moreover, the business rules search is key-based search.

3. EM-Based Classification of RDF Model Elements

The presented system uses an ontology model for the interpretation of rules. The presented system is a domain-specific solution and has all the details regarding a system specific to the domain. The key phase in generating RDF ontology model from business rules, is classification of vocabulary items of a SBVR rule to RDF model elements. In the used approach, a machine learning-based approach is presented based on Expectation-Maximisation (EM) [17] technique.

The Expectation-Maximisation technique plays an important role in the used machine learning-based approach for mapping SBVR business vocabulary concepts to RDF model elements.

A typical business rules has various vocabulary concepts such as noun concepts, object types, individual concepts and verb concepts. These concepts are needed to be mapped to RDF model elements such as resources, properties and classes. EM is a semi supervised algorithm used for classification of vocabulary concepts especially mentioned in natural language documents [18]. A benefit of EM technique is its ability to complete missing data in the training data. In expectation phase of EM technique, a probable value is expected for each missing value in data using the maximum likelihood approximation of existing values in the data [18]. In the maximisation phase of the EM technique, the parameters are estimated on the basis of filled data.

The EM technique works on a set of inputs R_l of labelled rules and R_u of unlabelled rules written in SBVR English. Equation (1) shows that ways naive Bayes classifier is used to perform expectation and maximisation phases on training data. The labelled rules are used in by EM initially and then the process of Equation (1) is repeated iteratively.

$$P(c_j \mid v_i) = \frac{P(c_j) \prod_{k=1}^{|v_i|} P(v_i \mid c_j)}{\sum_{r=1}^{|c|} P(c_r) \prod_{k=1}^{|v_i|} P(v_i \mid c_j)} \tag{1}$$

In Equation (2), a set of SBVR business rules are represented by R and an ordered list of vocabulary items is represented by v. Here, a vocabulary concept in an SBVR rule r_i is represented by v_i, such as $V = < v_1, v_2, \ldots, v_{|n|} >$ whereas, the targeted elements of the RDF model are represented as classes and a set of classes is denoted as $C = \{c_1, c_2, \ldots, c_{|n|}\}$. The EM technique is used to map vocabulary items V to the RDF model elements C. For mapping V to C, posterior probability, $P(c_j \mid v_i)$ is used to classify the elements of set V into possible classes C. Here, the posterior probability, $P(c_j \mid v_i)$, where v_i is a vocabulary item of a business rule and c_j is a target class. Equation (2) represents the Bayesian probability and the multinomial model.

$$P(c_j \mid v_i) = \frac{\sum_{i=1}^{|R|} P(c_j \mid v_i)}{|R|} \tag{2}$$

In Equation (3), the Laplacian smoothing process [19] is applied, where $N(v_t, r_i)$ is the total number of times the vocabulary item v_t appears in a rule r_i and where $P(c_j \mid r_i) \in \{0, 1, 2, \ldots . n\}$ depends on the class label of a vocabulary item. Here, the naive Bayesian classifier helps in identifying the class with the top $P(c_j \mid d_i)$ value and that class is assigned to that vocabulary item.

$$P(v_i \mid c_j) = \frac{1 + \sum_{i=1}^{|R|} N(v_t, r_i) P(c_j \mid r_i)}{|V| + \sum_{s=1}^{|V|} \sum_{i=1}^{|R|} N(v_s, r_i) P(c_j \mid r_i)} \tag{3}$$

It is discussed in [14,15] that multiple mixture components M-EM (Multiple-Expectation Maximization) performs better than basic EM with text classification. In our approach, we experimented with both variations of EM approach.

4. Used Approach

This section provides description of the steps of EM-based approach for generating RDF ontology model from rules written in SBVR English. A framework of the used approach is shown in Figure 2. Figure 2 describes that the presented approach first extracts features from a rule and then automatically generates a piece of RDF metadata model. The presented approach also provides a support of querying for the sake of interpretation of the rules. Each phase in the pretend approach is described in the following text.

The used framework is explained in Figure 2 where the proposed framework takes SBVR business rules as inputs. A set of processing steps are applied on the input rules explained in the following text.

4.1. Bag of Words Representation

In generation of an RDF ontology model from rules, the first step is to identify and detect phrases that act as vocabulary concepts. These bag of words may contain many features like people's names and their addresses, or an organisations name and their location. A majority of the vocabulary concepts have multiple words such as "Islamia University", "Microsoft Corporation" and "Credit Card". These features are treated as entities or concepts, then these entities are correctly recognised and tagged corresponding to the predefined taxonomy of rules. In this step, a SBVR rule is the input and a set of heuristics rules are applied to identify key terms to detect requirement phrases.

Each key term in a text corpus is called a token. Frequency (occurrence) of each token in a document is counted and vectorised. Each token frequency is treated as feature. Features of documents are then represented in numerical valued matrix using n-gram model. The strategy used to represent text corpus into matrix form encompasses three steps i.e., Tokenisation, Counting and Normalisation. This three-step strategy as a whole is called Bag of Words or bag of n-gram.

