OntoILPER: an ontology- and inductive logic programming-based system to extract entities and relations from text

Rinaldo Lima, Bernard Espinasse, Fred Freitas

OntoILPER: an Ontology- and Inductive Logic Programming-based System to Extract Entities and Relations from Text

Rinaldo Lima[1]
Bernard Espinasse[2]
Fred Freitas[3]

Abstract Named Entity Recognition (NER) and Relation Extraction (RE) are two important subtasks in Information Extraction (IE). Most of the current learning methods for NER and RE rely on supervised machine learning techniques with more accurate results for NER than RE. This paper presents OntoILPER a system for extracting entity and relation instances from unstructured texts using ontology and Inductive Logic Programming, a symbolic machine learning technique. OntoILPER uses the domain ontology and takes advantage of a higher expressive relational hypothesis space for representing examples whose structure is relevant to IE. It induces extraction rules that subsume examples of entities and relation instances from a specific graph-based model of sentence representation. Furthermore, OntoILPER enables the exploitation of the domain ontology and further background knowledge in the form of relational features. To evaluate OntoILPER, several experiments over the TREC corpus for both NER and RE tasks were conducted and the yielded results demonstrate its effectiveness in both tasks. This paper also provides a comparative assessment among OntoILPER and other NER and RE systems, showing that OntoILPER is very competitive on NER and outperforms the selected systems on RE.

Keywords Ontology-based Information Extraction ▪ Named Entity Recognition ▪ Relation Extraction ▪ Ontology Population ▪ Relational Learning ▪ Supervised Machine Learning

1 Introduction

Information Extraction (IE) consists in recognizing and extracting relevant elements such as entities and relationships from unstructured texts [44]. Two important subtasks in IE are *Named Entity Recognition* (NER) and *Relation Extraction* (RE). The former aims at finding named instances,

[1] *Informatics Center, Federal University of Pernambuco, Recife, PE, Brazil*
[2] *Aix-Marseille University, LSIS-UMR CNRS, Marseille, France*
[3] *Informatics Center, Federal University of Pernambuco, Recife, PE, Brazil*

including people´s names, locations, among others [25], whereas the latter consists of identifying relations among (named) entities in text [8]. Due to the high level of ambiguity present in natural language texts with words having multiple meanings, accurate information extraction is far from trivial. Thus, the development of efficient and robust IE systems constitutes a big challenge.

To alleviate this difficulty, *Ontology-Based Information Extraction* (OBIE) has emerged as a subfield of IE in which ontologies are used by an information extraction process and the output is usually presented through ontology [45]. Ontology is defined as an explicit specification of a shared conceptualization representing knowledge through concepts, relationships, and individuals [14]. These concepts and properties guide the extraction process in OBIE systems by providing additional background knowledge about the domain [36]. In OBIE, the extracted elements are expressed by predicates in the domain ontology, which are easy for sharing and reuse [11].

Most of the approaches to NER and RE are based on supervised machine learning techniques that build statistical classification models [25,19,7,29] and consist of the core learning components of robust, fully automatic IE systems. They use a propositional hypothesis space for representing examples, typically in the form of a vector of *attribute-value* pairs. Such approaches to NER and RE have the shortcoming of not being able to fully exploit structural information during model construction [18]. In other words, they present some difficulty in the extraction of complex relations, which demand contextual information about the involving entities. Other NER and RE methods found in the literature [35,17,42, 43] do not employ ontologies for guiding the extraction process.

The goal of this paper is to present *OntoILPER*, a novel OBIE system that attempts to overcome the limitations of the works mentioned above. OntoILPER is able to extract entity and relation instances from textual data using ontology and *Inductive Logic Programming* (ILP), a symbolic machine learning technique [12]. OntoILPER uses a domain ontology as formal background knowledge and provides a higher expressive relational hypothesis space for representing examples whose structure is relevant to both NER and RE tasks.

OntoILPER induces symbolic extraction rules in Prolog syntax that subsume examples denoting both entities and relation instances from a tailored *graph-based model* for sentence representation. We rely on the idea that the relationship between two entities in a sentence can be obtained by the (shortest) path between them according to this graph-based model that allows the construction of a well-structured hypothesis space. This hypothesis space not only integrates structural information about node properties and relations in the form of *relational features* expressing structural aspects of examples, and but can also be systematically explored by its ILP-based learning component. Therefore, during the searching and rule induction process, domain knowledge can be efficiently used as constraints to reduce search space.

Feature selection in OntoILPER is based on a careful investigation of the most effective features for NER and RE. This choice was motivated by the fact that individual features should have a clear meaning, i.e., their meaning should be easily understood by the domain expert. OntoILPER also takes into account efficiency issues by choosing a compact set of informative and relevant features, as opposed to hundreds or even thousands sparse features commonly used by kernel-based methods [25]. With this condensed set of features (Section 4.4.2), we aim at reducing learning time and avoiding redundant features.

Due to the diligent use of the domain ontology in the extraction process, OntoILPER can be seen as an OBIE system, as defined by Wimalasuriya and Dou (2009) [45]. Moreover, the extracted mentions of entities and relations are converted to ontological instances of concepts and relationships of the domain ontology. This last task is also called *Ontology Population* [36].

To evaluate OntoILPER, several experiments were conducted over the Text Retrieval Conference (TREC) benchmark corpus for NER and RE. The obtained results demonstrate the effectiveness of OntoILPER in both tasks. We also report on the results of a comparative assessment between OntoILPER and other NER and RE systems. The results showed that OntoILPER is very competitive on NER and outperforms other systems on the RE task.

The remainder of this paper is organized as follows: Section 2 describes fundamental concepts addressed in this paper. Related work on NER and RE is presented in Section 3. Section 4 presents OntoILPER, the proposed OBIE system, focusing on its principles, functional architecture, and main components. Section 5 reports on and discusses OntoILPER empirical results on the TREC corpus for NER and RE. Section 6 compares OntoILPER with other NER and RE systems. Finally, Section 7 concludes this paper and outlines future work.

2 Preliminaries

2.1 Named Entity Recognition

The aim of NER [19] is to identify named entities from natural languages texts and to classify them into a set of predefined types such as Person, Organization, Location, among others. NER is the most fundamental task in IE. The extraction of more complex structures such as relations and events depends upon accurate NER as a pre-processing step [44].

NER cannot be simply accomplished by string matching against pre-compiled lists of entities (e.g. gazetteers) because instances of a given entity type usually do not form a closed set and, therefore, any list of this kind would be incomplete [18]. In addition, the type of a named entity can be context or domain-dependent.

2.2 Relation Extraction

RE consists in *detecting* and *characterizing* semantic relations between entities in text [18]. By detecting, we refer to the task of only determining if a relation between two entities holds, whereas by characterizing, we address the classification problem of assigning a *relation type label* to a particular relation mention. Many works on RE focus on *binary relations*, i.e., relations between two entities [47,18,8,29]. Examples of such relations include *physical* (e.g. an entity is physically near another entity), and *employment/affiliation* (e.g. a person is employed by an organization).

2.3 Ontologies and Ontology Based Information Extraction

In one of the most cited definitions of ontologies, Gruber states that “an ontology is an explicit specification of a conceptualization” [14]. Therefore, ontologies comprise a body of formally represented knowledge that can be processed by a computer for a high number of tasks, such as communication and interoperation (using the ontology definitions as a shared vocabulary), business-to-business applications, intelligent agent communication and reasoning. In practical words, ontologies encompass definitions of concepts (by hierarchies), properties, relations, constraints, axioms and instances about a certain domain or universe of discourse. Moreover, they enable reuse of domain knowledge, which makes domain assumptions explicit, separating domain knowledge from the operational one.

