Automatic Identification of Ontology Versions Using Machine Learning Techniques

Carlo Allocca

Knowledge Media Institute (KMi), The Open University, Walton Hall, Milton Keynes MK7 6AA, United Kingdom
c.allocca@open.ac.uk

Abstract. When different versions of an ontology are published online, the links between them are often lost as the standard mechanisms (such as owl:versionInfo and owl:priorVersion) to expose these links are rarely used. This generates issues in scenarios where people or applications are required to make use of large scale, heterogenous ontology collections, implicitly containing multiple versions of ontologies. In this paper, we propose a method to detect automatically versioning links between ontologies which are available online through a Semantic Web search engine. Our approach is based on two main steps. The first step selects candidate pairs of ontologies by using versioning information expressed in their identifiers. In the second step, these candidate pairs are characterized through a set of features, including similarity measures, and classified by using Machine Learning Techniques, to distinguish the pairs that represent versions from the ones that do not. We discuss the features used, the methodology employed to train the classifiers and the precision obtained when applying this approach on the collection of ontologies of the Watson Semantic Web search engine.

1 Introduction

Ontologies evolve according to the modifications carried out in response to changes in the represented domain or its conceptualization [20]. Such an evolution of an ontology, O_i, produces a finite collection of documents $\{O_{i1}, O_{i2},...,O_{im}\}$, where each O_{ij} represents a particular version of O_i. Although, mechanisms exist to keep track of the links between different versions (e.g. owl:versionInfo, owl:priorVersion), ontologies that are exposed on the Web rarely make use of such ways to link versions [3] (e.g. `http://lsdis.cs.uga.edu/projects/semdis/sweto/testbed_v1_3.owl` and `http://lsdis.cs.uga.edu/projects/semdis/sweto/testbed_v1_4.owl`).

Semantic Web search engine (SWSEs) systems, such as Watson[1], Swoogle[2] or Sindice[3] collect and index ontologies in order to facilitate their reuse [1]. However, the lack of explicit links between versions of ontologies hampers the various scenarios where users or applications make use of such large scale, heterogenous

[1] `http://watson.kmi.open.ac.uk`
[2] `http://swoogle.umbc.edu/`
[3] `http://sindice.com/`

G. Antoniou et al. (Eds.): ESWC 2011, Part I, LNCS 6643, pp. 352–366, 2011.

collection of ontologies, implicitly containing multiple ontology versions [7]. In particular, users might require the most up-to-date information or need to compare different versions of an ontology, to be able to choose the most appropriate one.

In this paper, we investigate an approach *to automatically detect/discover implicit versioning links in large ontology collections, in order to make them explicit*. In contrast with the approaches described in [15,14,19,12,21,23] (see Section 5), we propose a version detection mechanism for ontologies where versioning metadata are not available and the ontologies are collected and indexed by a SWSE system. To this end, two important issues need to be addressed: *(1) how to select pairs of ontologies as candidate versions*; *(2) how to decide whether the candidate pairs are versions or not*. To tackle (1), we adopt a solution based on identifying and extracting versioning information codified in the ontology URIs. To deal with (2), we propose a set of features which characterize ontology versions and we apply machine learning techniques to classify the candidate ontology pairs as *PrevVersion* and *NotPrevVersion*. The method has been evaluated using the Watson collection of OWL ontologies, which is a dataset of 7000 ontologies. The evaluation shows that the features we considered provide an accurate characterization of ontology versions, leading to high precision when used in three different types of classifiers: *Naive Bayesian*, *Support Vector Machine* and *Decision Tree*. In particular, *Support Vector Machine*, the best and most stable classifier, achieved a precision of 87% in this task.

In the next section, we detail how we analyze and extract versioning information patterns from the ontology URIs to select candidate ontology pairs. In Section 3, we describe the set of features which characterize ontology version pairs and the use of machine learning techniques to classify them. In Section 4, we discuss our evaluation using the three different classifier models: *Naive Bayesian* (NB), *Support Vector Machine* (SVM) and *Decision Tree* (DT). In Section 5, we discuss the relevant related work. Finally, in Section 6, we summarize the key contribution of this work and outline our plans for future work.

2 Finding Candidate Versioning Links Based on Ontology URIs

Detecting ontology version links raises the issue of *selecting pairs of ontologies to check whether they are versions or not*. In the context of SWSEs where there are thousands of collected ontologies, it would be unrealistic to apply complex comparisons on all the possible pairs of ontologies. Therefore, we need to define a straightforward method to discover candidate pairs of ontologies for such a comparison. In an initial work [3], we manually analyzed a representative sample (nearly 1000) of URIs from the ontology repository of the Watson SWSE. We noticed that many of them contain numerical information concerning the version of the ontology (following versioning patterns). Specifically, we identified three classes of such patterns: (1) where the versioning information is encoded in a single number; (2) where the versioning information is expressed by two numbers, which are either the month and year of a date or the two numbers of a major and

minor release; and (3) where the versioning information is expressed by three numbers, which always correspond to a complete date. Based on these patterns, we designed six rules, specifically two rules for each pattern, to use to compare URIs which reflect the main characteristics of the classes:

R1 corresponds to the most straightforward case where there is only one numerical difference between two URIs. For example:

```
http://www.vistology.com/ont/tests/student1.owl;
http://www.vistology.com/ont/tests/student2.owl;
```

