Ontology Population from the Web: an Inductive Logic Programming-Based Approach

Rinaldo Lima, Hilário Oliveira, and Fred Freitas
Informatics Center, Federal University of Pernambuco
Recife, Brazil
{rjl4, htao, fred}@cin.ufpe.br

Bernard Espinasse
LSIS, Aix Marseille University
Marseille, France
bernard.espinasse@lsis.org

Abstract— **The rapid growth of the Web and the information overload problem demand the development of practical information extraction (IE) solutions for web content processing. Ontology Population (OP) concerns both the extraction and classification of instances of the concepts and relations defined by an ontology. Developing IE rules for OP is an intensive and time-consuming process. Thus, an automated mechanism, based on machine-learning techniques, able to convert textual data from web pages into ontology instances may be a crucial path. This paper presents an inductive logic programming-based method that automatic induces symbolic extraction rules, which are used for populating a domain ontology with instances of entity classes. This method uses domain-independent linguistic patterns for retrieving candidate instances from web pages, and a WordNet semantic similarity measure as background knowledge to be used as input by a generic inductive logic programming system. Experiments were conducted concerning both the instance classification problem and a comparison with other popular machine learning algorithms, with encouraging results.**

Keywords—Ontology Population, Inductive Logic Programming, Pattern Learning, Information Extraction

I. Introduction

The Web has become the major source of information, encompassing all the news, encyclopedic data, etc. Thus, it is natural to think of converting Web textual data to semantic models, such as ontologies, as a crucial path to structure knowledge [2].

Ontologies, from the computer science point of view, consist of logical theories that encode knowledge about a certain domain in a declarative way [5, 10]. Ontologies encompass definitions of concepts, properties, relations, constraints, axioms and instances about a certain domain or universe of discourse. They also provide conceptual and terminological agreements among the members of a group (humans or computational agents) that need to share information. On the other hand, the development of ontologies relies on domain experts or ontology engineers that typically adopt a manual construction process that turns out to be very time-consuming and error-prone [5]. Hence, an automated or semi-automated mechanism able to extract the information contained in existing web pages into ontologies is highly desired.

In this scenario, Ontology Population (OP) plays an important role as a means for knowledge base construction and maintenance that enables us to relate text data, from the web in particular, to ontologies [17].

This research work is based on a natural language preprocessing task that not only takes into account the typical lexical-syntactic aspects present in the English language, but also exploits semantic similarity between classes and candidate class instances. In addition, we perform an automatic induction of symbolic extraction rules from examples. In other words, the main goal of this paper is to describe and evaluate an approach to automatically generate, via an Inductive Logic Programming (ILP) framework, extraction rules (expressed as Horn clauses) for populating domain ontologies from the Web. The proposed ILP-based approach to OP also exploits the semantic similarity between classes and candidate class instances.

The rest of this paper is organized as follows: Section II is dedicated to related work. Section III presents some basic concepts addressed by the paper about OP and ILP. Our ILP-based method to populate ontologies is described in Section IV. Experimental results are presented and discussed in Section V. Finally, Section VI concludes this paper and outlines future work.

II. Related Work

Several approaches have been developed for extracting class instances from textual data in general.

ISOLDE [21] generates a domain ontology from a seed ontology by exploiting a general purpose NER system and lexico-syntactic patterns to extract candidates of concepts. These candidates are then filtered according to their statistical significance, and the knowledge obtained from available online resources, such as Wikipedia.

OPTIMA [11] consists of a semi-automated system for populating ontologies from unstructured or semi-structured texts. It assigns instances to concepts by calculating the fitness value between a candidate instance and each concept in the ontology, using the hierarchical syntactic information of the ontology schema.

BOEMIE [3] combines an Ontology-based Information Extraction engine with an inference engine for extracting “primitive” concept instances, which are then integrated and interpreted (through abductive reasoning) to form instances of “composite” and more abstract concepts. BOEMIE relies on a term/synonym extraction engine that uses machine learning to identify instances of both concept and relations.