Ontologies are implemented by formal languages, such as the OWL language [16], which is one of the most widespread expressive ontology languages. An OWL ontology is formally defined as a set of axioms α defined over the triple (N_C, N_R, N_O), where N_C is the set of *concept names* or *atomic concepts* (unary predicate symbols), N_R is the set of *roles* or *property names* (binary predicate symbols), and N_O the set of *individual names* (constants), instances of N_C and N_R.

OBIE can be defined as the process of identifying in text, relevant concepts, properties, and relations expressed by ontology [15]. Ontologies contain concepts arranged in class/sub-class hierarchies (e.g. a Country is a type of Geographical Location), relations among concepts (e.g., a *Country* has a *President*), and properties (class attributes).

An OBIE system is related to a domain ontology describing the targeted application domain, and employs an IE technique to discover both individuals (instances) for the classes and values for the properties defined by the domain ontology. One of the major components of an OBIE system is its IE module, which is guided by one or more ontologies. OBIE has the potential to automatically generate semantic contents for the Semantic Web [14], which intends to bring meaning to the current Web, creating an environment where software agents roaming from page to page can carry out sophisticated tasks [46].

2.4 Inductive Logic Programming

In one of the most cited definitions of ontologies, Gruber states, "an ontology is an explicit specification of a conceptualization" [14]. Therefore, ontologies comprise a body of formally represented knowledge that can be processed by a computer for a high number of tasks, such as communication and interoperation (using the ontology definitions as a shared vocabulary), business-to-business applications, intelligent agent communication and reasoning. In practical words, ontologies encompass definitions of concepts (by hierarchies), properties, relations, constraints, axioms and instances about a certain domain or universe of discourse. Moreover, they enable reuse of domain knowledge, which makes domain assumptions explicit, separating domain knowledge from the operational one.

The general ILP approach can be outlined more formally [30], as follows.

Given:

- a finite set E of examples, divided into positive E^+ and negative E^- examples, both expressed by non-empty sets of *ground facts* (definite clauses without variables), and
- *BK*, consisting of a finite set of extensional (ground) or intentional (with variables) Horn clauses[4].

The goal is to induce a *correct hypothesis H* (or a *theory*) composed of first-order clauses such that

- $\forall e \in E^+ : \mathrm{BK} \wedge H \models e$ (H is *complete*), and
- $\forall e \in E^- : \mathrm{BK} \wedge H \not\models e$ (H is *consistent*).

In practice, it is not always possible to find a correct hypothesis that strictly attends both criteria above, i.e., H is complete and consistent, and therefore both criteria must be relaxed. The interested reader is referred to [24,12] for more information on ILP.

[4] Horn clauses consist of first-order clauses containing at most one positive literal.

2.5 Work Assumptions

In this work, the task of identifying and extracting instances of entities and relations from textual data can be outlined as shown by the directed graph in Fig. 1. In this graph, nodes denote entities, or phrase constituents, whereas the edges represent binary relationships between entities. In OntoILPER, the identification of the types of entities and relations is cast as a classification problem.

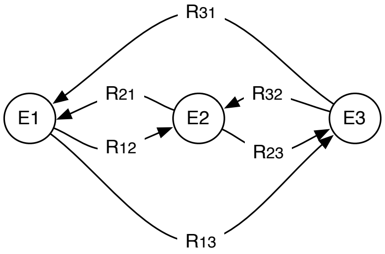


Fig. 1 Conceptual view of examples of entity and relations

Putting it more formally: given a sentence *S* formed by an ordered sequence of words *w* and entities e_i $\{e_1, e_2, ..., e_n\}$ in *S*, and a binary relation between a pair of entities mentions contained in *S*, i.e., $R_{ij} = (e_i, e_j)$, where e_i and e_j are the first and second argument of relation R_{ij} respectively, the main goal of the RE task is to correctly assign a *label* $t_i \in T_R$ to the set of all distinct relation mentions $\{R_{ij}\}$ in *S*. We also restrict the set of predefined entity and relation *labels* or *types* to T_E and T_R, respectively. The relation mentions, or relation instances R_{ij} are *directed*, i.e., $R_{ij} \neq R_{ji}$, since the evolving entities, e_i *and* e_j may play different roles in the same sentence *S*.

Other starting assumptions concern the *domain ontology* and the *input corpus*:

- Domain ontology must already exist before the entire OBIE process takes place. This ontology conveys concepts and relations relevant to the application domain;
- The entities in a sentence may be either annotated in the input corpus, or they can be recognized in the pre-processing phase. In other cases, an early classification of entity mentions (or class instances) has to be performed. An entity instance consists either of a single word or two or more consecutive words with a predefined boundary. In the last case, one can assume that nominal chunks, with their corresponding head word, characterize a *multi-word entity*.
- We only consider binary relations between entities within the same sentence. This is established by many benchmark datasets for evaluating RE systems, proposed in ACE RDC[5] shared tasks.
- We do not deal with reflexive relations.

3 Related Work

The first approach to NER and RE was based on the manual development of extraction rules [44]. Although such an approach achieves respectable effectiveness, it is usually very time-consuming. To mitigate this problem, several supervised machine-learning techniques that enable the automatic

[5] ACE (2004). Automatic Content Extraction. Relation Detection and Characterization 2004 Evaluation. http://www.itl.nist.gov/iad/mig/tests/ace/2004

construction of extraction models have been proposed [44, 18, 29]. This paper focuses on the application of supervised machine learning techniques to NER and RE. We present next related work on ILP-based systems for NER and RE. Other NER and RE systems based on supervised classification are presented in Section 6.

Ramakrishnan et al. (2008) [38] employ ILP for generating a large number of features describing named entities. Then, these features are used as input for Support Vector Machine (SVM) classifiers that build models with better performance than the best models based on handcrafted features.

Patel et al. (2010) [35] employed ILP to construct rules for extracting instances of named entities. They compared their approach of handcrafting rules by a domain expert with an ILP-based method. They found out that the development time of extraction rules using ILP was reduced by a factor of 240, and the ILP-based method provided a complete and consistent view of all the relevant patterns at the level of abstraction specified by the domain expert.

Horvath and colleagues (2009) [17] propose an interesting RE system that is similar to ours because they also uses dependency trees [10] as relational structures denoting binary relations between two entities. The authors assume a partial order on the set of unary predicates defined as a hierarchy of words, e.g., the predicate Person(X) is more general than the predicate Physicist(X). Their ILP-based approach is based on the notion of the Least General Generalization from [37]. Similar to our work, their approach generates a set of rules in the form of non-recursive Horn clauses satisfying some criteria of consistency, i.e., all the rules must cover a minimum number of positive examples, while accepting some negative examples as noise. Then, the learned rules are employed for generating a binary vector of attributes for each example. The resultant vectors are finally used for training a SVM classifier.

Seneviratne and Ranasinghe (2011) [42] propose an IE multi-agent system that relies on the ILP framework for learning extraction rules of binary relations. In this multi-agent system, one ILP-based agent is responsible for rule learning, while another one employs the learned rules on new documents to extract new relation instances. In this system, syntactical dependencies among the words in a sentence provide the background information that defines and constrains the search space. All of the learned relations are expressed as binary predicates with two entity arguments. The authors evaluated their system on 13 Wikipedia web pages about birds.

Smole et al. (2012) [43] propose a spatial data recommendation service in which an ILP-based component learns rules that extract relations from definitions of geographic entities in Slovene language. Their ILP-based component is rooted on the classical Progol ILP system [31]. They focus on the extraction of the five most frequent relations ("isA", "isLocated", "hasPurpose", "isResultOf", and "hasParts") found in a corpus composed of 1,308 definitions of spatial entities. A major drawback of their method is that the manual development of the chunk rules is time-consuming, and not scalable.

