An Inductive Logic Programming-Based Approach for Ontology Population from the Web

Rinaldo Lima[1], Bernard Espinasse[2], Hilário Oliveira[1], Rafael Ferreira[1], Luciano Cabral[1], Dimas Filho[1], Fred Freitas[1], and Renê Gadelha[1]

[1] Informatics Center, Federal University of Pernambuco, Recife, Brazil
{rjl4,htao,rflm,lsc4,dldmf,fred,rnsg}@cin.ufpe.br
[2] LSIS, Aix Marseille University, Marseille, France
bernard.espinasse@lsis.org

Abstract. Developing linguistically data-compliant rules for entity extraction is usually an intensive and time-consuming process for any ontology engineer. Thus, an automated mechanism to convert textual data into ontology instances (Ontology Population) may be crucial. In this context, this paper presents an inductive logic programming-based method that induces rules for extracting instances of various entity classes. This method uses two sources of evidence: domain-independent linguistic patterns for identifying candidates of class instances, and a WordNet semantic similarity measure. These two evidences are integrated as background knowledge to automatically generate extractions rules by a generic inductive logic programming system. Some experiments were conducted on the class instance classification problem with encouraging results.

Keywords: Ontology Population, Information Extraction, Pattern Learning, Inductive Logic Programming.

1 Introduction

Ontologies, from the computer science point of view, consist of logical theories that encode knowledge about a certain domain in a declarative way [2]. They also provide conceptual and terminological agreements among humans or computational agents that need to share information. On the other hand, the development of ontologies relies on domain experts that typically adopt a manual construction process, which turns out to be very time-consuming and error-prone. Hence, an automated or semi-automated mechanism able to extract the information contained in existing web pages into ontologies, *Ontology Population* (OP), is highly desired [2].

In this scenario, the main goal of this paper is to describe and evaluate a method to automatically induce, via an Inductive Logic Programming (ILP) framework, extraction rules for OP. The proposed method also exploits the semantic similarity between classes and candidate class instances. More precisely, this method relies on: (i) a natural language preprocessing which not only takes into account the typical lexical-syntactic aspects present in the English language, but also exploits semantic similarity between ontology classes and candidate class instances, and (ii) an ILP-based induction of symbolic extraction rules (expressed as Horn clauses) from examples.

The rest of this paper is organized as follows: Section 2 is dedicated to related work. Section 3 presents some basic concepts about ILP. The ILP-based method for OP is described in Section 4. We present and discuss experimental results of our method for OP in Section 5. Finally, in Section 6 we conclude and outline future work.

2 Related Work

Several approaches have been developed for extracting class instances from textual data using machine learning techniques. KnowItAll [5] is a hybrid named-entity extraction system that combines Hearst's and some learned patterns for extracting class instances from the Web using a search engine. In order to assess the candidate instances, KnowItAll uses the PMI metric and a Naïve Bayes classifier for achieving a rough estimate of the probability that each candidate instance is correct. In [4], the authors proposed the idea that learned patterns could be used as both extractors (to generate new information) and discriminators (to assess the truth of extracted information). More recently, [9] reports some experimental results using ILP techniques to induce rules that extract instances of various named entities. Moreover, [9] 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.

Although our approach has explored the same kind of surface patterns used by most of the approaches described above, our richer set of features (POS tagging, NER, and semantic similarity measure) seems to achieve promising results. Furthermore, our research hypothesis is that an ILP-based method would allow an easier and flexible integration of background knowledge (BK) provided by other levels of linguistic analysis, as potential future work.

3 Inductive Logic Programming

Inductive Logic Programming is a subfield of machine learning which uses first order clauses as a uniform representation for examples, background knowledge, and hypotheses [7]. According to [3], there are two main motivations for using ILP: (i) it overcomes the representational limitations of attribute-value (propositional) learning systems that employ a table-based example representation; (ii) it rather employs a declarative representation, which means that the hypotheses are understandable and interpretable by humans. Moreover, by using logic, ILP systems can exploit BK in its learning (induction) process. For instance, the BK can be expressed in the form of auxiliary predicate definitions provided by the user.