More recently, Patel et al. (2010) conducted some experiments using ILP techniques to induce rules that extract instances of various named entity classes. They also reported a substantial reduction in development time by a factor of 240 when ILP is used for inducing rules, instead of involving a domain specialist in the entire rule development process.

One point in common with all above systems is that they employ domain independent lexico-syntactic patterns, sometimes combined with statistical methods, for assessing candidate instances. Compared to the aforementioned related work, our contribution differs from them in several aspects. First, the proposed ILP-based method allows an easier and flexible integration of background knowledge provided by other levels of linguistic analysis, as demonstrated by the integration of a semantic similarity predicate in our learning model. Second, the main advantages of our ILP-based approach over statistical-based ones is that not only the learned patterns are expressed in a symbolic form which is more easily interpreted by a knowledge engineer, but also allows the integration of considerable amount of prior knowledge as part of the solution to the problem. Finally, according to [16], when compared with a handcrafting rule approach, an ILP-based method can provide a complete and consistent view of all significant patterns in the data at the level of abstraction specified by the knowledge engineer. Thus, an ILP-based approach enables the discovery of rules that could be missed by the domain expert, making also possible to scale the rule development to a larger training dataset.

III. Ontology Population and ILP

This section describes some basic concepts about Ontology Population, and ILP systems focusing on the most important machine learning aspects concerning our motivation for the induction of symbolic rules in our research work.

A. *Ontology Population*

Ontologies have become extremely popular as a mean for representing machine-readable semantic knowledge. On the other hand, manually acquiring domain knowledge aiming at enriching an ontology is also a time-consuming task. As a result, automated (semi-automated) construction, or enrichment of existing ontologies have become a major subject study in the *Ontology Learning and Population* subfield [14]. In particular, this work focus on the *Ontology Population* task, i.e., the acquisition of new instances of concepts (is-a relations) to an existing ontology [17]. Generally, OP addresses the both the extraction and classification of instances of the concepts and relations defined by an ontology. Instantiating ontologies with new factual knowledge is a relevant step towards the provision of valuable ontology-based knowledge services [5].

In the last few years, various approaches have been proposed to OP from unstructured text. Many of them combine Natural Language Processing (NLP) techniques, e.g., syntactic parsing and named-entity recognition, with machine-learning techniques. In a typical OP process, the elements learned are instances of concepts, instances of relations or both.

B. *Inductive Logic Programming*

The term Inductive Logic Programming (ILP), first introduced by Stephen Muggleton in 1991, represents a subfield of machine learning using first order logic programming as a uniform representation for examples, background knowledge and hypotheses [15].

The most commonly addressed task in ILP consists of learning logical definitions of relations, in which tuples that belong or do not belong to the target relation are given as examples. From these examples, an ILP system can induce a logic program (or a logical theory) defining the target relation in terms of other relations that are given as background knowledge [12].

According to De Raedt [18] there are two main motivations for using ILP. The first one is that it overcomes the representational limitations of attribute-value learning systems which employ a table-based representation in which the instances correspond to rows in a table, the attributes to columns, and, for each instance, a single value is assigned to each one of the attributes. In contrast, ILP can deal with multi-relational data [18]. The second motivation is that it rather employs logic, i.e., a declarative representation. This implies that hypotheses are understandable and interpretable by humans. By using logic, ILP systems are also able to integrate previous knowledge, or *background knowledge* (BK), of the domain into the induction process. Such BK can be provided by the domain expert in the form of definitions of auxiliary relations or predicates that can be used by the learner.

Compared to other propositional inductive learning techniques such as decision trees or decision lists, ILP is thus more powerful in several aspects:

- ILP uses an expressive first-order rule formalism enabling the representation of concepts and hypotheses that cannot be represented by the typical attribute-value (propositional) machine learning techniques.
- Many techniques and theoretical results from Computational Logic can be used and adapted for the needs of inductively generating rules from examples.