All of the surveyed ILP-based systems either perform NER or RE, and most of them assume that NER is already solved, i.e., they take profit of the pre-annotated named entities from the input corpus. This assumption limits their application to other corpora in which none of the named entities are already indicated. On the contrary, the proposed method OntoILPER can effectively perform both NER and RE tasks, as demonstrated by the experimental assessment provided in Section 6. Moreover, none of the above works can be considered as OBIE systems because they do not employ ontologies to guide the extraction process. Contrarily, OntoILPER offers all the benefits of the synergy between the ILP-based learner and the domain ontology: the former is able to generate symbolic extraction rules, while the latter can be fully exploited by the OBIE process for generalization purposes.

4 An Ontology and Inductive Logic Programming-based System for Entity and Relation Extraction

This section presents OntoILPER, an OBIE system that employs a supervised learning approach to extract entities and relations instances from free texts. We first present an overview of the OntoILPER extraction process, which uses the ProGolem ILP learner and exploits the domain ontology. Then the OntoILPER architecture and its main components are introduced in detail.

4.1 OntoILPER Overview

4.1.1 Extraction Process in OntoILPER

The extraction process is performed by rules induced by an ILP-based component, which is guided by the domain ontology. In the end of the extraction process, entity and relation instances extracted by OntoILPER populate the domain ontology (Fig. 2).

OntoILPER is rooted on an ILP-based learning module as the core component for building classification models (Fig. 2). In addition, the domain ontology integration into the IE process is of paramount importance, and the reasons for their use are twofold: (i) ontologies can capture knowledge about a given domain of interest, and (ii) they can be used for processing both information and semantic contents of textual sources.

Fig. 2 shows an overview of the processing flow in OntoILPER:

- The *Text Preprocessing* step annotates the input corpus with linguistic-based annotations producing rich annotated documents (Section 4.3);
- After that, the annotated documents (in XML format) are passed as input to the *Background Knowledge Generation* step (Section 4.4), which takes profit of the domain ontology. This ontology provides valuable information, by means of TBox axioms and ABox assertions[6], as BK that guides the entire IE process. This ontological BK allows OntoILPER to be an OBIE system more flexible and adaptive [45].
- Next, in the *Extraction Rule Learning* step (Section 4.5), a general ILP system, provided with a proper BK, induces symbolic rules expressed as a set of *logical programs*, or predicates in Prolog.
- Using this set of extraction rules, the *Instances Extraction* step (Section 4.6) applies them on unseen examples, which are, in turn, used for populating the domain ontology with class and relation instances.

[6] In an ontology, TBox statements describe a system in terms of a controlled vocabulary, or a set of classes and properties; whereas ABox is the assertional component, i.e., TBox-compliant statements about that vocabulary.

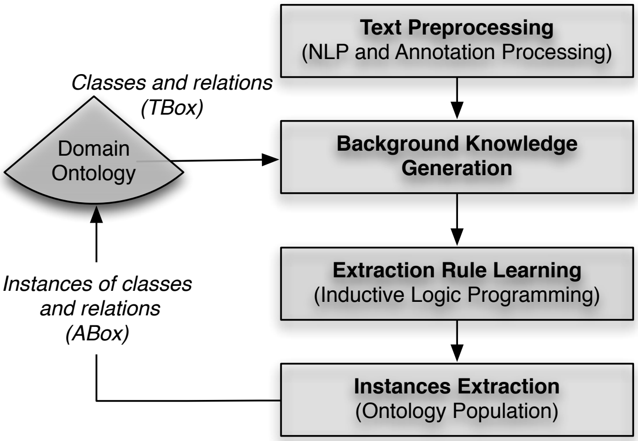


Fig. 2 Conceptual view of examples of entity and relations

4.1.2 ProGolem ILP Learner

The rule learning component in OntoILPER is based on ProGolem [41,33], an efficient bottom-up ILP learner capable of learning complex non-determinate concepts, i.e., target predicates. ProGolem combines the most-specific clause construction of Progol [31] with the bottom-up control strategy of Golem [32]. ProGolem is available as one of the ILP systems integrated into GILPS (General Inductive Logic Programming System) proposed in [41].

An advantage of ProGolem over classical top-down ILP systems, like Aleph[7], resides on the fact that it is able to learn long, non-determinate target concepts or predicates. Target predicate complexity is problem dependent and usually unknown a priori. For instance, many real-world applications, including the learning of chemical properties from atom and bond descriptions, require non-determinate BK. The basic ProGolem covering set algorithm is given below:

ProGolem Covering Set Algorithm

```
Input:  Examples $E$, background knowledge $B$, mode declarations $M$
Output: Theory $T$, a set of definite clauses or rules
$T = \{\}$
$E^+$ = all positive examples in $E$
while $E^+$ contains unseen positive examples do
    $e$ = first unseen positive example from $E+$
    Mark $e$ as seen
    $C$  = best_armg($e$, $E$, $M$)
    $C_e$ = negative_based_reduction($C$, $E$)
    if $C_e$ has positive score then
        $T$  := $T \cup C_e$
        $E^+_c$ := all positive examples that clause $C_e$ covers
        $E^+$ := $E^+ - E^+_c$
    end if
end while
return $T$
```

[7] The Aleph Manual. http://www.cs.ox.ac.uk/activities/machinelearning/Aleph/aleph

ProGolem is based on the covering set approach to construct a theory consisting of more than one clause. At each iteration of the covering set algorithm, ProGolem repeatedly constructs clauses using the beam-search iterated ARMG (*asymmetric relative minimal generalizations*) algorithm [33] (line 6) to select the highest-scoring armg with respect to an initial seed example *e* (line 4). Then, the clauses yielded by the beam-search iterated armg algorithm need to be further generalized. ProGolem employs a negative-based reduction algorithm (line 7) to prune literals from the body of the current clause (*C*) that are non-essential. A non-essential literal is a literal that, if removed, does not change the negative coverage of the clause. Then, if the current clause C_e achieves an expected accuracy score (line 8), it is added to the theory *T* and all the examples covered by it are removed from the set of training examples E^+. A detailed description of armg and negative-based reduction algorithms can be found in (Santos, 2010) [41].

4.1.3 Exploiting Domain Ontology In OntoILPER

OntoILPER enhances related work on ILP applied to NER and RE (Section 3), by taking profit of ontological elements, such as TBox and ABox [3], as BK for its ILP-based learning component that detects and classifies semantic relations between entities. Indeed, such an integration of ontologies into the IE process has produced positive results [11,21,45]. The rationale here is that not only ontologies can capture knowledge about a domain of interest, but can also be used in applications that need to process information content, as well as to reason about it, instead of only presenting information to users.

In OntoILPER, as reported in [27], the domain ontology guides the BK generation process by defining the level of abstraction of the BK predicate arguments that will be the building blocks (literals) of final induced rules. In other words, classes, data/object properties, taxonomical, and non-taxonomical relations are used for rule creation and generalization. Thus, TBox axioms of the domain ontology (class and property labels, data/object properties, is-a relationships, and domain/range of non-taxonomical relations) are taken into account during the *BK Generation* step (Section 4.4) in OntoILPER.

Furthermore, such an integration of domain ontologies in OntoILPER is in accordance with the first three levels of ontological knowledge used by most of the state-of-the-art OBIE systems, as discussed in [21]:

- At the first level, the ontological resources used by OntoILPER consist of domain entities (e.g., person, location) and their synonyms or *co-referents*, and words classes (keywords, terms, descriptors of entities). These resources are mainly applied by OntoILPER for NER [28];
- At the second level, semantic resources, e.g., domain entities organized in conceptual hierarchies, can be exploited by the NER process for generalizing/specializing extraction rules [27];
- At the third level, concepts properties and relations between concepts of the domain ontology are exploited, as they provide a richer extraction template for the entire IE process [28].