Informally, the ultimate goal of ILP is to explain all of the positive and none of the negative examples. More formally, given: (i) a set of examples $E = E^+ \cup E^-$, where E^+ (positive) and E^- (negative), and; (ii) background knowledge BK, the task of ILP is to find a hypothesis H such that: $\forall e \in E^+$: $BK \wedge H \models e$ (H is *complete*), and $\forall e \in E^-$: $BK \wedge H \not\models e$ (H is *consistent*).

Many existing ILP implementations, such as GILPS [10], are closely related to Prolog and, therefore, they impose the following typical restriction to the way of how the *BK* (in terms of predicates or rules) and examples are represented. In other words, the *BK* is restricted to Prolog clause in the form *head :- body*$_1$, *body*$_2$, ..., *body*$_n$. Thus, the head is implied by the body clauses, whereas E^+ and E^- are restricted to ground facts. We refer the reader to [3], [7], [10] for further details on ILP.

4 An ILP-Based Method for Populating Domain Ontologies

Our supervised method for OP takes profit of the high redundancy present in the 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 [5], [4]. Moreover, we take into account the portability issue, i.e., the method should able to perform independently of the domain. We adopted the ILP framework as the core component for machine learning in our method because it can provide extraction rules in a 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.

One of the main advantages of ILP over other statistical machine learning algorithms 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 under consideration. Moreover, according to [9], 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.

The proposed method is composed of four main steps as illustrated in Fig. 1.

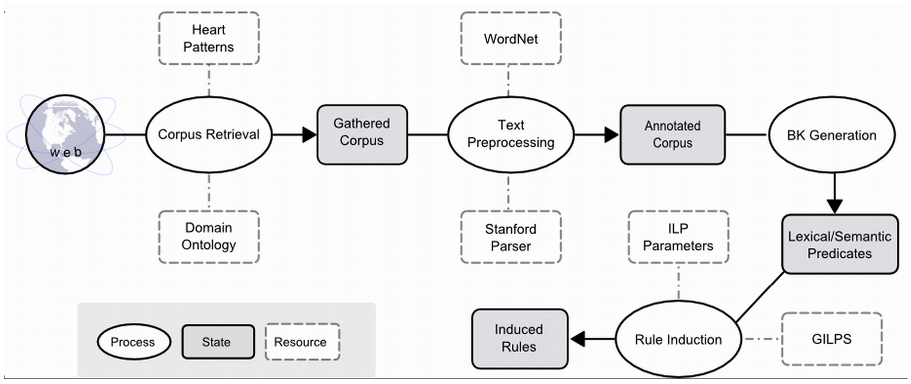

Fig. 1. Overview of the ILP-based ontology population system

In general terms, the method consists of a supervised approach to automatically generating extraction rules that subsumes lexico-syntactic patterns present in textual documents. As a result, the induced rules can be applied on an unseen set of preprocessed documents in order to extract instances that populate an ontology. In the remain part of this section, we present each system component in more detail.

4.1 Corpus Retrieval

The first step, the corpus retrieval process, starts retrieving sentences from web pages in order to constitute a corpus. We rely on a set of domain-independent linguistic patterns for this task. Fig. 2 presents the patterns used for gathering relevant documents containing candidate instances of concepts (classes) of a domain ontology.

After the user's choice of a class from this 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 above patterns 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 [2], [4], [5].

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)

Fig. 2. Domain-independent Hearst patterns

Next, 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.

4.2 Text Preprocessing

Two text preprocessing techniques are performed at this step (ii) *lexico-syntactic analysis*, and (ii) *semantic similarity measuring*.