Further details on ILP can be found in [15, 12, 19].

IV. An ILP-based Method for Populating Domain Ontologies

The supervised method for ontology population presented in this paper takes profit of the high redundancy present in Web content, considering it as a big corpus. Sharing the same idea, several authors pointed it out as an important feature because the amount of redundant information can represent a measure of its relevance. Moreover, we take into account the portability issue, i.e., the method should able to perform independently of the domain.

We adopted the ILP as the core component for machine learning in our method because it can provide rules in symbolic form, which can be fully interpreted by a knowledge engineer.

Consequently, the user can either refine these rules or simply converting them to other rule formalisms, such as SWRL[1].

As shown in Fig. 1, the method proceeds according to two phases, in this order: the *Learning phase* which generates extraction rules from a corpus of annotated documents, followed by the *Exploitation phase* which applies the learned rules to extract class instances from unseen documents. After the initial Corpus Retrieval step, these two stages firstly have to perform the *Text Preprocessing* and the *BK Generation* steps, which are common to both. Then, in the Learning phase, the system performs the *Rule Induction* step, whereas in the Exploiting phase, the *Rule Application* task is in charge of applying the learned rules on the generated BK in order to finally extract ontological class instances.

In the following section, we describe the aforementioned steps of the system architecture in more detail.

A. Corpus Retrieval

The first task in our method, the corpus retrieval task, starts retrieving sentences from the Web in order to build a working corpus. We rely on a set of domain-independent linguistic patterns for that. Fig. 2 presents the patterns, i.e., Hearst's pattern [9] used for gathering relevant documents containing candidate instances of concepts in a domain ontology.

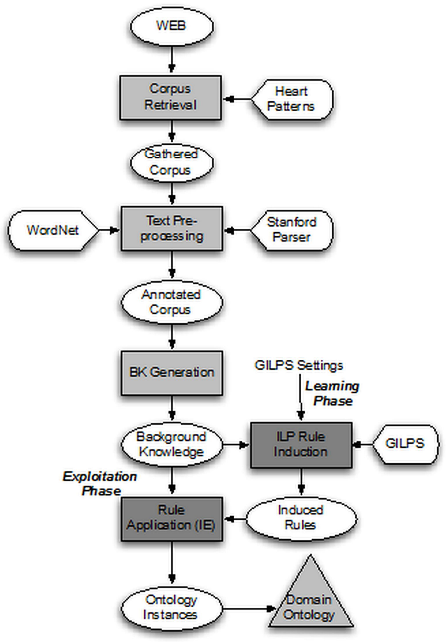

Figure 1. Overview of the ILP-based ontology population system

P1:	<CANDIDATE> *is a/an* <CLASS>
P2:	<CLASS>(s) *such as* <CANDIDATES>
P3:	*such* <CLASS>(s) *as* <CANDIDATES>
P4 and P5:	<CLASS>(s) (*especially/including*) <CANDIDATES>
P6 and P7:	<CANDIDATES> (*and/or*) *other* <CLASS>(s)

Figure 2. Domain-independent Hearst patterns

After the user's choice of a class from a domain ontology, the system retrieves some documents based on both the *label* of the chosen class, and the patterns *P* (Fig. 2). For instance, selecting the *Country* class, the patterns above would match sentences in natural language such as: *"is a country"; "countries such as"; "such countries as"; "countries especially"; "countries including"; "and other countries"; "or other countries*". These phrases likely include instances of the *Country* class in the CANDIDATE(S) part [9].

Each query is submitted to a web search engine, and the first *n* web documents are fetched for each pattern. We are interested in extracting sentences like, "such countries as CANDIDATES" or "CANDIDATE is a country" where CANDIDATE(S) denotes a single noun phrase or a list of noun phrases. For instance, in the sentence: "Why did countries such as Portugal, France grow rapidly in the 1930's?", the terms "Portugal" and "France" are extracted as candidate instances of the *Country* class.