In the rest of this section, the main components of OntoILPER implementation are presented.

4.2 OntoILPER Architecture

The IE process in OntoILPER is carried out in two distinct phases. First, a set of symbolic extraction rules (classification model) is induced from an annotated corpus converted to a BK base. This corresponds to the *Learning Phase* in Fig. 3 which is performed by the *ILP Rule Learning* component. Then, in the *Application Phase*, the previous set of induced rules is applied to extract instances of entities and relations from new annotated documents. This is performed by the *Rules Application* component in Fig. 3. The extracted instances are used by the *Ontology Population* component, which populates the *domain ontology*. The domain ontology also guides the IE process by providing information about its classes and relationships to the *Background Knowledge Generation* component. In both phases, several natural language processing (NLP) techniques are executed in pipeline by the *Natural Language Processing* component, which produces a fully annotated version of input corpus (*Annotated Corpus*). An automatic generation and representation of the examples follow the corpus annotation by the *Background Knowledge Generation* component.

Fig. 3 depicts the OntoILPER architecture with its components (gray boxes) performing each one a specific task in the global extraction process. In the remainder of this section, the OntoILPER components are described in detail.

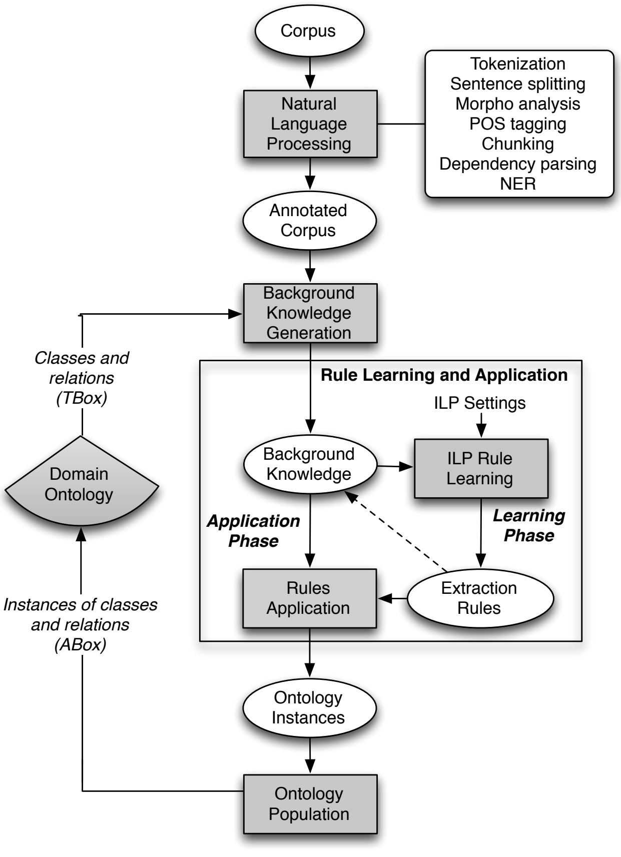


Fig. 3 Overview of the components in OntoILPER Implementation

4.3 Natural Language Processing Component

The *Natural Language Processing* component performs the automatic annotation of the input corpus. The output of the annotation is composed of both morphosyntactical and semantic aspects present in natural language texts. For carrying out this annotation, we integrated in this component two NLP tools: *Stanford CoreNLP*[8] and *OpenNLP*[9]. The former performs the following general-purpose NLP subtasks: *sentence splitting*, *tokenization*, *Part-of-Speech (POS) tagging*, *lemmatization*, *NER*, and *dependency parsing* [10] while the latter is responsible for the *chunking analysis*. In general, these NLP subtasks are performed in pipeline mode, starting with simpler analysis (sentence splitting and tokenization) whose output results are used as input by the more complex subtasks such as POS tagging and dependency parsing. Fig. 4 depicts the NLP pipeline developed in OntoILPER.

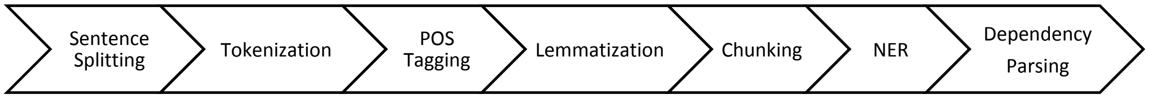


Fig. 4 Pipeline of NLP-subtasks performed in OntoILPER system

4.4 Background Knowledge Generation Component

After the Natural Language Processing step, OntoILPER carry out the critical task of identifying, extracting, and appropriately representing relevant BK. This task is performed by the *Background Knowledge Generation* component.

In propositional machine learning, the incorporation of expert knowledge about a given domain is usually done by introducing new features, whose values are computed from other attributes values. In most of related work on IE, and in RE in particular, expert knowledge is defined by adding new columns as function of other data columns. This is particularly evident in kernel-based methods for RE [25,2] in which the structural representation of sentence parsing trees is converted to features in a vector-based representation. This conversion is usually performed by applying similarity functions, on the sentence parsing trees. As a result, part of the relational knowledge, i.e., the structural information is lost in this transformation process [12,18].

Another limitation of the vector representation of examples is the serious restriction of having a unique representation format for all the examples, i.e., one feature is created for each element in the domain, and the same feature is used for characterizing all examples under consideration. In general, this results in a very sparse data table because most of the attributes will contain null values, due to the difference among the examples. Yet, Brown and Kros (2003) [5] pointed out that this data sparseness problem is even more critical when deep knowledge is explored, which can cause serious problems for propositional machine learning algorithms.

By contrast, in OntoILPER, each example is represented independently of the others. Thus, the data sparseness problem for representing the examples is highly reduced [12]. Thereby, the above limitations are alleviated by employing first-order formalism, for representing both BK and examples. This enables that several sources of information, either propositional or relational in

[8] Stanford CoreNLP Tools. http://nlp.stanford.edu/software/corenlp.shtml.

[9] Apache OpenNLP. The Apache Software Foundation. http://opennlp.apache.org

nature, to be effectively represented without the drawbacks of the propositional approaches mentioned above. Moreover, we argue that the ability to take into consideration relational BK and the expressive power of the language of the discovered patterns are distinctive features of OntoILPER. In short, we want to test the working hypothesis that, by using the richer ILP formalism, we should be able to directly represent a vast amount of BK extracted from ontologies, semantic resources, and shallow and deep analysis originated from NLP tools.

4.4.1 Relational Modeling of Sentences and Examples

OntoILPER relies on a graph-based model representation of sentences and examples first introduced in [28]. In this model, a binary relationship can be specified between concepts. All of these binary relationships, as well as entity attributes, can similarly be described by the Entity-Relationship (E-R) diagram depicted in Fig. 5. From the perspective of this E-R data model, *entity* attributes denotes predicates defining *properties*, whereas *relationships* between entities correspond to *structural* predicates. We argue that when learning about objects in relational domains, feature construction should be guided by the structure of the examples.

The model in Fig. 5 represents a collection of binary relations, and their arguments can be enriched with additional constraints on the types of the arguments. These additional binary relations are used by the ILP-based induction-learning component responsible to link terms in a sentence with classes and relations from domain ontology. For example, if the predicate to be learned is *read (X, Y),* or putting it as ontological terms, the object property *read(X, Y)*, then the first argument *X* must be an instance of the *Person* class, whereas the second one *Y* must be an instance of the *Publication* class in the domain ontology. In sum, instances of classes and relations can be viewed, respectively, as nodes and edges in our model. Each node can have many attributes, e.g., the ontological class label, which it belongs to.