However, there can be many variants of such a pattern. In the following example, a time-stamp is used to mark a particular version of the ontology:

```
http://160.45.117.10/semweb/webrdf/#generate_time
   stamp_1176978024.owl
http://160.45.117.10/semweb/webrdf/#generate_time
   stamp_1178119183.owl
```

R2 corresponds to those cases where two numbers differ from one URI to the other. The version information corresponds to a version number, in which case the number on the left is more significant. For example:

```
http://lsdis.cs.uga.edu/projects/semdis/sweto/test
   bed_v1_0.owl
http://lsdis.cs.uga.edu/projects/semdis/sweto/test
   bed_v1_1.owl
```

R3 **R4** correspond to those cases where two numbers differ from one URI to the other. The version information corresponds to a date including the year and month only, in which case, the year is more significant. For example:

```
http://loki.cae.drexel.edu/~wbs/ontology/2003/02/
   iso-metadata
http://loki.cae.drexel.edu/~wbs/ontology/2003/10/
   iso-metadata
http://loki.cae.drexel.edu/~wbs/ontology/2004/01/
   iso-metadata
http://loki.cae.drexel.edu/~wbs/ontology/2004/04/
   iso-metadata
```

R5 **R6** correspond to the cases where three numerical differences exist between the considered URIs. In our dataset, we have not encountered examples other than the representation of dates using 3 numbers. Therefore, we only define the rules corresponding to dates, either in big endian or in little endian form. For example:

```
http://ontobroker.semanticweb.org/ontologies/ka2-
   onto-2000-11-07.daml
http://ontobroker.semanticweb.org/ontologies/ka2-
   onto-2001-03-04.daml
```

Moreover, our analysis also showed that these six rules cover nearly 90% of the versioning information codified in the URIs (we counted the number of ontology URIs detected for each rule out of 7000). The other 10% concern either versioning information expressed through the combination of words and numbers, e.g. `/january2007/` and `/october2006/` or patterns based on more than four numbers. Based on the three classes of versioning patterns, we devised a polynomial algorithm which compares URIs to extract the numerical differences and use the six rules to generate candidate versioning relations between ontologies [3]. Running it over a dataset of 7000 OWL/RDF-ontologies, the algorithm detected 24644 pairs of candidate ontology versions, in just a few minutes (between 3-4 minutes) on a MacBookPro laptop, Intel Core 2 Duo - 2.6 GHz. A subset of them (531 out of 24644) were manually analyzed, looking in particular at their content. The analysis showed that only about half of these pairs indicated versioning relations. In addition, we also noticed that when two ontologies were linked by versioning relations, it was possible to establish an overlapping between their vocabularies and sets of axioms. In the next Section we show how we exploit such features in a machine learning approach to automatically decide whether ontology pairs are in a versioning relation.

3 Applying Machine Learning Classifiers

Once we obtain an initial set of candidate ontology pairs which potentially represent versions, the issue is then to *automatically differentiate the pairs which are versions from the ones which are not.* To this purpose, we first identified a set of features which characterize ontology versions and then we applied machine learning techniques to classify the pairs in two classes: *PrevVersion* and *NotPrevVersion.*

3.1 Attributes

Here we describe in detail the set of ontology features that are exploited by Machine Learning techniques to identify versioning relations.

Length of Chain. The *Length of Chain* attribute represents the length of the sequence of successive ontologies that a pair is part of. We define and compute such *chains* of ontologies as the connected paths in the graph formed by the links detected by the mechanism described in Section 2. More formally,

Definition 1 (Ontology Chain). *Given a collection of pairs of ontologies* $P=\{(O_i, O_j)\}$*, an ontology chain is defined as the longest sequence of ontologies,* $O_1, O_2, ..., O_{n-1}, O_n$ *such that* $(O_k, O_{k+1}) \in P$ *and* O_k *represents the candidate previous version of* O_{k+1}*. Here, the collection* P *corresponds to the set of candidate pairs selected according to the approach described in Section 2. It is important to notice that, according to this approach, a given candidate pair can only be part of one chain of ontologies.*

Based on Definition 1, we computed 1365 ontology chains out of 24644 ontology pairs detected with the previous step. Manually analyzing a small number of these chains (39 out of 1365) we noticed that shorter chains (from 4 to 6) are more likely to actually represent sequences of ontology versions. The main reason for this is that many sequences come from automatically generated ontologies and URIs in which numbers are not representing version information but a "record number". This typically happens when database tuples are exported in RDF or OWL. In other terms, to each record from the database is assigned a number expressed in the URI when translated it into ontology language (e.g ...*student-record-1*, *student-record-2*).

Similarity Measures. Similarity measures are considered to be relevant characteristics in studying the versioning problem [15], as a reasonable intuition is that ontology versions should be similar to a certain extent.