B. Text Pre-processing

Once the corpus retrieval step is finished, two text preprocessing techniques take place (ii) *lexico-syntactic analysis*, and (ii) *semantic similarity measuring*.

Lexico-Syntactic Analysis. The main goal of the system is to automatically induce extraction patterns that discovers *hypernymy relations* (is-a relations) between two terms. For doing that, we need a representation formalism that expresses these patterns in a simple and effective way. We defined a set of lexico-syntactic features produced by the preprocessing component in our architecture. These features are the building blocks that composes the BK that necessary to the induction of extraction rules.

The prototype system developed for validating the proposed approach relies on the Stanford CoreNLP[2], a suite of core NLP tools proposed by the Stanford NLP Group. This software performs the following sequence of NLP sub-tasks: *sentence splitting*, *tokenization*, *Part-of-Speech* (POS) *tagging*, *lemmatization* (which determines the base form of words), and *Named Entity Recognition* (NER) which labels sequences of words in a text into predefined categories such as *Person*, *Organization*, *Date*, to name a few.

Semantic Similarity Measuring. Semantic similarity measures based on WordNet[3] have been widely used in NLP applications, and they typically take into account the WordNet taxonomical structure to produce a numerical value for assessing the degree of the semantic similarity between two terms.

We adopted the semantic similarity measure proposed by Wu and Palmer (1994). This similarity measure provides the degree of similarity between the class *C* and a candidate instance C_i. It relies on finding the most specific concept that subsumes both the concepts under measurement in WordNet. The path length from the shared concept to the root is scaled by the sum of the distances of the concepts to the subsuming

[1] http://www.w3.org/Submission/SWRL/

[2] http://nlp.stanford.edu/software/corenlp.shtml
[3] http://wordnet.princeton.edu/

concept. This similarity measure has a competitive performance against other ones [13].

C. Background Knowledge Generation

After the text-preprocessing task, we carry out the critical tasks on identifying, extracting, and appropriately representing relevant BK to the problem at hand. This process is not trivial because, without it, the ILP desirable advantages that make it different from traditional learning algorithms cannot be truly exploited.

Previous research has shown that shallow semantic parsing can provide very useful features in several information extraction related tasks [8]. Accordingly, we explore the features listed in Tab. 1, which constitute the BK in our method. These features provide a suitable feature space for the classification problem of candidate instances, as they describe each token in the corpus. Furthermore, we calculate the similarity degree between each token (tagged as singular or plural noun by the POS tagger), and a class in the input ontology. We illustrate in Tab. 1 the BK that characterizes the candidate instance of the *Mammal* class, "Lion".

Given that the WordNet similarity values are in [0,1] range, we perform a discretization of this numerical attribute by creating 10 bins of equal sizes (0.1 each). For example, if the WordNet similarity value between the class instance "Lion" and the class *Mammal* is 0.96, we put this value in the 10th bin which corresponds to the predicate t*_wnsim(t_id, mammal, '09-10')* that will be finally included in the BK. The predicate t_wnsim(A, mammal, '09-10') means that the token 'A' has a similarity score between 0.9 and 1.0 with the *Mammal* class.

Differently from other machine learning approaches that use feature vectors for representing context windows (*n* tokens on the right/left of a given word *w* in a sentence), we use the *next/2* predicate which relates one token to its immediate successor in a sentence, in conjunction with other 7 additional predicates that constitute our BK (Tab. 1).

D. ILP Rule Induction

In this point of processing, we had to adjust some parameters of GILPS for the task at hand. One of such parameters defines the *language bias* that delimits and biases the possibly huge hypothesis search space. In GILPS, this is achieved by providing appropriate *mode declarations*.

Mode declarations characterize the format of a valid hypothesis (rule). They also inform both the *type*, and the *input/output modes* of the predicate arguments in a rule. There are two types of mode declarations in GILPS: *head* and *body*. A mode head declaration (*modeh*) denotes the target predicate, i.e., the head of a valid rule that the ILP system has to induce, whereas a mode body declaration (*modeb*) specifies the literals, or ground predicates, that may appear in a rule body. Mode body declarations usually refer to predicates defined in the background knowledge, but they can also refer to the target predicate in the case of recursive theories [15].