Fig. 6 depicts an instantiation of the model shown in Fig. 5 corresponding to the sentence: "*Myron Kandel at the Newsdesk CNNfn in New York*". The graph instance is composed of a set of binary relations or predicates, including *det(Newsdesk, the), nn(Newsdesk, CNNfn), prep_in(Myron-Kandel, New-York), nextToken(the, Newsdesk).*These sentence annotations were obtained by the integration of: (i) a dependency graph with *collapsed dependencies* [10] (e.g. *prep_on*) according to the Stanford dependency parser, (ii) a *chunking analysis* (head tokens in bold), (iii) the sequencing of tokens in a sentence (*NextToken* edges), (iv) *morpho-syntactic features* as nodes attributes (arrows in gray colour), and (v) semantic attributes, such as named entities. The interested reader can refer to [28] for more information about OntoILPER sentence annotations.

Thus, the task of identifying the labels of candidate classes and relations instances is defined as the *target predicate* in our learning problem formulation. We learn such target predicates as a combination of several sentence elements given by the graph-based model for sentence representation described above.

Most previous work in NER [19,25] and RE [13,39,18,29] have only considered a vector of attribute-value pairs as features (propositional features) derived from input text data. Instead, OntoILPER relies on a first-order logic representation of examples, which provides much richer representation formalism, allowing classification of objects whose structure is relevant to the classification task [12]. Other complex combinations of features, such as statistical ratios were not considered in OntoILPER feature selection, mainly due to less effective results demonstrated by previous work on RE [18].

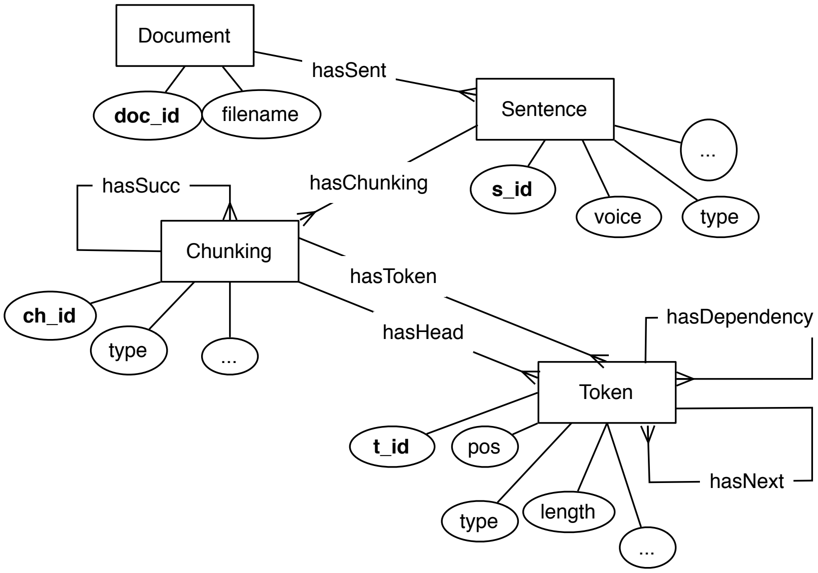


Fig. 5 Entity-Relationship model for sentence representation in OntoILPER

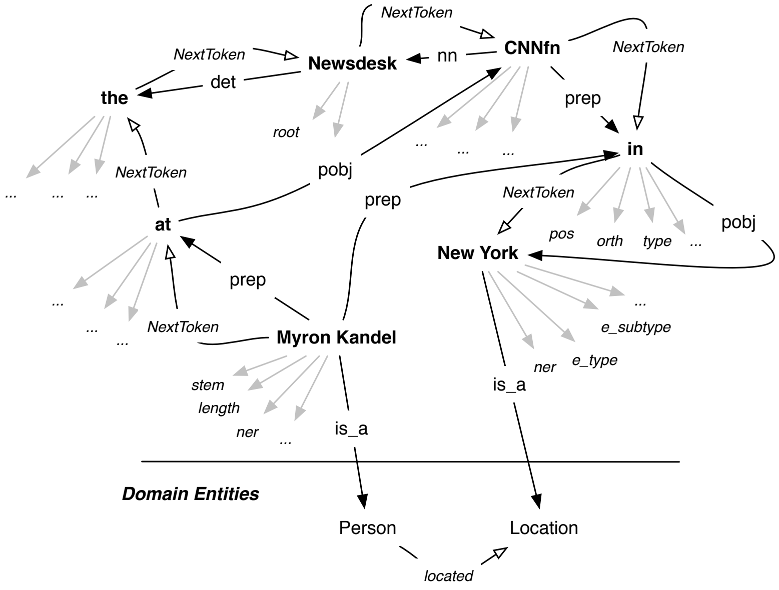


Fig. 6 Instantiation of the graph-based model for the sentence: "Myron Kandel at the Newsdesk CNNfn in New York"

4.4.2 Structural and Property Features

Previous research on IE has shown that morphological analysis and syntactic parsing of natural language texts can provide very useful features for many IE subtasks, including NER and RE [20, 25, 19]. In this work, we explore the features listed in Tab. 1, which constitute the main elements of BK explored in our approach.

Table 1 Prolog predicates describing the token "Myron" (t_1)

Group	Prolog Predicates	Meaning

Corpus entities	*doc(d_1)*
	sent(s_1)
	chunk(ck_1)
	token (t_1)
		d_1 is a document identifier
		s_1 is a sentence identifier
		ck_1 is a chunk identifier
		t_1 is a token identifier

Lexical features	*t_stem (t_1, "Myron")* *t_length (t_1, 5)* *t_orth (t_1, upperInit)* *t_morph_type(t_1, word)*	token *t_1* stemming is "Myron" token *t_1* has length of 5 characters token *t_1* begins with an initial uppercase letter token *t_1* is has the morphological type *word*
Syntactical features		
POS and POS n-grams	*t_pos (t_1, nnp)* *t_gpos(t_1,nn)* *t_bigPosBef (t_1, ...)* *t_bigPosAft (t_1, vbz-vbg)* *t_trigPosBef (t_1, ...)* *t_trigPosAft (t_1,vbz-vbg-dt)*	token *t_1* is a singular proper noun token *t_1* is a canonical noun (no plurals) POS tag bigram before token *t_1* POS tag bigram after token *t_1* POS tag trigram before token *t_1* POS tag trigram after token *t_1*

Chunking analysis	*ck_hasHead(ck_1, t_1)*
	ck_hasType(ck_1, np)
	t_isHeadNP(t_1)
	ck_dist_to_root(ck_n, near)
	t_ck_tag_type(t_1, np)
		ck_1 has *t_1* as its token head
		ck_1 is a nominal chunk
		t_1 is the head token of a nominal chunk
		ck_n is near the main verb of the sentence
		token *t_1* has the chunking type *np*

Semantic features	*t_ner(t_1, person)*	*t_1* was annotated by the NER as PERSON entity

Predefined corpus annotation types	*t_type(t_1, person)*
	t_subtype(t_1, none)
	t_mtype(t_1, name)
		t_1 has the PERSON corpus type
		t_1 has no subtype
		t_1 is a named proper noun

Structural features	*t_next (t_1, t_2)* *t_next_head (t_1, t_3)* *ck_hasToken(ck_1, t_1)* *ck_hasSucc(ck_1, ck_2)* *t_hasDep (nn, t_2, t_1)* *t_root (t_n)*	token *t_1* is followed by the token *t_2* head token *t_1* is followed by head token *t_3* *t_1* is one the tokens in the chunk *ck_1* *ck_1* is followed by the chunk *ck_2* *t_1* has a multi-word dependency with *t_2* *t_n* is the root (main verb) of the dependency tree

These features provide a suitable hypothesis space, describing each semantic unit in the corpus. In OntoILPER, we distinguish four main groups of features:

i. **Lexical features** which concern word, lemma, length, and general morphological type information.

ii. **Syntactic features** which consist of *word POS tags*; *head word* of nominal, prepositional or verbal chunk; *bi-grams* and *tri-grams* of consecutive POS tags of words as they appear in the sentence[10]; c*hunking* features that segment sentences into noun, prepositional, and verb groups providing *chunk type* information (nominal, verbal or prepositional), *chunk head word*, and its *relative position* to the main verb of the sentence.