Ontology similarity has been described as a measure to assess how close two ontologies are [8]. Various ways to compute the similarity between two ontologies have been studied to be relevant in different application contexts [8,10,24]. However, no particular similarity function has been identified as being the most effective one for detecting versions of ontologies. Following the basic operations in ontology evolution of adding, deleting and modifying ontology axioms or just changing the ontology terminology [12,21], we focus here on two different similarity functions: 1) *lexicographicSimilarity* and 2) *syntacticSimilarity*, which are defined as follows

$$\textbf{\textit{lexicographicSimilarity}}(O_i,\ O_j) = \frac{|Voc(O_j) \bigcap Voc(O_j)|}{max(|Voc(O_i)|, |Voc(O_j)|)}$$

where $Voc(O_k)$ is the normalized[4] vocabulary VOC of the ontology O_k, k=i,j.

$$\textbf{\textit{syntacticSimilarity}}(O_i,O_j) = \frac{|Axioms(O_i) \bigcap Axioms(O_j)|}{max(|Axioms(O_i)|, |Axioms(O_j)|)}$$

where $Axioms(O_k)$ is the set of normalized[5] axioms of the ontology O_k, k=i,j.

Both these measures correspond to the ratio between the size of the overlap and the size of the largest vocabulary and set of axioms of the ontologies, respectively. We use the size of the larger ontology rather than the smaller because we believe that it gives intuitively better results in those cases where we need to compare ontologies of very different size.

Looking at Fig. 1, we notice that there is a non-trivial relationship between similarity and the presence of a versioning link in a given pair of ontologies. In other words, no threshold naturally emerges from these values that can help selecting pairs of ontologies not representing versions. Indeed, both very dissimilar and very similar ontologies appear unlikely to represent ontology versions. One

[4] We normalize the elements of the vocabulary of an ontology transforming upper case letters to lower case and removing separators such as underscore and hyphen.

[5] Normalized axioms are axioms where the elements of the vocabulary have been normalized.

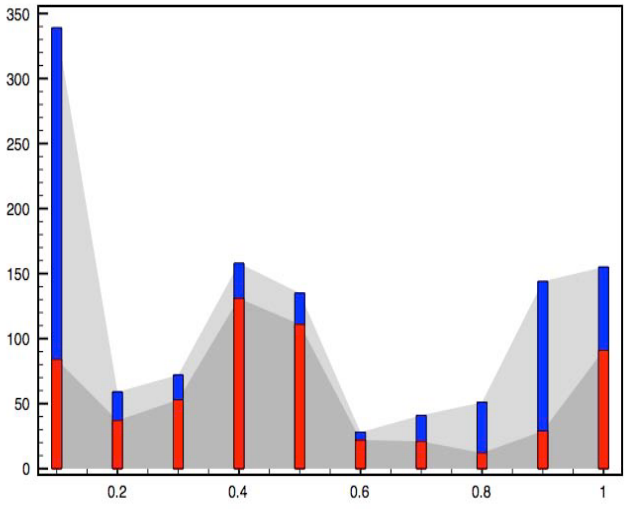

Fig. 1. Distribution of syntactic similarity in manually evaluated ontology pairs. Red represents pairs that are recognized as ontology versions, and blue represents the ones that are not.

of the goals of using machine learning classifiers here is therefore to find a mechanism to exploit this non-trivial relationship between similarity and versioning, combined with other characteristics of the ontology pairs.

Ontology Versioning Patterns. Based on the analysis of ontology URIs, we observed that some of the patterns described in Section 2 give better results than others in terms of both precision and recall. For example, R2, which represents patterns with two numbers (expressing a version number such as v1.1, v1.2), gives results which, while numerous, appear more accurate than R1, which considers only one number to express versioning information. Using such information, the classifier can compute different levels of confidence for each rule, and learn that some patterns provide more accurate information than others.

3.2 Methodology

A classifier model is an arbitrarily complex mapping from all-but-one dataset attributes to a class attribute C [5]. The specific form and creation of this mapping, or model, differs from classifier to classifier. In our case, the input attributes correspond to the three features described above and the class attribute, C, takes values in {*PrevVersion*, *NotPrevVersion*}. *PrevVersion* represents the class of ontology pairs (O_i, O_j) where O_i is a direct or indirect previous version of O_j, while *NotPrevVersion* represents the class of ontology pairs, (O_i, O_j), where O_i is not a previous version of O_j, directly or indirectly. In our experiments, we compared the performances of three different types of classifiers: *Naive Bayesian (NB), Support Vector Machine (SVM) and Decision Tree (DT)*, and analyzed

to what extent the set of the proposed attributes supports the discovery of different versions of the same ontology. For our experiment, we use the Machine Learning Toolkit Weka [5] and the performance measures obtained are based on Weka's model implementation in its default configuration. The reader interested in finding out more details about the classifiers is referred to [11,17] for NB, [30] for SVM and [18] for DT.

A systematic approach to applying machine learning classifiers follows four phases: 1) *Training*; 2) *Testing*; 3) *Validation*; and 4) *Classification* [9,5]. Each phase receives an input set of *training*, *testing*, *validation* and *classification* tuples of feature values, respectively. In our case, an arbitrary tuple characterizing a candidate pair of ontologies (O_i, O_j) has the following structure: [*CL*, *Rn*, *VocSim*, *SynSim*, *C*] where *CL* is the length of the chain that includes (O_i, O_j); *Rn* is the number of the rule (R1-R6) used to select (O_i, O_j); *VocSim* is the *lexicographicSimilarity* measure value between O_i and O_j, *SynSim* is the *syntacticSimilarity* measure value between O_i and O_j and the value of *C* is either *PrevVersion* or *NotPrevVersion*.