4.2. Sparse Representation

Feature extraction is a process of driving values or features possibly which are informative and non-redundant from the initial set of data [15]. It involves learning and generalisation steps. The initial set of data suspected to be large and redundant. This process is used to transform this large data into a reduced set of features which are expected to contain relevant information. The aim is to reduce the resources (e.g., memory space and time to run an algorithm) required to process data. This process is triggered by training data. Training data contains possible rules. This mechanism is induced to tag new text. Training mechanism is induced to structure phrases that make sense. For training of EM, a data set is carefully selected that contains desired patterns. Here, relations and co-relations also resolved.

A machine cannot process textual data, therefore it should be in numerical form. Sparse representation of numerical matrix diminished those features that rarely occurred in documents. Elimination features having zero frequency may not a good decision [14]. Many documents will use a small subset of corpus. The vocabulary used in small documents will include many unique words individually in the order of 100 to 1000 unique words. Maintaining such a huge vocabulary in memory is a problem. We are interested in those features that are part of process. Key terms representing particular features are frequently used in documents [16,22]. Here, only normalised matrix results are selected of these features for further processing. Sparse representation is formulated on given dictionary. A sparse matrix contains mostly non-zero elements. A sparse representation of matrix is given in Table 1. It contains only 9 non-zero elements. These are those most occurred features in documents.

Table 1. Sparse Matrix of features.

$$\begin{bmatrix} 11 & 22 & 0 & 0 & 0 & 0 & 0 \\ 0 & 33 & 44 & 0 & 0 & 0 & 0 \\ 0 & 0 & 55 & 66 & 77 & 0 & 0 \\ 0 & 0 & 0 & 0 & 88 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 99 \end{bmatrix}$$

4.3. Feature Vectorisation

In the domain of pattern recognition, features are represented using numerical values of objects and these numerical values are typically represented in a feature vector that is an n-dimensional representation of numerical values. The process of creating a feature vector from the extracted features from text is called feature vectorisation [15]. It is a subsequent process of feature extraction. Subsets of relevant features are selected to construct a particular model. An advantage of such feature selection techniques is to use to eliminate redundant and irrelevant features from data.

In our approach, feature vectorisation to generate feature vectors and the vector-based represented generated result of this process is passed to next phase for further processing. Following is the modified code taken from [15].

```
>>> from sklearn.feature extraction.textimport Count Vectorizer
```

We used this method with default parameter values.

```
>>>vectorizer = CountVectorizer(min_df = 1)
>>>vectorizer
CountVectorizer(analyzer = . . . 'word', binary = False, charset = None,
charset_error = None, decode_error = . . . 'strict',
dtype = < . . . 'numpy.int64'>, encoding = . . . 'utf-8', input = . . . 'content',
 lowercase = True, max_df = 1.0, max_features=None, min_df = 1,
ngram_range = (1,1), preprocessor = None, stop_words = None,
strip_accents = None, token_pattern = . . . '(?u)\\b\\w\\w+\\b',
tokenizer = None, vocabulary = None)
```

Example: Tokenisation and counting the word occurrences of a minimalistic corpus of text example:

```
>>>corpus = [
. . . 'This is the first example.',
. . . 'This is the second example.',
. . . 'And the third one.',
. . . 'Is this the first example?',
. . . ]
>>> X = vectorizer.fit_transform(corpus)
>>> X
<4x9 sparse matrix of type '< . . . 'numpy.int64'>' with 19 stored elements in Compressed Sparse . . . format>
```

The default configured parameter tokenised the string by extracting words of at least 2 letters. This function can be called explicitly:

```
>>>analyze = vectorizer.build_analyzer()
>>>analyze("This is a text example to vectorized.") == (
. . . ['this','is','text','example','to','vectorized'])
True
```

Each term individual term assigned a unique identifier preferably an integer that is corresponding to a column in matrix. Columns are retrieved as:

```
>>>vectorizer.get_feature_names() == (
... ['and','document','first','is','one',
... 'second','the','third','this'])
True
>>>X.toarray()
array ([[0, 1, 1, 1, 0, 0, 1, 0, 1],
        [0, 1, 0, 1, 0, 2, 1, 0, 1],
        [1, 0, 0, 0, 1, 0, 1, 1, 0],
        [0, 1, 1, 1, 0, 0, 1, 0, 1]]...)
```

The unseen words in the training corpus will be completely ignored in future calls to the transform method:

```
>>>vectorizer.transform([‘For new example.’]).toarray()
...
array ([[0, 0, 0, 0, 0, 0, 0, 0, 0]]...)
```

In the previous corpus, the first and the last example encoded in the equal vectors because they have exactly the same words. We lost the information in the interrogative form in the third example (is this). To preserve this information we can extract bi-grams of words in addition to the 1-grams (individual words):

```
>>>bigram_vectorizer = CountVectorizer (ngram_range = (1,2),
... token_pattern = r’\b\w+\b’, min_df = 1)
>>>analyze = bigram_vectorizer.build_analyzer()
>>>analyze(‘Bi-grams are cool!’) == (
... [‘bi’,’grams’,’are’,’cool’,’bigrams’,’gramsare’,’are cool’])
True
```

The vocabulary extracted by this vectoriser is hence much bigger and can now resolve ambiguities encoded in local positioning patterns:

```
>>>X_2 = bigram_vectorizer.fit_transform(corpus).toarray()
>>>X_2
. . .
array([[0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0],
[0, 0, 1, 0, 0, 1, 1, 0, 0, 2, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0],
[1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0],
[0, 0, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1]] . . . )
```