[10] We have also experimented with 4-grams, but bi-grams and tri-grams achieved better results in our preliminary experiments

iii. **Semantic features** include the recognized named entities in the text pre-processing phase, and any of the additional *entity mention* feature provided by the input corpus. For instance, in the TREC dataset, each annotated entity has its *entity mention type* (person, organization, or location).

iv. **Structural features** consist of the structural elements connecting all the other features in the graph-based model for sentence representation. They denote (i) the *sequencing of tokens* which preserves the token order in the input sentence; (ii) the *part-whole* relation between tokens and the chunk containing them, i.e., the tokens are grouped in its corresponding chunk; (iii) the sequencing of chunks is represented by edges between their head tokens; and (iv) the *grammatical dependency* between two tokens in a sentence according the typed dependencies between words given by the Stanford dependency parser.

As Prolog is also employed as the representation language of the examples in OntoILPER, domain entities, relations, and all the types of features mentioned above are converted to the corresponding Prolog predicates. We illustrate the complete set of the features introduced above with the instance of the *Person* class, "*Myron*" in Tab. 1.

For most of the predicates in Tab. 1, the first-order logic representation of the features is straightforward: an *unary predicate* in Prolog denotes identifiers, whereas *binary predicates* correspond to features (attribute-value pairs), and relations, e.g., $rel(arg_1, arg_2)$. Differently from other machine learning approaches that employ feature vectors for representing *context windows* (*n* tokens on the right/left of a given word *w* in a sentence), we employ the binary predicate *next/2* which relates one token to its immediate successor in a sentence, as shown in Tab. 1.

4.4.3 User-defined Background Knowledge

In OntoILPER, the user can specify any form of additional declarative knowledge to help the rule induction process. The predicates displayed in Fig. 7 were also integrated as BK into OntoILPER.

```
% Token length type definition
length_type(short). length_type(medium). length_type(long).

tok_length(T, short) :- token(T), t_length(T, X), X  =< 5.
tok_length(T, medium):- token(T), t_length(T, X), X > 5, X =< 15.
tok_length(T, long)  :- token(T), t_length(T, X), X > 15.


%  Chunking distance to the main verb
ck_dist_root(CK, near):-  ck_posRelPred(CK, X), X >= -3, X  =< 3.
ck_dist_root(CK, far) :-  ck_posRelPred(CK, X), ( ( X >= -8, X < -3) ;
                          (X > 3,  X =< 8) ).
ck_dist_root(CK, very_far):-  ck_posRelPred(CK, X),(( X < -8); (X > 8)).
```

Fig. 7 Intentional predicates added to the original BK in OntoILPER

These user-defined predicates consist in two intentional predicates that discretize numerical features, including *token length/2* and *chunk dist_to_root/2*: the first predicate categorizes the token length as *short, medium or long size*, while the second discretizes the distance (in number of tokens) between a chunk and the main verb (*root*) of the sentence. Such user-defined predicates intend to enable better rule generalizations.

4.5 Rule-Learning Component

The rule-learning component in OntoILPER integrates the ILP general learner ProGolem for inducing extraction rules. It relies on the *predictive setting* of the ILP that consists in using ILP for constructing classification models expressed as symbolic rules able to distinguish between positive and negative examples. In addition, we impose some restrictions over the induced extraction rules:

- They have to reflect the BK in terms of both structural and property features defined by our graph-based model of sentence representation describe in Section 4.4.
- They must be well-formed with respect to the *linkedness* of the variables in the rules, i.e., it must exists a chain of literals connecting the input variables in the head of a rule to the variables in the body of the rules [41].
- Their qualitative aspects, expressed by pertinent linguist patterns have to be easily understandable by the domain expert.

4.5.1 Rule-Learning Scenarios

A special feature of the OntoILPER learning component consists in its capability to employ rules learned in a previous learning step (*iteration i*) as additional BK predicates at a posterior learning step (*iteration i + 1*). Roth & Yih (2007) [39] call this capability as the *pipeline method* for model generation. Fig. 8 depicts the flows of information exchanged between the *BK Generation Component* and the *Rule Learning* component in OntoILPER corresponding to two distinct learning setting that can produce *composite* rules.

The first learning setting, indicated by the edge *A* in Fig. 8, denotes the most common RE shared tasks, including the ones proposed by ACE RDC, in which all the entity labels (BK) are already provided by the training dataset. For example, a pair of entities with its labels, denoting the two arguments of a target relation, is given to a relation learner. However, we should emphasize that this learning setting may not reflect a real world need for information extraction, as it is expected that the labels of the entity arguments of a relation are already provided by the training dataset.

The second setting, illustrated by edge B starting from the Rule Learning component and pointing to the BK component, denotes a possible more realistic IE scenario in which the relation classifier does not know the labels of its entity arguments, for example. In this case, the Rule Learning component should identify the labels of the argument entities first, which implies in generating extraction rules for classifying the two argument entities. Then, the previous extraction rules are used as complementary background information by the BK Generation component. In conclusion, the two steps displayed in Fig. 8 can be executed in loop a number of times. This allows that discovered rules in a previous iteration *i* to be used or compose new rules learned in a posterior iteration *i +1*.

A composite rule for the target relation *live_in* learned according to the information flow denoted by the edge *B* is illustrated next:

live_in(A,B):- t_pos(A,nn),*per(A),*t_hasDep(amod,B,C),t_next(C,B),*loc(B),*t_isHeadNP(B).

The rule above means that "A" lives in "B", if "A" is an entity instance classified as "Person", and the head token of the nominal chunk "B" is classified as an instance of *Location* class. The other literals (predicates) in this rule give additional contextual restrictions on the relation arguments. In this example, the unary predicates *per(A)* and *loc(B)* are learned first in an iteration of the learning process, and then used as BK for learning the target relation *live_in* in the next learning iteration.

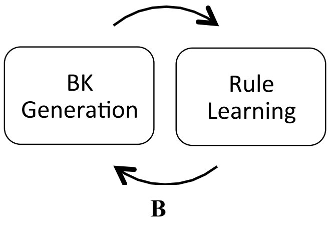


Fig. 8 Flow of information during the generation of composite rules in OntoILPER

4.5.2 Extraction Models

According to [39], there are three different types of extraction models for classifying instances of entities and relations (Fig. 9).