Given a classifier T, the *Training* phase is responsible for building the classifier model M_T. To do this, T receives the input set of *training* tuples, manually classified. The *Testing* phase is responsible for checking the accuracy of the model M_T. Here *T* receives the input set of *testing* tuples (disjoint from the *training* set), manually classified, and assigns a value to the class attribute C of each tuple. The accuracy of the model is measured by comparing the value of the class attribute C of the *testing* tuple set to the value assigned by the classifier T. The accuracy rate is given by the percentage of testing set samples correctly classified by the model. The *Validation* phase is in charge of evaluating the stability of the classifier. The classifier *T* receives the input set of manually classified *validation* tuples (disjoint from the *training* and *testing* sets). Running the model M_T over the *validation* set, the classifier computes its accuracy rate and performance one more time. Comparing these values with the ones obtained from the *Testing* phase, the stability of the classifier T is calculated. Finally, in the *Classification* phase T receives the input set of *classification* tuples and assigns one of two values {*PrevVersion*, *NotPrevVersion*} to the class attribute C.

4 Evaluation

In this Section, we detail our evaluation of the approach described above, using the collection of OWL ontologies from the Watson SWSE.

4.1 Experimental Set-Up: Generating Datasets

Fig. 2 shows the main steps to build datasets for the classifiers. In particular, (Fig. 2 (a)) represents the Watson collection of 7000 ontologies. We first ran the versioning patterns based mechanism described in Section 2, which identified a set of 24644 candidate pairs of ontologies (Fig. 2 (b)). Then we manually analyzed a subset of these candidate pairs, producing 591 pairs as *PrevVersion*

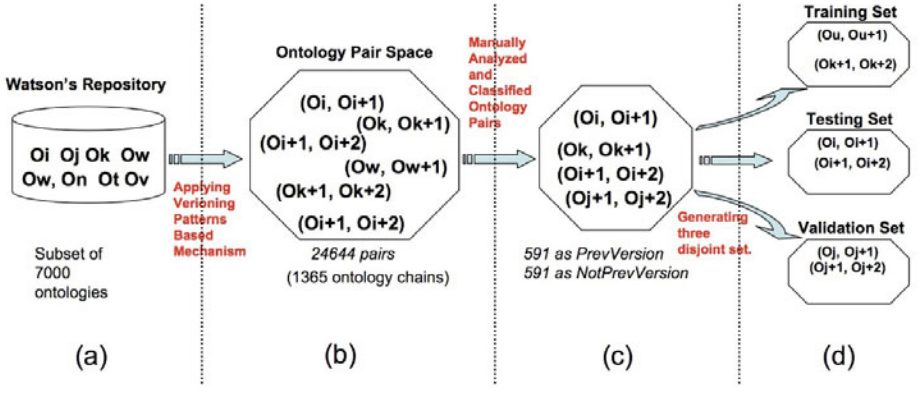

Fig. 2. Flow of building the datasets

and 591 pairs as *NotPrevVersion* (Fig. 2 (c)). From this set of 1182 pairs, we then created three disjoint datasets - a) *training* b) *testing* and c) *validation* (Fig 2 (d)), by means of the following steps:

1. We established a random order for the selected ontology pairs, using a simple algorithm to add a random number to the ontology pair (Fig. 2(c));
2. We sorted the set generated in the previous step;
3. We partitioned it into three equal sets, containing the same number of *PrevVersion* and *NotPrevVersion* pairs, obtaining the following dataset: 1) *training*; 2) *testing* and 3) *validation*, (Fig. 2 (d));
4. For each dataset, we generated Weka Attribute-Relation File Format (ARFF)[6] files of tuples.

4.2 Results of the Classifier's Performances

The bar diagrams in Fig. 3, Fig 4 and Fig. 5 show the performance results of the three classifiers, with respect to the rate of classification, precision and recall. As legend we use:

Classified(Correct) indicates the number of tuples correctly classified.

Classified(Incorrect) indicates the number of tuples incorrectly classified.

Precision(PrevVersion) indicates the proportion of tuples correctly classified as $PrevVersion$ among all those that were classified as $PrevVersion$, i.e., $\frac{CPV \cap EPV}{CPV}$ where CPV is the set of tuples classified as $PrevVersion$, and EPV is the set of tuples identified as $PrevVersion$ through manual evaluation.

[6] An ARFF file is a text file that describes a list of instances sharing a set of attributes, see `http://www.cs.waikato.ac.nz/ml/weka/`.

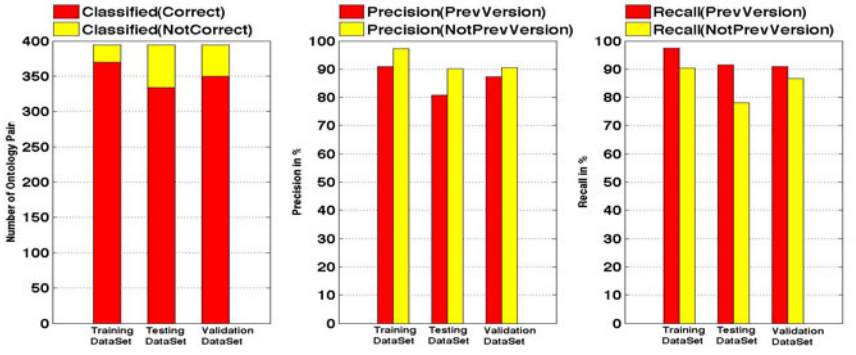

Fig. 3. *Left*: DT classification performance; *Middle*: DT precision performance; *Right*: DT recall performance

Precision(NotPrevVersion) similarly, indicates the proportion of tuples correctly classified as $NotPrevVersion$ among all those that where classified as $NotPrevVersion$

Recall(PrevVersion) indicates the proportion of tuples classified as $PrevVersion$, among those that were identified as $PrevVersion$ in our evaluation, i.e., $\frac{CPV \cap EPV}{EPV}$.