Lexico-Syntactic Analysis. The main goal of our 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 compose the background knowledge that will be used later by the component responsible for the induction of extraction rules. The prototype system developed in this work relies on the Stanford CoreNLP [11], a Natural Language Processing (NLP) tool. This NLP tool performs the following sequence of processing task: *sentence splitting*, *tokenization*, *Part-of-Speech* (POS) *tagging*, *lemmatization,* and *Named Entity Recognition* (NER) which labels sequences of words in a text into predefined categories such as *Person*, *Organization*, *Date*, etc.

Semantic Similarity Measuring. Semantic similarity measures based on WordNet [8] have been widely used in NLP applications, and they 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 similarity measure proposed in [12] which provides the degree of similarity between the class C and a candidate class instance C_i. It relies on finding the most specific concept that subsumes both the concepts in WordNet.

4.3 Background Knowledge Generation

This step consist of identifying, extracting, and appropriately representing relevant BK for the task at hand. Previous research have shown that shallow semantic parsing can provide very useful features in several information extraction related tasks [6]. Accordingly, we explore the features listed in Tab. 1, which constitute the BK in our approach. 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 domain ontology. We illustrate in Tab. 1 the BK that characterizes the candidate instance of the *Country* class, "*France*".

Table 1. ILP Predicates for the token "France"

Predicate Generated	Meaning
token (t_1)	t_1 is the token identifier
t_length (t_1, 6)	t_1 has length of 6
t_ner (t_1, location)	t_1 is a location entity according to the NER
t_orth (t_1, upperInitial)	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, country, '09-10')	t_1 has a similarity score between 0.9 and 1.0 with the *Country* class

Given that the WordNet similarity values are in the [0,1] range, we perform a discretization of this numerical feature by creating 10 bins of equal sizes (0.1 each). Thus, for example, if the WordNet similarity value between the candidate class instance "*France*", and the class "*Country*" is 0.96, we put this value in the 10th bin which corresponds to the predicate *t_wnsim(t_id, country, '09-10')*. In other words, the predicate t_wnsim(A, country, '09-10') means that the token 'A' has a similarity score between 0.9 and 1.0 with the *Country* class.

4.4 Rule Induction

In this last step, we have to define the *language bias* which both 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. Mode head declarations (*modeh*) state the target predicate, i.e., the head of a valid rule that the ILP system has to induce, whereas *mode body* declarations (*modeb*) determine the literals (or predicates), which may appear in a rule body. In addition, the *engine* parameter in GILPS permits the user to choose the way rules are specialized/generalized, i.e., how the hypothesis space are traversed, either in *top-down* or *bottom-up* manner. The top-down approach was selected because it enables the construction of shorter theories (in term of the number of clauses) [10]. Finally, GILPS induces a set of rules that can be applied on an unseen set of preprocessed documents in order to extract instances that populate an ontology.

5 Experimental Evaluation

In this section, we describe how the corpus was created and annotated. Next, we present and discuss the results of our experiments on the OP task.

5.1 Corpora Creation and Annotation of Examples

The corpus used in this evaluation was compiled using the 7 surface patterns listed in Section 4.1. We have performed an evaluation on 5 classes, namely *Country*, *Disease, Bird*, *Fish, and Mammal* classes. For each class, the system retrieved approximately 420 sentences equally distributed into sentences containing positive and negative candidate instances. We used the Bing Search Engine API [1] for collecting a total of 2100 sentences (420 sentences for 5 classes).

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 tagged 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.

5.2 Evaluation Measures and GILPS Parameters

The performance evaluation is based on the classical measures used in IR systems, i.e., *Precision* P, *Recall* R, and F1-*measure*. In all experiments, we used 10-fold cross-validation that provides unbiased performance estimates of the learning algorithm. GILPS was run with its default parameters, except for the following specific settings: *theory_construction = incremental, evalfn = compression, clause_length = 8, nodes = 1000.* In incremental theory construction, when the best hypothesis from an example is found, all the positive examples covered by this hypothesis are retracted from the training set, whereas the *nodes* parameter determines the maximum number of hypotheses that may be derived from a single positive example.