Another important decision is related to the *engine* parameter in GILPS, since it allows the user to define the way rules are specialized/generalized, i.e., how the hypotheses space are traversed, top-down or bottom-up manner. For the purposes of this research work, the top-down approach was selected because it enables the construction of shorter rules [19].

TABLE 1. ILP PREDICATES FOR THE CANDIDATE INSTANCE "LION"

Predicates	Meaning
token (t_1)	t_1 is the token identifier
t_length (t_1, 4)	t_1 has length of 4
t_ner (t_1, o)	t_1 is not any NE according to the NER
t_orth(t_1, upperInit)	t_1 has an initial uppercase letter
t_pos (t_1, nnp)	t_1 is a singular proper noun
t_next(t_1, t_2)	t_1 is followed by the token t_2
t_type (t_1, word)	t_1 is categorized as a word
t_wnsim(t_1, mammal, '09-10')	t_1 has a similarity score between 0.9 and 1.0 with the *Mammal* class

E. Rule Application

The set of rules induced by the previous step are finally applied on an unseen preprocessed corpus for extracting instances that populate the domain ontology.

Furthermore, this last step is in charge of the domain ontology enrichment by providing the typical services of redundancy control, and automatic conversion from the textual representation of the retrieved instances to the corresponding class assertions axioms in the domain ontology. For performing the last service, we used the OWL/API[4] for manipulating and serializing the final extracted instances. Such instances are finally integrated into our domain ontology that was originally created with the Protégé[5] ontology editor .

V. EXPERIMENTAL EVALUATION

In this section, we first describe how the corpus was created and annotated. Next, we present and discuss the extraction performance results of two experiments concerning the *instance classification* problem, and a *comparative assessment* with other popular machine learning algorithms.

A. Corpora Creation and Annotation of Examples

The corpora used in this evaluation were compiled using the 7 surface patterns listed in Section IV. Although our domain ontology has numerous classes, for the purposes of our research, we restricted the evaluation on 5 classes, namely *Country*, *Disease, Bird*, *Fish, and Mammal* classes. For each class, the system retrieved 420 sentences that were equally distributed into sentences containing positive and negative candidate instances. We used the Bing Search Engine API[6] for collecting a total of 2100 sentences (420 sentences each class).

The task of inducing target predicates in GILPS requires that positive and negative examples be explicitly indicated before the generation of the classification model. Thus, two human annotators manually tag the positive instances. There is no need to annotate the negative examples because they can be automatically identified as the complement of the positive ones.

[4] http://owlapi.sourceforge.net/

[5] http://protege.stanford.edu/

[6] http://www.bing.com/developers/s/APIBasics.html

B. Evaluation Measures and GILPS Parameters

Aiming at assessing the effectiveness of our approach, we conducted several experiments on the corpora. The performance evaluation is based on the classical measures used in IR systems, i.e., *Precision* P, *Recall* R, and *F1-measure* [1]. In all experiments reported here, we used 10-fold cross-validation that provides unbiased performance estimates of the learning algorithms.

The generic ILP system GILPS was run with its default parameters, except for the following specific settings: *theory_construction = incremental, evalfn = compression,* and *clause_length = 8.*

C. Results and Discussion

In order to estimate the classification performance of the learned rules for each class, we used 2 versions of the compiled corpora. The first version was annotated only with lexico-syntactic features (Section IV), whereas the second version has an additional WordNet semantic similarity feature. Each class was assessed separately by building a binary classifier for each one.

The experimental results shown in Tab. 2 are encouraging, since the method seems to successfully extract a significant number of positive instances from the corpora. Considering the F1 score for the *Country* class, one can observe that, as being an entity type recognized by the parser (*named entity = location*), the *Country* class classifier had a very tiny improvement on the sample with the additional WordNet predicate.