In particular the interrogative form “is this” is only present in the last document:

```
>>>feature_index = bigram_vectorizer.vocabulary_.get(‘is this’)
>>>X_2[:,feature_index]
```

4.4. Term Weighting

In pattern recognition, one of the processes is the measuring frequency of extracted features from input text. Such frequency measurements reflect the number of times a word or term is used in a piece of text or document. It is common knowledge that the more times a term is used, the higher the importance that term becomes. Various applications like text summarisation [6] and search engines use the tf-idf algorithm [13] to measure the frequency of terms in a piece of text and weighing the terms accordingly. Term Frequency–Inverse Document Frequency (tf-idf) is a statistical weighting technique used in information retrieval and text mining. It is useful to adjust frequencies of those words that appears most frequently such as stop-words or other words (such as ‘a’, ’an’, ’the’, ’is’ and ’are’.) in documents than other words. It is mostly used in search engine optimisation but it is as good for document features classification.

text_Tf-idf_Transformer class is used to implement normalisation:

```
>>>fromsklearn.feature_extraction.textimportTfidfTransformer
>>>transformer = TfidfTranformer()
>>>transformer
TfidfTransformer(norm = . . . ’12’, smooth_idf = True, sublinear_tf = False, use_idf = True)
```

Following is another processed example:

```
>>>counts=[[3,0,1],
... [2,0,0],
... [3,0,0],
... [4,0,0],
... [3,2,0],
... [3,0,2],
...
>>>tfidf=transformer.fit_transform(counts)
>>>tfidf
<6x3 sparse matrix of type '< ... 'numpy.float64'>'
With 9 stored elements in Compressed sparse ... format>
>>>tfidf.toarray()
array([[ 0.85 ... , 0. ... , 0.52 ... ],
[ 1. ... , 0. ... , 0. ... ],
[ 1. ... , 0. ... , 0. ... ],
[ 1. ... , 0. ... , 0. ... ],
[ 0.55 ... , 0.83 ... , 0. ... ],
[ 0.63 ... , 0. ... , 0.77 ... ]])
```

4.5. Vocabulary Classification to RDF

RDF resources are saved as triple-store which can be accessed by using a query language. A statement is a class in JAVA that represents a single triple. Semantic extension means it allows describing group of resources and relationship among these resources. The class and property structure of RDF Schema is similar to the object-oriented structure of java. RDF schema is written in RDF syntax. In object orientation we define a class as University with an attribute Address of type Location, whereas in RDF Schema we can define it as domain and range. It could be defined as Property, e.g., Address of domain Institute and range of location. The EM algorithm described in Section 3 is used to map SBVR vocabulary concepts to RDF Schema elements such as RDF resource, property and class. The description of resource in terms of properties is also a resource and is identified with a unique identifier. Once a RDF resource is defined it can be used and described. A resource can divide into classes. A group of classes are denoted as a resource, which is also defined as a class in RDF Schema. The detailed working of the EM technique is described in Section 3. A class can easily be extended by its class extension or by creating its instance. There are many techniques to describe meanings of classes and properties in the RDF Schema. Other techniques are concept building and abstract of syntax. We can describe the relationship between the subject resource and the object resource as a concept of RDF property.

4.6. RDF Model Modelling using Jena

RDF is a set of classes with some properties by using RDF knowledge representing language. It provides support to represent basic elements in ontologies description with the help of a RDF resource. RDF resources are saved as triple-store which can be accessed by using a query language. A resource can be anything to be identified. It can have properties about to which data needs to be extracted and stored. It is simply a source of raw data. On web a resource is identified by its Universal Resource Identifier (URI). An example is shown in Figure 3. The visual representation of RDF schema is generated from W3C validator.

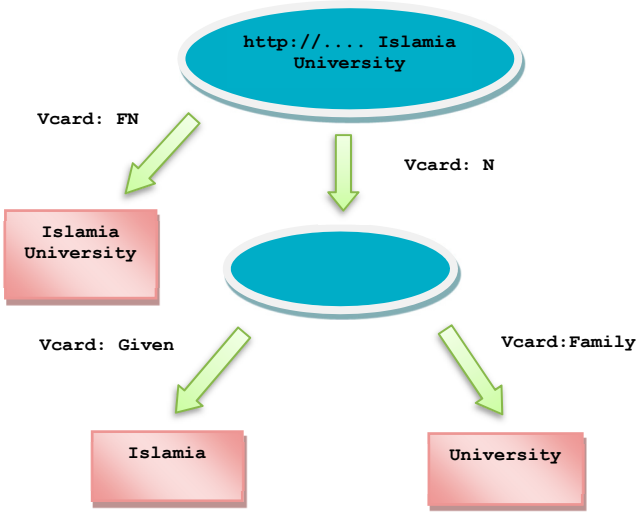

Figure 3. An example of RDF Schema.