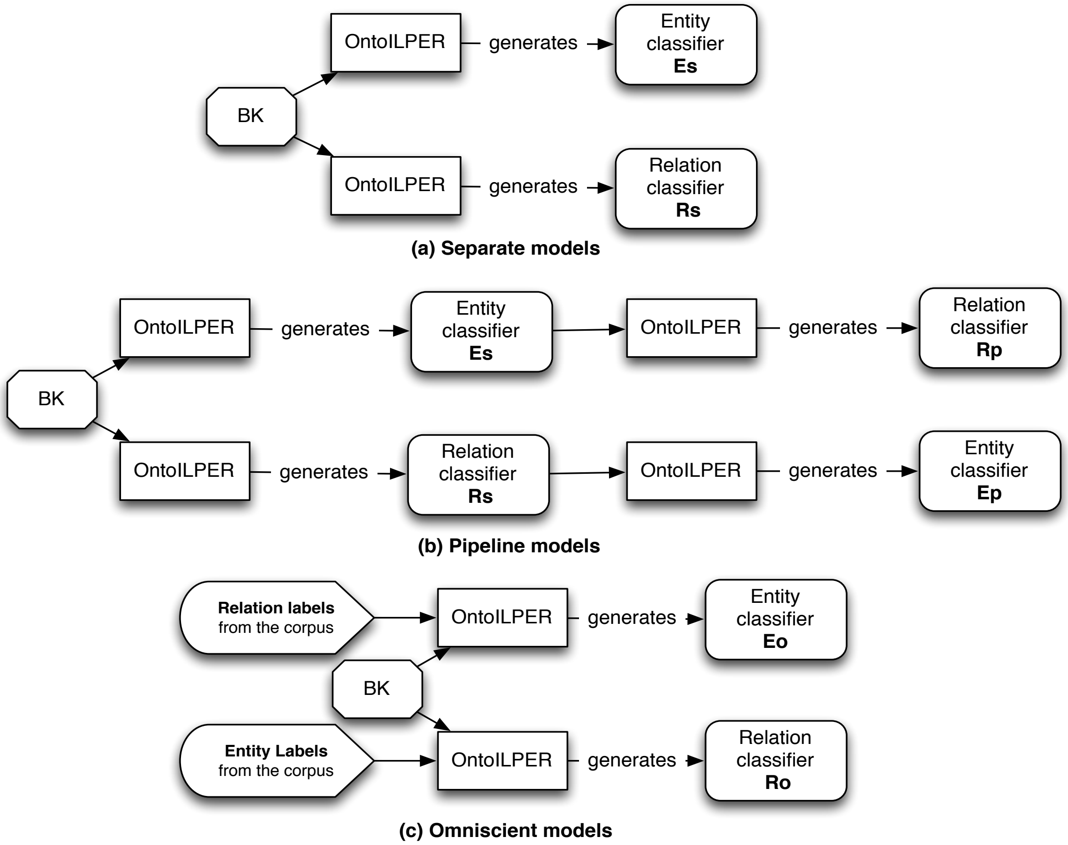


Fig. 9 Entity and relation extraction models in OntoILPER

i. *Separate Models*. In the separate model construction, both the *Separate entity* classifier (E_S) and the *Separate relation* classifier (R_S) are constructed using only the BK as input. This characterizes the most realistic scenario for the majority of information extraction needs. Fig. 9(a) illustrates the way these classification models are generated.

ii. *Pipeline Models*. The *Pipeline entity* classifier, denoted by E_P, is obtained by first building a separate relation model R_S as it is done in Fig. 9 (a). Then, another OntoILPER instance uses the previous R_S classifier for constructing the final E_P classifier. Inversely, the *Pipeline relation* classifier (R_P) is obtained by first building a separate entity model E_S. Then, another OntoILPER instance employs this previous E_S model for constructing the final R_P classifier. The E_P and R_P classifier construction processes are displayed in Fig. 9(b) using two OntoILPER instances in each case.

iii. *Omniscient Models.* For building the *Entity omniscient* model (E_O), all the relation labels are taken as input from the annotated corpus. Analogously, for generating the *Relation omniscient* model (R_O), all the entity labels provided by the annotated corpus are used as input. Fig. 9(c) illustrates the construction process of these omniscient models in OntoILPER.

4.5.3 Generating Extraction Models

During learning in OntoILPER, the search for rules in the hypothesis space that ProGolem[11] has to perform is computational demanding because it is necessary to test each candidate rules with respect to the positive and negative examples. Indeed, this is the most expensive task in the entire learning process.

To speed up learning, ProGolem intelligently goes through the hypothesis space, taking advantage of its particular structure, only exploring the portions of the hypothesis space containing high accuracy extraction rules. For that, the hypothesis space is structured by a *quasi-order* relation between two hypotheses, which allows an efficient navigation among the candidate rules [41].

More concretely, ProGolem employs (i) *mode declarations*, for delimiting and biasing the possibly huge hypothesis search space; and (ii) *parameter settings*, for modifying its default rule construction process:

- **Mode Declarations**. Mode declaration [31] is one of the most known types of bias employed by ILP systems, including ProGolem, for defining syntactical constrains on the form of the valid rules. There are two types of mode declarations in ProGolem: *head* and *body*. Mode head declarations (*modeh*) defines the target predicate, the head of a valid rule that the ILP system has to induce, whereas mode body declarations (*modeb*) determine the literals (or ground atoms) which may appear in the body part of the rule. Mode declarations also impose restrictions on the types of the variables used as arguments of a predicate. Such types are simply declared by Prolog predicates of the form *type(value)*, e.g., *token(t_1)* and *chunck(ck_1)* which are used as identifiers of tokens and chunks, respectively. The mode declarations corresponding to some of the features in Table 1 are listed below:

[11] ProGolem ILP system runs on the YAP Prolog (http://www.dcc.fc.up.pt/~vsc/Yap)

```
:- modeh(1, work_for(+token, +token)).  % Head or target predicate
:- modeb(*, t_hasDep(#dep, +token, -token)).   % Structural
:- modeb(*, t_next(+token, -token)).
:- modeb(*, ck_has_tokens(-chunk, +token)).     % Chunking
:- modeb(*, ck_hasSucc(+chunk, -chunk)).
:- modeb(*, t_pos(+token,#postag)).             % Syntactic (POS)
:- modeb(*, t_trigPosBef(+token,#trigposbef)).
:- modeb(*, ck_hasType(+chunk, #ck_tag)).      % Chunking-related
:- modeb(*, ck_hasHead(+chunk, #token)).
:- modeb(*, t_ner(+token,#ner)).               % Semantic NER
```

At the beginning of a mode declaration definition the symbol "1" means that only one instance of the accompanying predicate can appear in the rule, while "*" means that any number of accompanying predicate can appear in the body part of the rule. For instance, the first mode declaration above denotes the head of the rule *work_for*, i.e., only one instance of the target predicate *work_for (token, token)* is allowed in the rule, denoting a binary relation between two tokens. The third mode declaration denotes the predicate *t_next(token,token)* that links a token to the next one in a sentence. Finally, the symbols "+" and "-" restrict the way a predicate (or literal) is "concatenated" with the following one during rule learning. The interested reader is refer to [31,41] for more information about mode declarations in ILP.

- **Parameter Setting**. In its learning stage, users are allowed to customize the learning task by choosing the combination of BK layers (structural, morphosyntactical, and semantic) that is more appropriate to their IE needs. In addition, users may directly intervene in the learning task by defining the ProGolem parameters summarized in Tab. 2. Among them, for example, the *noise* parameter is related to the well-known problem in machine learning: real-world databases very often contain noisy data, i.e., erroneous or incomplete instances. Noisy data can also cause overfitting, a major issue for all machine-learning techniques. Particularly for ILP, overfitting can cause the induction of very specific extraction rules that only memorizes the examples instead of generalizing them. As a result, the size of the final extraction models may increase in function of the training set.

Table 2 The most important parameters used by ProGolem ILP system in training

Parameter	Description
Evaluation function	evaluation function for scoring a clause (coverage, precision, recall, compression ratio, etc.)
Variable depth (i)	It determines the number of layers of new variables to consider during the construction of the bottom clause.
Minimum precision or accuracy	a real number [0-1] specifying the minimum precision (or accuracy) a candidate extraction rule has to have.
Minimum number of positive examples	Minimum number of positive examples a clause has to cover
Noise tolerance	It allows the induced extraction rules to be more tolerant to noisy examples in the training data, since to obtain consistent extraction rules that covers no negative example is practically impossible due to common noisy training data.

4.5.4 Example of Induced Rules

Two induced rules for *part_whole* relation are illustrated next.