On the other hand, for the remaining classes, the rules only based on lexico-syntactic predicates are highly precise, but the achieved recall score is lower than those ones when the WordNet predicate is used. This suggests that the semantic similarity measure provided by WordNet can be very useful. In fact, a statistical significance test (*paired Student t-test*) for the difference between the F1 scores of the two experiments above were performed. This test revealed that there is a significant difference at $\alpha = 0.05$ (95% confidence interval) between them. Thus, this assessment suggests that the additional WordNet similarity predicate in the BK actually contributed to achieve better performance results.

In the following, we list some rules expressed in terms of (*number of literals)*, (*positive examples covered)*, (*negative examples covered)*, and the (*rule precision score P)*.

Rule 1: #Literals = 4, PosScore = 17, NegScore = 0, P = 100%
isa_mammal(A):- t_ner(A,misc), t_orth(A, upperinitial), t_pos(A, nn).

Rule 2: #Literals = 3, PosScore = 629, NegScore = 14, P = 97.8%
isa_country(A):- t_wnsim_country(A, '09-10'), t_ner(A, location).

Rule 3: #Literals = 4, PosScore = 329, NegScore = 28, P = 92.0%
isa_disease(A):- t_length(A,8), t_type(A, word), wnsim(A, disease,'09-10').

The ILP-based system found a perfect extraction rule for the *Mammal* class (*Rule 1*), i.e., an instance beginning with an uppercase letter, identified as "miscellaneous" by the NER, and tagged as a singular noun by the POS tagger. In *Rule 2*, the high precision score of the *Country* class is mainly due to the NER that has identified the candidate instance as a "location" combined with a high score similarity with the WordNet synset "country". *Rule* 3 classifies an instance of the *Disease* class if it is a term with 8 characters, and its similarity score with the WordNet synset "disease" is between 0.9 and 1.0.

TABLE 2. CLASSIFICATION PERFORMANCE OF THE INDUCED RULES

	No WordNet			With WordNet		
Class	**P**	**R**	**F1**	**P**	**R**	**F1**
Country	0.96	0.92	0.94	0.98	0.96	0.97
Disease	0.95	0.69	0.80	0.97	0.84	0.96
Bird	0.93	0.53	0.67	0.95	0.73	0.82
Fish	0.93	0.42	0.58	0.94	0.50	0.65
Mammal	0.93	0.39	0.55	0.93	0.49	0.64

D. Comparative evaluation

In the second experiment, we attempted to conduct a comparative evaluation of our ILP-based learning component with Decision Tree, Support Vector Machines, and Naïve Bayes algorithms. We justify this choice because they are among the top data mining algorithms according to [22].

We rely on the WEKA implementation of the pruned C4.5 decision tree (*J48*), *SMO (a variant of SVM)*, and *Naïve Bayes* algorithms [22]. In all training datasets for this comparative evaluation, the number of positive examples (minority class) were around 20% of the number of negative examples (majority class). Consequently, we employed the supervised filter *SMOTE* because these 3 statistical algorithms are, in some degree, sensible to highly skewed class distribution between positive and negative classes. The SMOTE filter is a implementation of the oversampling technique proposed in [4], which introduces new non-replicated minority-class examples. Note that SMOTE oversampling can be considered as a wrapper-based method that can make any learning algorithm cost-sensitive, according to [4].

When constructing the training datasets used in this comparative evaluation, we had to consider a sliding *window of 3 tokens* (left and right) around a candidate instance to be classified. This was necessary for simulating the role that the *next/2* predicate plays in the ILP-based representations of the examples, as a means for representing the token chaining in a sentence. Fig. 3 depicts comparative results yielded using the full set of features (lexico-semantic + WordNet predicate) on the datasets of all classes.