The model given in Figure 3 is generated by using Jena library [7] code as:

```
public static void main(String[[] args) {
 // TODO Auto-generated method stub
 // some definitions
 String personURI = "http://somewhere/Islamia University";
 String fistName = "Islamia";
 String SecondName = "University";
 String fullName = firstName + "" + secondName;
 // create the resource
 // and add the properties cascading style
 Resource IslamiaUniversity = model.createResource(personURI)
 .addProperty(VCARD.FN, fullName).addProperty(VCARD.N,
 model.createResource().addProperty(VCARD.Given, firstName)
    . addProperty(VCARD.Family, secondName));
//johsmith.addProperty(p, "hello world", XSDDatatype.XSDstring);
    Model.write(System.out, "Trutle");
```

The produced output model is:

```
<http://somewhere/IslmiaUniversity>
      <http://www.w3.org/2001/vcard-rdf/3.0#FN>
      "Islamia University";
      <http://www.w3.org/2001/vcard-rdf/3.0#Name>
        [ <http://www.w3.org/2001/vcard-rdf/3.0#firstName>
            Islamia
          <http://www.w3.org/2001/vcard-rdf/3.0#secondName>
            University
        ]
```

In the last step, Resource Description Framework (RDF) schema is generated with the help of Jena [7] library. The next phase is the generated Web Ontology Language (OWL) script. A concept or relation in ontology is represented by a relationship in the relational database. A concept is represented by a class in OWL. A relation is mapped to a class in the OWL file [29]. Here are some rules to construct ontology from relational database. Relationships in database are directly mapped to concepts as classes.

The data table is mapped to a concept by representing a table name as a concept, and attributes as properties of concepts.

5. Experiments and Results

This section discusses the experimentation details and the results of the experiments performed to test effectiveness of the presented approach with the help of real life examples. A few examples of business rules used in the experiments are taken from the Cafeteria Ordering System (COS) [30], given below. The SBVR representation of these COS rules is available in [31].

It is obligatory that the system *shall let*, each Patron who *is logged* into the Cafeteria Ordering System, *place* at least one order for at least one or more meals.
It is obligatory that the system *shall confirm* that the 'Patron' *is registered* for payroll deduction to *place* at least one order.
If the Patron *is* not *registered* for payroll deduction, It is obligatory that the system *shall give* the Patron options to *register* and *continue placing* at least one order, to *place* at least one order for *pickup* in the cafeteria, or to *exit* from the COS.
It is obligatoy the system *shall prompt* the Patron for the *meal date*.
If the *meal date is* the current date and the current time *is* after the order *cutoff* time, it is obligatory that the system *shall inform* the Patron that it's too late to *place* at least one order for today.
It is possibility that the Patron *may change* the *meal date* or *cancel* the order.
It is obligatory the Patron *shall specify* whether the order *is* to *be picked* or *delivered*.
If the order *is* to *be delivered* and there still *are* available *delivery times* for the *meal date*, it is obligatory that the Patron *shall provide* at least one valid *delivery location*.

A set of experiments were performed to test the used approach. In this chapter, three text documents selected from a diverse domain are selected and processed with the used approach. The output of all three different inputs is analysed and the results are disseminated at the end of the chapter. Table 2 shows the term Frequencies of three examples, 1, 2 and 3:

Table 2. Term Frequencies of three examples, 1, 2 and 3.

Terms	d1	d2	d3
A	6	17	3
About	0	0	1
Accept	1	0	0
Achievement	0	0	1
Administrator	0	0	1
Advance	1	0	0
Agrees	0	1	0
Alike	0	0	1
All	0	0	4
Allowed	1	0	0
Also	2	0	0

Following is the vectorisation of the problem of Example 1, 2 and 3:

[6, 0, 1, 0, 0, 1, 0, 0, 0, 1, 2, 0, 0, 1, 1, 0, 0, 0, 4, 0, 1, 0, 2, 0, 0, 0, 5, 2, 0, 2, 2, 0, 1, 10, 0, 1, 0, 1, 0, 1, 11, 0, 0, 0, 0, 0, 0, 1, 0, 00, 0, 5, 0, 2, 0, 0, 1, 0, 0, 2, 0, 0, 0, 1, 1, 3, 0, 0, 0, 0, 0, 1, 3, 1, 0, 0, 0, 2, 3, 1, 0, 1, 0, 0, 0, 4, 1, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 3, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 4, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 4, 1, 3, 1, 1, 0, 2, 0, 1, 0, 0, 2, 2, 0,

0, 0, 0, 0, 0, 0, 1, 1, 0, 6, 2, 1, 0, 3, 0, 0, 0, 3, 0, 0, 0, 0, 3, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 2, 3, 14, 0, 2, 2, 6, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0]

[17, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 4, 0, 5, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0,0, 5, 0, 3, 0, 0, 9, 0, 0, 0, 3, 2, 0, 2, 0, 0, 4, 8, 1, 1, 0, 0, 0, 0, 4, 6, 0, 0, 0, 0, 0, 0, 0, 4, 0, 2, 1, 0, 9, 0, 0, 0, 1, 0, 0, 1, 0, 0, 3, 0,0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 8, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 26, 13, 3, 1, 0, 4, 0, 0, 0, 0, 0, 1, 3, 0, 0, 1, 0, 0, 2, 2, 0, 0, 4, 1, 0, 5, 3,0, 1, 0, 0, 0, 6, 14, 0, 0, 3, 1, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0,0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 1, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0, 13,20, 0, 0, 0, 5, 0, 0, 3, 1, 1, 5, 1, 0, 0, 0, 0, 0]