Recall(NotPrevVersion) similarly, indicates the proportion of tuples classified as $NotPrevVersion$, among those that were identified as $NotPrevVersion$ in our evaluation.

In general, all three classifiers performed well. SVM performed better than the other two reaching an average of 90% of the tuples correctly classified, an average precision of 87.2% and an average recall of 95% of tuples classified in the *PrevVersion* class[7]. Moreover, SVM is also more stable than DT and NB over the three phases of *Training*, *Testing* and *Validation*. In fact, compared to the other two classifiers, SVM presents very little fluctuations of the values of *Classification(Correct)*, *Precision (PrevVersion)* and *Recall (PrevVersion)* (Fig. 4). This confirms the fact that SVM is, in general, more accurate and gives better results than NB and DT [30]; in particular with complex distribution of data such as the one shown in Fig. 1, where the relation between the data attribute and the class attribute is complex. Furthermore, while having similar complexity to NB (polynomial time, [30,17]), SVM was also the fastest. It took less than one minute to build the model and to use it to classify new tuples. DT is the second best classifier with an average of 89% of tuples correctly classified, an average precision of 86% and an average recall of 93% of tuples classified as *PrevVersion*. It was less stable and slower in running performance than NB with an average of 80%

[7] The precision and recall measures being correlated for $NotPrevVersion$, we focus here on the values for $PrevVersion$. It is worth noticing also that the measures of recall are relative to the set of ontology pairs selected using the method described in Section 2.

of tuples correctly classified, an average precision of 71.5% and an average recall of 100% of tuples classified as *PrevVersion*. This can be observed comparing the diagrams in the middle of Fig. 3 and Fig. 5. The DT *Precision(PrevVersion)* performance is more fluctuant than the one of NB. Similar differences appear for *Classification(Correct)* and *Recall(PrevVersion)*. Moreover, due to the exponential complexity of DT [16], NB required about half of the time of DT (three minutes) to built the model and to use it. Finally, NB was the worst performer with 71% classification accuracy compared to 87.2% for SVM. This may be due to the fact that NB is based on a very strong assumption that the ontology versioning features are all independent from each other. In our case, however, *syntacticSimilarity* and *lexicographicSimilarity* are related to each other.

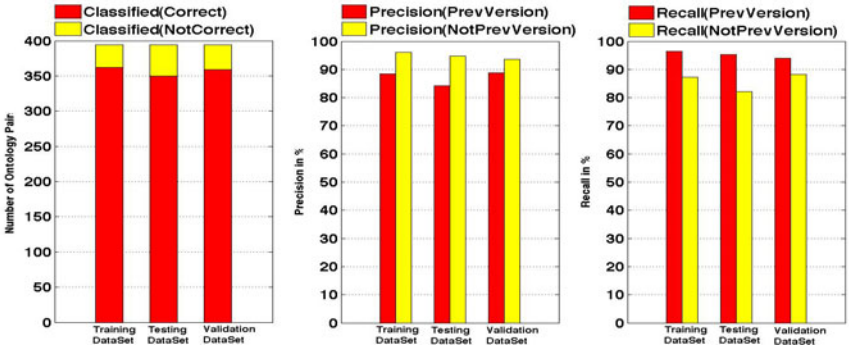

Fig. 4. *Left*: SVM classification performance; *Middle*: SVM precision performance; *Right*: SVM recall performance.

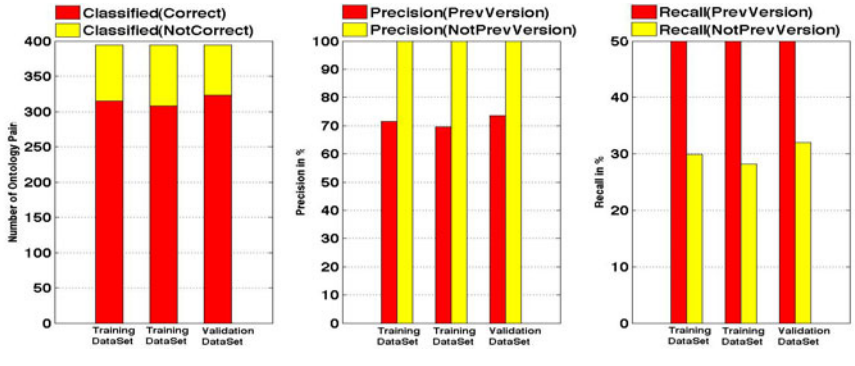

Fig. 5. *Left*: NB classification performance; *Middle*: NB precision performance; *Right*: NB recall performance.