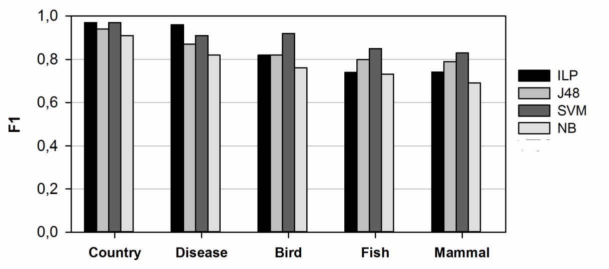

Figure 3. Comparative evaluation of the classifiers

According to the results shown in Fig. 3, the SVM and J48 algorithms yielded the more stable results. All algorithms practically had the same performance in terms of F1-measure for both *Fish* and *Mammal* classes. On the other hand, for these two last classes, the ILP-based method achieved its lowest scores. Since, by default, GILPS employs a greedy strategy for evaluating the rules that it induces, it is quite plausible that it has missed the better rules. Thus, fine-tuning the parameters of GILPS may improve a little the results on these classes. Further experiments would confirm or not that hypothesis.

The decreasing scores obtained by all classifiers (Fig. 3), suggests that a considerable number of class instances were not found in the WordNet, for the last 3 classes. In other words, all classifiers have a considerable error rate (20%-40%) when they only use lexico-syntactic features. However, at this stage of our research, we are more concerned about assessing the benefits from adding semantic-related features, rather than the typical lexico-syntactic ones, on the preprocessing step. The obtained results drive us towards considering additional more semantic-related features to the text preprocessing in future work.

Again, we carried out several statistical significance tests (*paired Student t-test* at $\alpha = 0.05$) between each pair of classifiers and, based on the results, we did not find any significant difference between the ILP-based algorithm and the other algorithms. However, the SMO performance was significantly higher than J48 and Naïve Bayes.

We planned to conduct comparative evaluations with other approaches presented in Related Work section, but this would only be possible if the compared approaches were under the same experimental setup (same corpus, same ontology, etc). Since we are using corpora retrieved from the Web, which is high dynamic in nature, we cannot provide a fair and direct comparison in this case. In any case, just for the sake of relative comparison, [6] has reported a precision performance up to 80% for the *City* class, and [24] achieved precision up to 95% for the *Country* class.

VI. Conclusion and Future Work

This paper has proposed an ILP-based method for ontology population, which mainly relies on shallow syntactic parsing, and semantic similarity measures. Two experiments were conducted in order to evaluate the effectiveness of the proposed approach. Although we have achieved encouraging results, there are still many opportunities for improvement. Indeed, the method presented here currently relies on a set of domain-independent extraction rules that usually fails to generalize on the most linguist variations. Thus, in order to improve its recall, we plan to use another formalism for sentence representation based on *dependency grammar*, which are more robust to linguist variations [9]. This formalism can be seen as dependency graphs that represent syntactic relations between words in a sentence. Finally, we intend to extract instances of relations as well.

It is worth mentioning that our method for OP does not consider the class hierarchy when assigning class instances to classes in the domain ontology. The prototype system that implements our approach only populates those classes chosen by the user at the beginning of the population process. Indeed, we intend to overcome this limitation in order to achieve a more robust framework for ontology population.

References

[1] R. Baeza-Yates, and A. B. Ribeiro-Neto, Modern Information Retrieval: Addison -Wesley Longman Publishing. Boston, USA, 1999.

[2] C. Bizer, T. Health, T. Berners-Lee. Linked Data – The Story so far. Inter. Jour. of Semantic Web and Information Systems, 5(3): 1-22, 2009.

[3] S. Castano, I. S. E. Peraldi, C. Ferrara, V. Karkaletsis, A. Kaya, R. Möller, S. Montanelli, G. Petasis, M. Wessel, Multimedia Interpretation for Dynamic Ontology Evolution. J. of Logic and Computation, 2008.

[4] V. Chawla, K. Bowyer, L. O. Hall, W. P. Kegelmeyer. SMOTE: Synthetic Minority Over-sampling Technique. J. Artif. Intell. Res. (JAIR) 16: 321-357, 2002.