[3, 1, 0, 1, 1, 0, 0, 1, 4, 0, 0, 0, 2, 10, 0, 3, 0, 2, 4, 1, 0, 1, 0, 1, 2,1, 0, 0, 0, 2, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 2, 1, 0, 1, 0, 0, 0, 0, 0, 3, 1, 0, 1, 0, 7, 0,1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 9, 2, 1, 1, 1, 1, 1, 1, 1, 3, 1,0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 2, 0, 0, 1, 2, 0, 5, 0, 0, 0,1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 10, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1,1, 2, 0, 1, 4, 1, 0, 0, 0, 0, 0, 16, 0, 2, 0, 0, 1, 4, 4, 4, 1, 15, 2, 1, 1,8, 3, 1, 0, 1, 1, 0, 0, 2, 0, 0, 0, 0, 1, 7, 5, 4]

Following Table 3 is the Term Weighting of the problem of Example 1, 2, 3:

Table 3. Term weighting of the problem of Example 1, 2, 3.

Terms	**Occ(ti,di)/occ(tmam,di)*log N/n(t)**		
A	0	0	0
Accept	0.079520209	0	0
Advance	0.079520209	0	0
Also	0.029348543	0	0.058697086
And	0.079520209	0	0
Any	0	0	0.159040418
Are	0.318080836	0	0
Ask	0.079520209	0	0
Assigned	0.079520209	0	0
At	0	0	0
Available	0.159040418	0	0
Be	0	0.02934854	0.058697086
Been	0.029348543	0.02934854	0
Booking	0	0	0.159040418
Branch	0	0	0
By	0.079520209	0	0
Can	0.079520209	0	0
. . . . .	. . . . .	. . . . .	. . . . .

Following is the vectorisation after term weighting of the problem of Example 1, 2 and 3:

[0.000, 0.000, 0.034, 0.000, 0.000, 0.034, 0.000, 0.000, 0.000, 0.034, 0.068, 0.000, 0.000, 0.000, 0.034, 0.000, 0.000, 0.000, 0.050, 0.000, 0.034, 0.000, 0.068, 0.000, 0.000, 0.000, ...]

[0.000, 0.000, 0.000, 0.000, 0.000, 0.000, 0.018, 0.000, 0.000, 0.000, 0.000, 0.073, 0.000, 0.000, 0.000, 0.000, 0.055, 0.000, 0.000, 0.000, 0.000, 0.000, 0.000, 0.000, 0.000, 0.000, ...]

[0.000, 0.030, 0.000, 0.030, 0.030, 0.000, 0.000, 0.030, 0.119, 0.000, 0.000, 0.000, 0.060, 0.000, 0.000, 0.089, 0.000, 0.060, 0.044, 0.030, 0.000, 0.030, 0.000, 0.030, 0.060, 0.030 . . . . .]

After normalisation some terms are eliminated and we have the following terms lefts. An example of RDF schema of Reservation class model is shown in Figure 4.

The used approach for the interpretation of rules generates an ontology model to get the context of a rule, and this context can be used to interpret a particular rule. The used approach first identifies and extracts the features or properties of a rule and then on the basis of the extracted features. The identified features are used to design a decision tree. For this purpose, Fit binary classification decision tree for multiclass classification available in MATLAB is used. This decision tree is further mapped to a

RDF representation. The presented approach automatically maps a decision tree to a RDF equivalent representation using Jena library [7]. The used tool is shown in Figure 5.

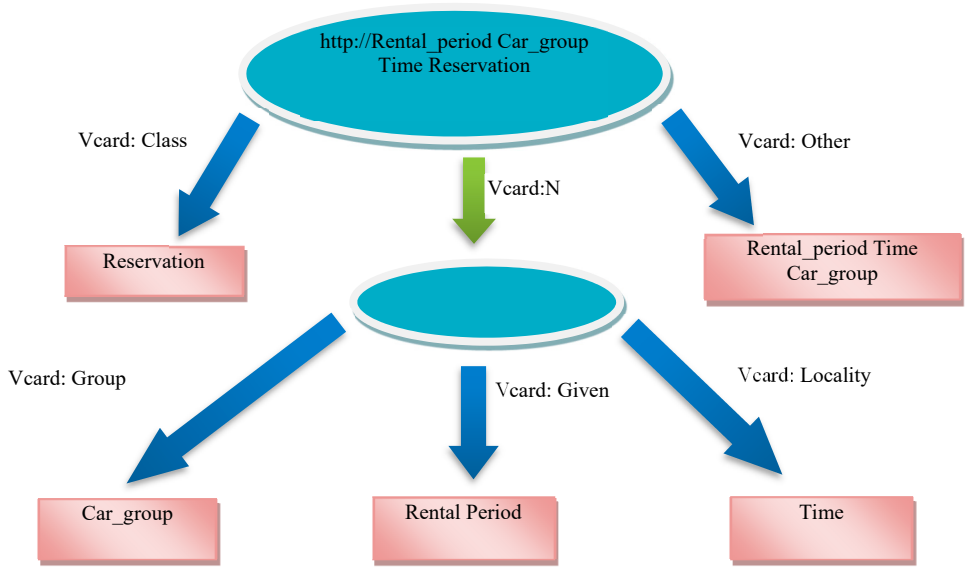

Figure 4. Output of the experiment.