4.3 Discussion

One of the advantages of DT over SVM was that using DT we could analyze the tree model it generates to examine in details the effect of each attribute on

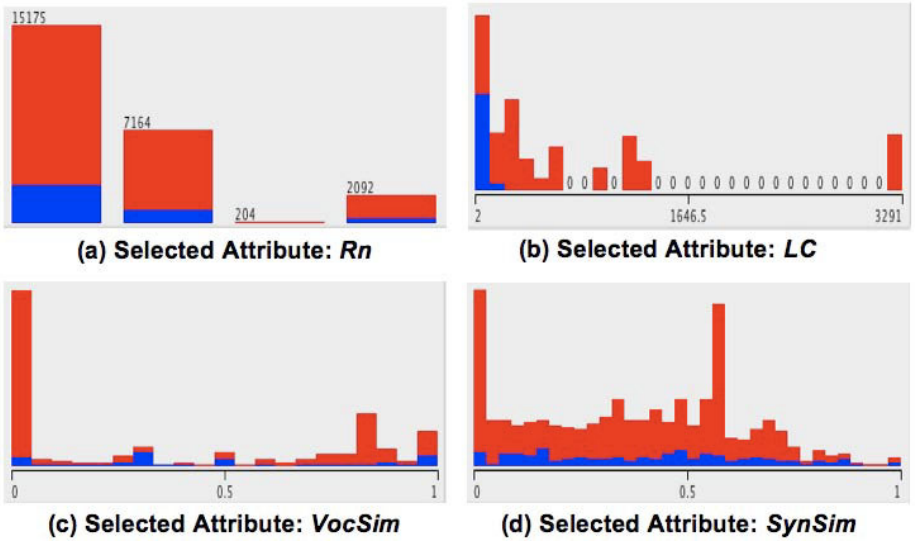

Fig. 6. Each diagram represents the single attribute contribution to identifying ontology versions. Blue color represents pairs classified as versions by the Decision Tree classifier, and red color are pairs which are not recognized as versions.

the identification of ontology versions (see Fig. 6). The ontology URI versioning patterns (R1-R6 rules) have shown to represent a good starting point to detect candidate versioning pairs of ontologies to be analyzed further by the classifiers, Fig. 6 (a).

Unsurprisingly, we could observe that the length of the ontology chains (LC) is the most relevant attribute to distinguish ontology versions from otherwise related ontology pairs Fig. 6 (b). Basically, we can easily show in our results that long sequences of versions tend to be rare, and that most of the actual version sequences have no more than 6 steps. This can probably be explained by the fact that publishing ontologies online is a relatively recent practice. The the average length of the chains of ontology versions online can therefore be expected to increase with time. As "secondary attributes", the similarity measures we employed also played an important role in the classification, particularly in distinguishing version pairs in the cases where the length of the chain was not sufficient (i.e., when the chains are reasonably short, see Fig. 6 (c,d)). Looking at Fig. 7. obtained from plotting the classification of our dataset with respect to both the lexicographic and syntactic similarity measures, an interesting phenomenon appears. Indeed, it shows that the place on the plot where one would most likely find versions is not at any of the extremes, but somewhere in the middle. This seems to indicate that, contrary to our initial intuition, ontology version pairs not only have a certain level of overlapping, but also a significant number of "changes", both in their vocabularies and structures.

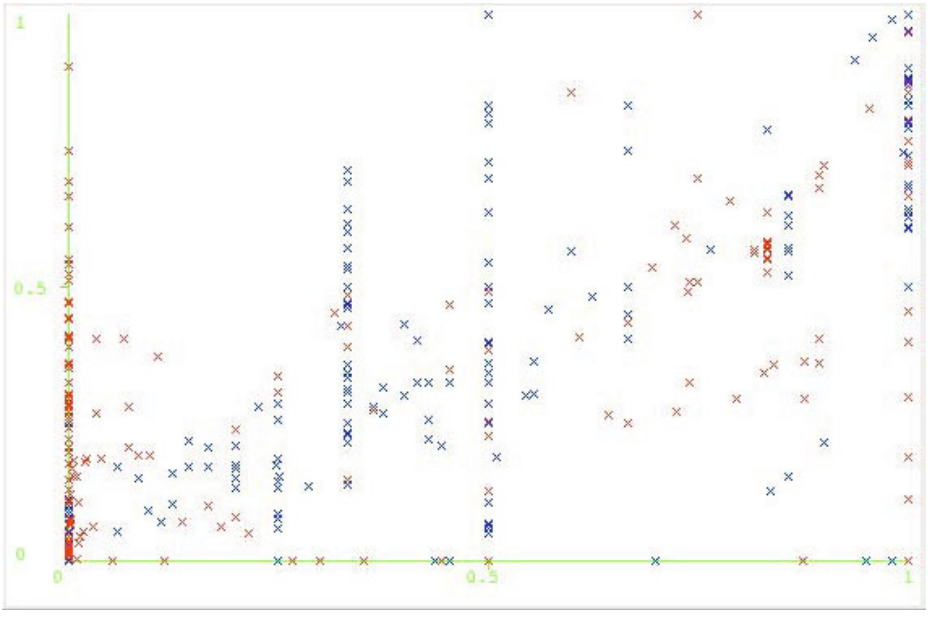

Fig. 7. Plot of the syntactic (x) and the lexicographic (y) similarity measures. Blue dots represent pairs classified as versions by the Decision Tree classifier, and red dots are pairs which are not recognized as versions.