[5] P. Cimiano, Ontology Learning and Population from Text: Algorithms, Evaluation and Applications. Springer-Verlag, New York, USA, 2006.

[6] D. Downey et al.. Learning Text Patterns for Web Information Extraction and Assessment. Proceedings of the 19th National Conference on Artificial Intelligence Workshop on Adaptive Text Extraction and Mining. San Jose, USA, 2004.

[7] J. Fan, A. Kalyanpur, D. Gondek, D. A. Ferrucci. Automatic Knowledge Extraction from Documents. IBM Journal of Research and Development 56(3): 5, 2012.

[8] A. Finn. A Multi-Level Boundary Classification Approach to Information Extraction, Phd thesis, University College Dublin, 2006.

[9] M. A. Hearst. Automatic Acquisition of Hyponyms from Large Text Corpora. Proc. of the 14th conference on computational linguistics. Nantes, France, pp. 539–545, 1992.

[10] P. Hitzler, M. Krötzsch, and S. Rudolph. Foundations of Semantic Web Technologies. Chapman & Hall/CRC, 2009.

[11] S. S. Kim, J. W. Son, S. B. Park, S. Y. Park, C. Lee, J. H. Wang, M. G. Jang, H. G. Park, OPTIMA: An Ontology Population System. In: 3rd Workshop on Ontology Learning and Population, 2008.

[12] N. Lavrac, and S.Dzeroski , Inductive Logic Programming: Techniques and Applications. Ellis Horwood, new York, 1994.

[13] D. Lin, An Information-theoretic Definition of Similarity. Proc. of the 15th Int. Conf. on Machine Learning. San Francisco, USA, pp. 296–304 , 1998.

[14] A. Maedche, S. Staab, Ontology Learning. In: Handbook on Ontologies , 2004.

[15] S. Muggleton, Inductive Logic Programming. New Generation Computing. Tokyo, Japan, 2001.

[16] A. Patel, G. Ramakrishnan, and P. Bhattacharya, Incorporating Linguistic Expertise Using ILP for Named Entity Recognition in Data Hungry Indian Languages, LNCS, vol. 5989, pp. 178-185, Springer Berlin Heidelberg, 2010.

[17] G. Petasis, V. Karkaletsis, G. Paliouras, A. Krithara, and E. Zavitsanos, Ontology Population and Enrichment: State of the Art, in G. Paliouras et al. (Eds.): Multimedia IE, LNAI 6050, pp. 134–166, 2011.

[18] L. De Raedt , Inductive Logic Programming. Encyclopedia of Machine Learning, pp:529-537, 2010.

[19] J. Santos, Efficient Learning and Evaluation of Complex Concepts in Inductive Logic Programming, Ph.D. Thesis, Imperial College, 2010.

[20] H. Tanev and B. Magnini, Weakly supervised approaches for ontology population, Proc. of EACL- 2006, Trento, pp.3-7, 2006.

[21] N. Weber, P. Buitelaar: Web-based Ontology Learning with ISOLDE. In: Proceedings of the Workshop on Web Content Mining with Human Language at the International Semantic Web Conference, USA, 2006.

[22] X. Wu, V. Kumar, J. R. Quinlan, J. Ghosh, Q. Yang, H. Motoda, G. McLachlan, J. Ng, B. Liu, P. Yu, Zhou S. Z., M. Steinbach, D. J. Hand, D. Steinberg. Top 10 algorithms in data mining. Knowledge Information. Systems, 14(1): 1-37, 2008.

[23] Z. Wu, and M. Palmer, Verb Semantics and Lexical Selection. Proc. of the 32nd Annual Meeting of the Association for Comp. Linguistics, New Mexico, USA, 133 –138, 1994.

[24] G. Geleijnse and J. Korst, Learning effective surface text patterns for Information Extraction, Proceedings of the EACL Workshop on Adaptive Text Extraction and Mining, pp. 1–8, 2006.