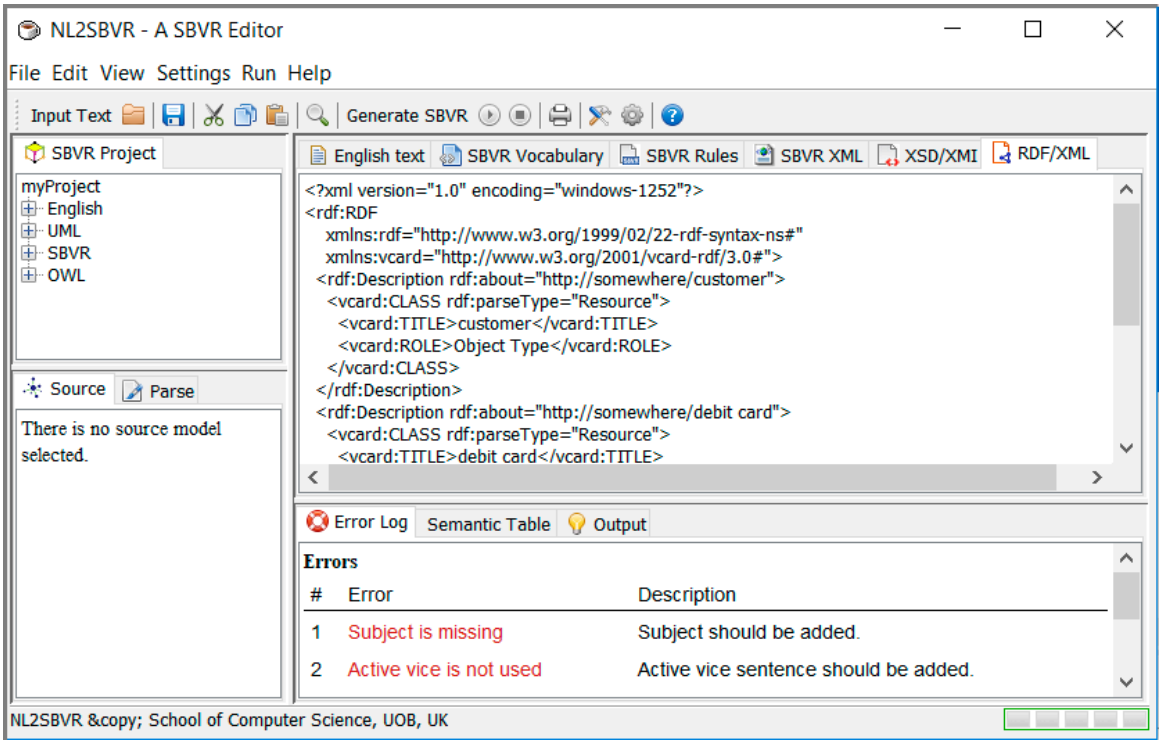

Figure 5. Eclipse Implementation of the tool.

The used approach presents a method to that not only supports automated generation of RDF metadata model of rules but also provide support for querying for the sake of interpretation of the rules. The presented approach provides support for consistency checking of a set of business rules, and also makes the process of interpretation of rules simple. The precision and recall of this example is shown in Table 4.

Besides this case study, some other case studies (Table 4) were taken from legal documents of banks and universities. All these case studies were unseen. The solved case studies were of different

lengths. The largest case study was composed of 209 words and 12 sentences. The smallest case study was composed of 69 words and 5 sentences. Calculated recall, and precision values of the solved case studies are shown in Table 5 and the results are visualised in Figure 6.

Table 4. Overall Results of case study of framework used for semantic annotation.

Data Type	Total Terms	Correct Terms	Missed Terms	Incorrect Terms	Recall	Precision
Concept Name Classification	21	18	1	2	85.71%	90.00%
Semantic Annotation	6	5	1	1	83.33%	83.33%

Table 5. Evaluation results of experiments.

Case Study	Total Terms	Correct Terms	Missed Terms	Incorrect Terms	Recall	Precision
C 1	28	24	3	1	85.71%	88.88%
	11	8	1	2	72.72%	80.00%
C 2	33	26	3	4	78.78%	86.66%
	13	11	0	2	84.61%	84.61%
C 3	19	15	1	3	78.94%	83.33%
	7	6	0	1	85.71%	85.71%
C 4	21	18	1	2	85.71%	90.00%
	6	5	1	1	83.33%	83.33%
				Average	**81.93%**	**85.32%**

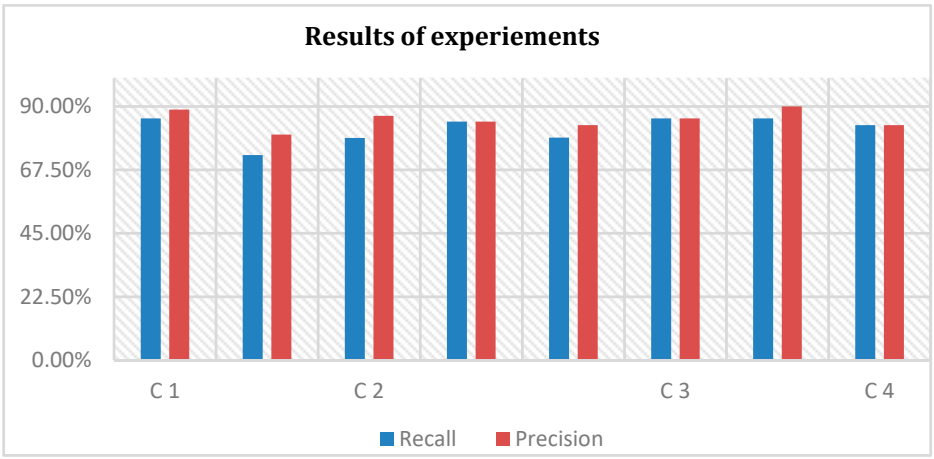

Figure 6. The results of the experiments in recall and precision.