5 Related Work

To the best of our knowledge, detecting versions of ontologies in the context of SWSE systems has not yet been approached by any of the current SWSEs, such as Swoogle, Sindice and Watson.

From both theoretical and practical points of view, most of the previous works address the ontology versioning problem focusing on *Ontology Version management* [20,22,27,28] and on *Ontology Evolution* [12,21,23,25] during the phase of ontology development. Klein [19] defines the *Ontology Versioning* problem as "*the ability to manage ontology changes and their effects by creating and maintaining different variants/mutants/versions of the ontology*". However, such sophisticated versioning mechanisms or system are not yet available. Heflin [15] formalized a semantic model for managing ontologies in distributed environments, such as the Web. On this basis, Heflin [14] developed Simple HTML Ontology Extensions (SHOE) as an extension of HTML to represent ontology-based knowledge using additional tags such as Backward-Compatible-With to specify that the current version of an ontology is compatible with previous versions. Later, Patel-Schneider [4] extended OWL (Ontology Web Language) providing new mechanisms, such as *owl:versionInfo*, *owl:priorVersion*, *owl:backwardCompatibleWith* and *owl:incompatibleWith*, to make the links explicit between versions of ontologies. The main drawback of all these mechanisms is that they are very rarely used, as ontology developers hardly ever make the

effort of applying them. Instead, they tend to codify information related to the version of an ontology directly in its URI [3]. Hartmann and Palma proposed the Ontology Metadata Vocabulary (OMV) [13], which provides an extensive vocabulary for describing ontologies, including provenance (e.g., creator, contributor); relationship (e.g., import, backward compatibility), format (e.g., ontology language and syntax), etc. However, the ontologies collected by SWSEs are rarely described using OMV. Several practical approaches have focused on comparing two different versions of ontologies in order to find out the differences and identify the changes. In particular, PROMTDIFF [26] compares the structure of ontologies and OWLDiff (`http://semanticweb.org/wiki/OWLDiff`) computes the differences by checking the entailment of the two sets of axioms. SemVersion [31] compares two ontologies and computes the differences at both structural and semantic levels. OntoDiff is a software tool that compares versions to identify and manage the ontology changes locally at both structural and content levels [29].

Closer to our work but applied on XML documents, [9] attempts to identify versions of documents using naive Bayes classifiers. The authors use a similarity measure dedicated to XML documents as input for the classifiers, and apply the approach on a set of automatically generated documents in a closed domain.

6 Conclusion and Future Work

In this paper, we have presented an approach to detect versions of ontologies in the context of Semantic Web Search Engines. The method is based on two main steps. The first step tackles the issue of selecting pairs of ontologies as candidate versions (see Section 2). The second step deals with the issue of deciding whether the selected ontology pairs are versions or not (see Section 3). Our approach solution is based on the application of machine learning techniques to a set of attributes, which tend to characterize ontology versions. In particular, we compared the *Naive Bayesian*, *Support Vector Machine* and *Decision Tree* classifiers. SVM achieved the highest level of precision in this task, 87%. This result shows that characteristics such as similarity and length of chain can be used as features to decide whether a pair of ontologies represent a version link or not, by using complex algorithms from Machine Learning. Our approach therefore provides a reasonably accurate method that can be used by a SWSE to detect ontology versions automatically.

This work will be the starting point for both empirical and practical future studies. From an empirical point of view, it is possible to analyze different ontology versions to better understand the ontology engineering practices behind the modeling process. These different versions provide data about the development process of ontologies, showing how they reach stability or adapt to changes in the domain. In other words, once we have detected the links between different versions of the ontologies, it becomes possible to explore how ontology versions actually evolve on the Semantic Web, in particular with the aim of discovering relevant *patterns* in ontology evolution. It will also be interesting to study, for

example, the inter-consistency of ontology versions. Furthermore, as part of our broader work on building a framework for the management of ontology relationships (see e.g., [2]), one of our future directions of research is to consider versioning links in combination with other high level ontology relationships such as *inclusion*, *compatibility*, or *agreement* [6].

From a practical point of view, we believe that making explicit such relations between ontologies, including versioning links, can facilitate the ontology selection in a SWSE. This assumption is currently being tested based on an integration with the Watson system (see `http://smartproducts1.kmi.open.ac.uk:8080/WatsonWUI-K/`).