5.3 Results and Discussion

In order to estimate the classification performance of the learned rules for each class, we used two versions of the compiled corpus as described in Section 5.1. The first version of the corpus was only annotated with lexico-syntactic features (see Section 4.2). In the second version, we added a WordNet semantic similarity feature. Each class was assessed separately by building a binary classifier for each one.

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

The results shown in Tab. 2 are encouraging, since the proposed 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 had a very tiny improvement on the sample with the additional WordNet predicate. On the other hand, for the other classes, the rules based only on lexico-syntactic predicates are highly precise, but its achieved recall score is lower than those ones when the WordNet predicate is used. This comparison shows 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. The 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 induced rules expressed in terms of (*number of literals)*, (*positive examples covered)*, (*negative examples covered)*, and the (*rule precision P)*:

Rule 1: #Literals = 4, PosScore = 17, NegScore = 0, P = 100%
is_a_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%
is_a_country(A):- t_wnsim_country(A, '09-10'), t_ner(A, location).
Rule 3: #Literals = 4, PosScore = 329, NegScore = 28, P = 92.0%
is_a_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, tagged "miscellaneous" by the NER, and tagged as a singular noun. In Rule 2, the high precision score of the *Country* class is mainly due to the NER that has tagged the 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.

6 Conclusion and Future Work

We have presented an ILP-based method for ontology population, which mainly relies on shallow syntactic parsing and a semantic similarity measure. Although we have achieved encouraging results so far, there are still some opportunities for improvement. Indeed, our method is currently based 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 intend to use another sentence representation formalism based on *dependency grammar*, which was proven to be more robust to linguist variations. Finally, we intend to extract instances of relations as well.

References

1. Bing Search Engine API. API Basics, `http://www.bing.com/developers/s/APIBasics.html`
2. Cimiano, P.: Ontology Learning and Population from Text: Algorithms, Evaluation and Applications. Springer, New York (2006)
3. De Raedt, L.: Inductive Logic Programming. In: Encyclopedia of Machine Learning, pp. 529–537 (2010)
4. Downey, D., et al.: Learning Text Patterns for Web Information Extraction and Assessment. In: Proceedings of the 19th National Conference on Artificial Intelligence Workshop on Adaptive Text Extraction and Mining, San Jose, USA (2004)
5. Etzioni, O., et al.: Web-Scale Information Extraction in KnowItAll. In: Proc. of the 13th International World Wide Web Conference (WWW 2004), New York, USA, pp. 100–110 (2004)
6. Finn, A.: A Multi-Level Boundary Classification Approach to Information Extraction. Phd thesis, University College Dublin (2006)
7. Lavrac, N., Dzeroski, S.: Inductive Logic Programming: Techniques and Applications. Ellis Horwood, New York (1994)
8. Miller, G.A.: WordNet: a Lexical Database for English. Communications of the ACM 38(11), 39–41 (1995)
9. Patel, A., Ramakrishnan, G., Bhattacharya, P.: Incorporating Linguistic Expertise Using ILP for Named Entity Recognition in Data Hungry Indian Languages. In: De Raedt, L. (ed.) ILP 2009. LNCS, vol. 5989, pp. 178–185. Springer, Heidelberg (2010)
10. Santos, J.: Efficient Learning and Evaluation of Complex Concepts in Inductive Logic Programming. Ph.D. Thesis, Imperial College (2010)
11. Stanford CoreNLP Tools. The Stanford Natural Language Processing Group, `http://nlp.stanford.edu/software/corenlp.shtml`
12. Wu, Z., Palmer, M.: Verb Semantics and Lexical Selection. In: Proc. of the 32nd Annual Meeting of the Association for Comp. Linguistics, New Mexico, USA, pp. 133–138 (1994)