The average overall Recall value (81.93%) and Precision value (85.32%) is encouraging for initial experiments. We cannot compare our results to any other tool as no other tool is available that can classify and annotate entities in legal text. However, we can note that other language processing technologies, such as information extraction systems, and machine translation systems, have found commercial applications with precision and recall shown in Figure 6. The results of previous studies in terms of accuracy are not available, which is why we could not compare the results. However, the results of the experiments shown in Figure 6 show the effectiveness of the approach and the tool as well. The paper relates to the big data as it takes data from Hadoop, NOSQL, etc. The size of experiment data sets is small but large case studies can also be executed with this approach and tool.

6. Conclusions

This paper addresses the problem of automated interpretation of rules expressed in a controlled natural language such as SBVR. It has addressed how to cop with the challenge of semantic inconsistency in rules. The presented system generates an ontology model for the interpretation of a set of rules. Once the model is generated it can be validated using web services like W3C validator. Additionally, the semantics of a rule can also be queried using SPARQL language. The presented system is a domain-specific solution and has all the details regarding a system specific to the domain. For the sake of generating an ontology model, an algorithm is designed. The presented approach is also implemented in Java as a proof of concept. This paper presents a robust method that not only supports semantic analysis of rules but also generates RDF metadata model of rules and provide support of querying for the sake of semantic interpretation of the rules. The results of the experiments manifest that consistency checking of a set of big data rules is possible through automated tools. The prototype too is an Eclipse plug-in. A set of experiments were performed to test the used approach. The results of the experiments discussed in this paper highlight that the presented approach can help in improving the process of interpretation of rules.

Author Contributions: R.S. contributed in design, implementation, and experimentation of this research and writing this manuscript. I.S.B. supervised this work and also edited this manuscript. R.K. contributed in experiments and evaluation of this research.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Tadikonda, V.; Rosca, D. Informed and Timely Business Decisions-A Data-driven Approach. In Proceedings of the SEKE, San Francisco, CA, USA, 1–3 July 2016; pp. 24–30.
2. OMG. Semantics of Business Vocabulary and Business Rules (SBVR), Version 1.4. May 2017. Available online: https://www.omg.org/spec/SBVR (accessed on 11 July 2018).
3. Bajwa, I.S.; Lee, M.G.; Bordbar, B. SBVR Business Rules Generation from Natural Language Specification. In Proceedings of the AAAI Spring Symposium: AI for Business Agility, Palo Alto, CA, USA, 23 March 2011; pp. 2–8.
4. Grover, V.; Chiang, R.H.; Liang, T.P.; Zhang, D. Creating Strategic Value from Big Data Analytics: A Research Framework. *J. Manag. Inf. Syst.* **2018**, *35*, 388–423. [CrossRef]
5. Carroll, J.; Herman, I.; Patel-Schneider, P.F. *OWL 2 Web Ontology Language RDF-Based Semantics*; W3C Recommendation; Available online: https://www.w3.org/TR/owl2-rdf-based-semantics/ (accessed on 13 August 2018).
6. Čebirić, Š.; Goasdoué, F.; Manolescu, I. Query-oriented summarization of RDF graphs. *Proc. VLDB Endow.* **2015**, *8*, 2012–2015.
7. Ceravolo, P.; Fugazza, C.; Leida, M. Modeling semantics of business rules. In Proceedings of the Inaugural IEEE-IES Digital EcoSystems and Technologies Conference, 2007 (DEST'07), Cairns, Australia, 21–23 February 2007; pp. 171–176.
8. Choksi, A.T.; Jinwala, D.C. A Novel Way to Relate Ontology Classes. *Sci. World J.* **2015**, *1*, 724196. [CrossRef] [PubMed]
9. Corcho, O.; Fernández-López, M.; Gómez-Pérez, A. Methodologies, tools and languages for building ontologies. Where is their meeting point? *Data Knowl. Eng.* **2003**, *46*, 41–64. [CrossRef]
10. Dai, W.; Xue, G.R.; Yang, Q.; Yu, Y. Transferring naive bayes classifiers for text classification. In Proceedings of the National Conference on Artificial Intelligence, Vancouver, BC, Canada, 22–26 July 2007; AAAI Press: Menlo Park, CA, USA; Cambridge, MA, USA; London, UK, 1999; Volume 22, p. 540.
11. Ferreira, R.; de Souza Cabral, L.; Lins, R.D.; e Silva, G.P.; Freitas, F.; Cavalcanti, G.D.; Lima, R.; Simske, S.J.; Favaro, L. Assessing sentence scoring techniques for extractive text summarization. *Expert Syst. Appl.* **2013**, *40*, 5755–5764. [CrossRef]