References

1. Alani, H., Brewster, C., Shadbolt, N.: Ranking ontologies with AKTiveRank. In: 5th Int. Semantic Web Conf., Athens, Georgia, USA (2006)
2. Allocca, C., d'Acquin, M., Motta, E.: Door: Towards a formalization of ontology relations. In: Proc. of the IC on Knowledge Eng. and Ontology Dev., KEOD (2009)
3. Allocca, C., d'Aquin, M., Motta, E.: Detecting different versions of ontologies in large ontology repositories. In: International Workshop on Ontology Dynamic at International Semantic Web Conference (2009)
4. Bechhofer, S., Harmelen, F.V., Hendler, J., Horrocks, I., McGuinness, D.L., Patel-Schneider, P.F., Stein, L.A.: OWL Web Ontology Language Reference. Technical report (2004)
5. Bouckaert, R.R., Frank, E., Hall, M., Kirkby, R., Reutemann, P., Seewald, A., Scuse, D.: WEKA Manual for Version 3-6-2, Berlin (2010)
6. d'Aquin, M.: Formally measuring agreement and disagreement in ontologies. In: K-CAP, pp. 145–152. ACM, New York (2009)
7. d'Aquin, M., Motta, E., Sabou, M., Angeletou, S., Gridinoc, L., Lopez, V., Guidi, D.: Toward a new generation of semantic web applications. Intelligent Systems 23(3), 20–28 (2008)
8. David, J., Euzenat, J.: Comparison between ontology distances (Preliminary results). In: Sheth, A.P., Staab, S., Dean, M., Paolucci, M., Maynard, D., Finin, T., Thirunarayan, K. (eds.) ISWC 2008. LNCS, vol. 5318, pp. 245–260. Springer, Heidelberg (2008)
9. de Brum Saccol, D., Edelweiss, N., de Matos Galante, R., Zaniolo, C.: Xml version detection. In: ACM Symp. on Doc. Eng., pp. 79–88 (2007)
10. Ehrig, M., Haase, P., Hefke, M., Stojanovic, N.: Similarity for ontologies - a comprehensive framework. In: Karagiannis, D., Reimer, U. (eds.) PAKM 2004. LNCS (LNAI), vol. 3336. Springer, Heidelberg (2004)
11. Friedman, N., Geiger, D., Goldszmidt, M., Provan, G., Langley, P., Smyth, P.: Bayesian network classifiers. Machine Learning, 131–163 (1997)
12. Antoniou, G., Flouris, G., Plexousakis, D.: Evolving ontology evolution. In: Wiedermann, J., Tel, G., Pokorný, J., Bieliková, M., Štuller, J. (eds.) SOFSEM 2006. LNCS, vol. 3831, pp. 14–29. Springer, Heidelberg (2006)
13. Hartmann, J., Palma, R., et al.: Ontology metadata vocabulary and applications. pp. 906–915 (October 2005)
14. Heflin, J., Hendler, J.A.: Dynamic ontologies on the web. In: Proc. of the 7th Nat. Conf. on AI and 12th Conf. on Inn. App. of AI, pp. 443–449. AAAI Press / The MIT Press (2000)

15. Heflin, J., Pan, Z.: A model theoretic semantics for ontology versioning. In: McIlraith, S.A., Plexousakis, D., van Harmelen, F. (eds.) ISWC 2004. LNCS, vol. 3298, pp. 62–76. Springer, Heidelberg (2004)
16. Hyafil, L., Rivest, R.: Constructing optimal binary decision trees is np-complete. Information Processing Letters 5(1) (1976)
17. John, G., Langley, P.: Estimating continuous distributions in bayesian classifiers. In: Proc. of the 11th Conf. on Uncertainty in AI, pp. 338–345. Morgan Kaufmann, San Francisco (1995)
18. Quinlan, J.R.: Induction of decision trees. Machine Learning 1, 81–106 (1986)
19. Klein, M., Fensel, D.: Ontology versioning on the semantic web. In: Proc. of the Inter. Semantic Web Working Symposium (SWWS), pp. 75–91 (2001)
20. Klein, M., Kiryakov, A., Ognyanoff, D., Fensel, D.: Finding and specifying relations between ontology versions (2002)
21. Liang, Y., Alani, H., Shadbolt, N.R.: Change management: The core task of ontology versioning and evolution. In: Proc. of Postgraduate Research Conf. in Elect, Photonics, Communication. and Net, and Computing Science 2005 Lancaster, United Kingdom (PREP 2005), pp. 221–222 (2005)
22. Maedche, A., Alexander, Motik, B., Stojanovic, L., Studer, R., Volz, R.: Managing multiple ontologies and ontology evolution in ontologging. In: Proc of the IFIP 17th World Computer Congress - TC12 Stream on IIP, Deventer, The Netherlands, pp. 51–63. Kluwer, B.V., Dordrecht (2002)
23. Maedche, A., Motik, B., Stojanovic, L., Stojanovic, N.: User-driven ontology evolution management, pp. 285–300. Springer, Heidelberg (2002)
24. Maedche, A., Staab, S.: Comparing ontologies-similarity measures and a comparison study. In: Proc. of EKAW 2002 (2002)
25. Noy, N.F., Klein, M.: Ontology evolution: Not the same as schema evolution. Knowledge and Information Systems 6(4), 428–440 (2004)
26. Noy, N.F., Musen, M.A.: Promptdiff: A fixed-point algorithm for comparing ontology versions. In: 18th National Conf. on Artificial Intelligence, AAAI (2002)
27. Noy, N.F., Musen, M.A.: Ontology versioning in an ontology management framework. IEEE Intelligent Systems 19, 6–13 (2004)
28. Redmond, T., Smith, M., Drummond, N., Tudorache, T.: Managing change: An ontology version control system. In: OWLED. CEUR Proc., vol. 432 (2008)
29. Tury, M., Bieliková, M.: An approach to detection ontology changes. In: ICWE Proc. of the 6th Int. Conf. on Web Eng., p. 14. ACM, New York (2006)
30. Vapnik, V.: The Nature of Statistical Learning Theory. Springer, New York (1995)
31. Volkel, M.: D2.3.3.v2 SemVersion Versioning RDF and Ontologies. EU-IST Network of Excellence (NoE) IST-2004-507482 KWEB