12. Jena.apache.org. Apache Jena—Jena Ontology API. N.p. 2016. Available online: https://jena.apache.org/documentation/ontology/ (accessed on 1 March 2018).
13. Krieger, H.U. A Detailed Comparison of Seven Approaches for the Annotation of Time-Dependent Factual Knowledge in RDF and OWL. In Proceedings of the 10th Joint ISO-ACL SIGSEM Workshop on Interoperable Semantic Annotation, Reykjavik, Iceland, 26 May 2014; p. 1.
14. Liang, P.; Klein, D. Online EM for unsupervised models. In Proceedings of the Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Boulder, CO, USA, 31 May–5 June 2009; pp. 611–619.
15. Lin, Y.S.; Jiang, J.Y.; Lee, S.J. A similarity measure for text classification and clustering. *Knowl. Data Eng.* **2014**, *26*, 1575–1590. [CrossRef]
16. Lin, H.T.; Bui, N.; Honavar, V. Learning classifiers from remote RDF data stores augmented with RDFS subclass hierarchies. In Proceedings of the 2015 IEEE International Conference on Big Data (Big Data), Santa Clara, CA, USA, 29 October–1 November 2015; pp. 1807–1813.
17. Nigam, K.; McCallum, A.K.; Thrun, S.; Mitchell, T. Text classification from labelled and unlabelled documents using EM. *Mach. Learn.* **2000**, *39*, 103–134. [CrossRef]
18. Nigam, K.; McCallum, A.; Mitchell, T. Semi-Supervised Text *Classification Using EM*. Semi-Supervised Learning. MIT Press: Boston. 2006, pp. 33–56. Available online: http://parnec.nuaa.edu.cn/seminar/2012_Spring/20120323/%E8%92%8B%E8%90%8D/Semi-Supervised%20Text%20Classification%20Using%20EM.pdf (accessed on 19 February 2019).
19. Paik, J.H. A novel TF-IDF weighting scheme for effective ranking. In Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval, Dublin, Ireland, 28 July–1 August 2013; pp. 343–352.
20. Saggion, H.; Funk, A.; Maynard, D.; Bontcheva, K. Ontology-based information extraction for business intelligence. In Proceedings of the 6th International Conference on Semantic Web, Busan, Korea, 11–15 November 2007; pp. 843–856.
21. Scikit-learn.org. 4.1. Feature Extraction Scikit-Learn 0.15.2 Documentation. 2015. Available online: http://scikit-learn.org/stable/modules/feature_extraction.html#feature-extraction (accessed on 15 February 2015).
22. Baudel, T.; Frank, V. Rule correlation to rules input attributes according to disparate distribution analysis. U.S. Patent No. 88,25,588, 2 September 2014.
23. Guissé, A.; Lévy, F.; Nazarenko, A. From regulatory texts to BRMS: How to guide the acquisition of business rules? In Proceedings of the International Workshop on Rules and Rule Markup Languages for the Semantic Web, Montpellier, France, 27–29 August 2012; Springer: Berlin/Heidelberg, Germany, 2012; pp. 77–91.
24. Sitharamulu, V.; Babu, B.R. A Novel Proposal for Bridging Gap between RDB-RDF Semantic Web using Bidirectional Approach. *Int. J. Appl. Eng. Res.* **2016**, *11*, 4456–4460.
25. Paulheim, H.; Plendl, R.; Probst, F.; Oberle, D. Mapping pragmatic class models to reference ontologies. In Proceedings of the 2011 IEEE 27th International Conference on Data Engineering Workshops (ICDEW), Hannover, Germany, 11–16 April 2011; pp. 200–205.
26. Lu, R.; Sadiq, S. A survey of comparative business process modeling approaches. In Proceedings of the International Conference on Business Information Systems, Poznan, Poland, 25–27 April 2007; Springer: Berlin/Heidelberg, Germany, 2007; pp. 82–94.
27. Cimiano, P.; Haase, P.; Herold, M.; Mantel, M.; Buitelaar, P. Lexonto: A model for ontology lexicons for ontology-based NLP. In Proceedings of the OntoLex07 Workshop Held in Conjunction with ISWC'07, Busan, Korea, 11–15 November 2007.
28. W3.org. RDF Schema 1.1. 2014. Available online: http://www.w3.org/TR/2014/PER-rdf-schema-20140109/ (accessed on 30 January 2015).
29. Alani, H.; Kim, S.; Millard, D.E.; Weal, M.J.; Hall, W.; Lewis, P.H.; Shadbolt, N.R. Automatic ontology-based knowledge extraction from web documents. *Intell. Syst. IEEE* **2003**, *18*, 14–21. [CrossRef]
30. SRS for Cafeteria Ordering System—Seidenberg School of... (n.d.). Available online: http://csis.pace.edu/~{}marchese/SE616_New/Samples/SE616_SRS.doc (accessed on 13 August 2018).
31. Umber, A.; Bajwa, I.S. A Step towards Ambiguity less Natural Language Software Requirements Specifications. *Int. J. Web Appl.* **2012**, *4*, 12–21.