Rule 1:

#Literals = 4, Positive Score = 90; Negative Score = 1; Precision = 98.9%

part_whole(A,B):- t_gpos(A,nn), t_next(A,B), t_subtype(B,state-or-province)

Rule 2:

#Literals = 5, Positive Score = 31; Negative Score = 7; Precision = 77.4%

part_whole(A,B):- t_next(A,B), t_pos(A,nnp), t_ne_type(B,gpl),t_subtype(A,pop-center)

The above rules were evaluated using the scoring function *compression ratio*: *(positive examples - negative examples)/clause length*. We set other parameters as well: *i* = 3, *minimum precision* = 0.0, *minimum positive examples* = 5, and *noise* = 20%, leaving the remaining parameters with their default values. These rules are expressed in terms of *number of literals*, *positive examples covered*, *negative examples covered*, and *rule precision P:*

- *Rule 1* classifies an instance of the *Part-Whole* relation. Its high precision (*P = 98.9*) is due to the high number of sentences containing *two adjacent tokens (*or *phrases)* where the first (*A*) is a *noun,* and the second one (*B*) is tagged with respect to the domain ontology as an instance of the *"State-or-Provence"* class. This rules highlights that places (*A*), such as cities, are located, or are part of either a *State* or *Provence*.
- *Rule 2* is very similar to Rule 1, in which the entity instances (tokens variables *A* and *B)* are also adjacent. Token *A* is a *proper noun* and an instance of the *Geographical Political Location* (*GPL*) class, while token *B* is mapped to the *Population-Center* class in the domain ontology.

4.6 Ontology Population Component

The Ontology Population component applies the final set of rules on the Prolog knowledge base generated from new documents similar to the ones used in training. As a result, new instances of entities (or classes) and relations are extracted, and they can be finally integrated into the domain ontology. For instance, the extracted instances of the two classes and the relation present in the sentence "*Myron Kandel at the Newsdesk CNNfn in New York*" could be saved into the domain ontology:

Person("Myron Kandel") // "Myron Kandel" is an instance of the *Person* class
Location("New York") // "New York" is an instance of the *Location* Class
is_located("Myron Kandel", "New York") // "Myron Kandel" and "New York" are related

Finally, before converting the Prolog predicates as OWL facts in the domain ontology, OntoILPER performs a redundancy checking step, to avoid repeated instances in the domain ontology.

It is worth mentioning that ontology population systems are closely related to OBIE systems, since the latter provide mechanisms to link instances, represented as textual information, with elements of the ontology. Thus, every OBIE system can be regarded as an ontology population system, since it is able to incorporate the extracted instances into the ontology.

5 Experimental Evaluations

The main goal of the experiments reported in this section is to investigate the effectiveness of OntoILPER. First, we present the TREC dataset used to evaluate OntoILPER performance on NER and RE. Then, the strategy for generating negative examples, the evaluation metrics, and the parameters settings are presented. Finally, this section reports on and discusses the empirical results.

5.1 TREC Dataset

The experiments reported here based on the TREC dataset[12] for NER and RE proposed by Roth and Yih (2004) [40]. This dataset was selected because it has been used in previous papers, which enables the comparative assessment presented in Section 6. The TREC dataset has been annotated with named entities and relation labels, containing 1,441 sentences with 5,349 entities, namely: 1,691 people, 1,968 locations, 984 organizations, and 706 miscellaneous names. Each one of the 1,441 sentences has at least one active relation. Some examples of the binary relations in this corpus are illustrated in Tab. 3, as well as their frequency distribution. This table also shows examples of each relation with its entity labels and argument types. The great majority of the candidate binary relations are negative which results in an unbalanced distribution between positive and negative examples. Fig. 10 depicts the domain ontology created for storing the instances extracted by OntoILPER. This ontology also represents the domain of the TREC corpus on news articles.

Table 3 Binary relation and their arguments types

Relation	Arg-1	Arg-2	Example	# Relations
located_in	LOC	LOC	(Toledo, Ohio)	405
work_for	PER	ORG	(Winter, Court)	401
orgBased_in	ORG	LOC	(HP, Palo Alto)	452
live_in	PER	LOC	(Tvazir, Israel)	521
kill	PER	PER	(Oswald, JFK)	268

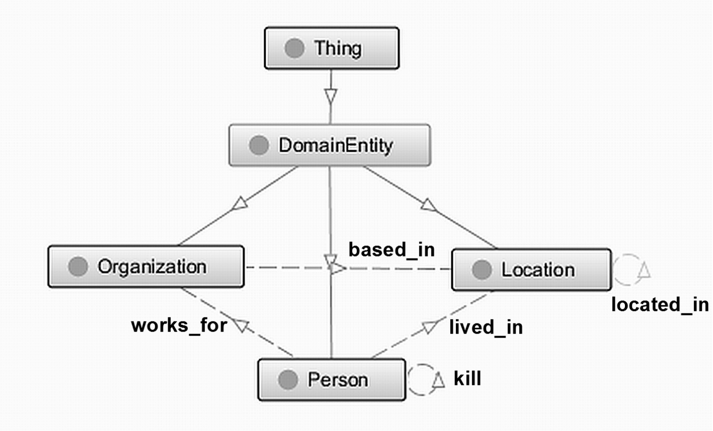


Fig. 10 Domain ontology with entities and relations types derived fom the TREC dataset

[12] http://cogcomp.cs.illinois.edu/Data/ER/conll04.corp

5.2 Generation of Negative Examples

In OntoILPER, the task of inducing the target predicates requires that positive and negative examples be explicitly indicated before learning. For NER, we create negative examples as the complement of the positive ones, according to the *one vs. all class binarization* technique [1]. In short, the underlying idea of the one vs. all strategy consists in producing several 2-class learning datasets by discriminating each class against the union of all the other classes. Thus, given the set of N possible entity classes C_i, $i = 1..N$, for each positive instance c_i of a given class C_i in the training set, a negative example is created for each one of the other $N - 1$ classes. Thereby, a multiclass learning problem is reduced to several binary classification problems.

The RE extraction task in OntoILPER is also seen as a binary classification problem where argument pairs that are actually related to each other in a relation denote the positive examples, whereas the other pairs of co-occurring entities in the same sentence are negative examples. As a result, for each sentence and each relation, $C_{n,2} = n! / 2*(n - 2)!$ Examples are created; where n is the total number of entities in a sentence.

5.3 Evaluation Metric, Cross-Validation, and Optimal Parameters

The performance evaluation is based on the information retrieval classical measures of Precision P, Recall R, and *F1-measure* [4]. We employed 5-fold cross validation which allows both the maximal use of the available training data, and comparison with existing NER and RE systems (Section 6.1). In addition, we performed several preliminary experiments for determining the optimal ILP learning parameters according to the criteria of achieving high accuracy, and preventing model overfitting. We estimated the best parameters values by applying the method proposed in [23]. As a result, the following parameter setting was determined and is employed in all experiments reported in this section: *evalfn* = coverage, *i* (depth) = 3, *minpos* = 5, and *noise* = 0.2.

5.4 Results and Discussion on NER and RE

Several experiments on NER and RE using the TREC dataset were conducted for evaluating the effectiveness of the extraction models for entities and relations generated by OntoILPER. In particular, we discuss the implications of the results achieved by the three types of extraction models for entities and relations proposed by [39] and already introduced in Section 4.5.2.

For all the experiments, we adopted the 5-fold cross-validation that not only provides unbiased performance estimates of the learning algorithms, but also enables the comparison with other IE systems evaluated over the same corpus. Moreover, although OntoILPER provides a named entity tagger (from the Standford CoreNLP) in its preprocessing component, we decided not to employ it in order to have a fair experimental setup when comparing it with other systems compared in this section.

Tables 4 and 5 summarize the classification results achieved by all the three aforementioned models. Tab. 4 shows that all the classification models for the entities *Location* (LOC), *Organization* (ORG), and *Person* (PER) obtained high overall accuracy. On the one hand, all of these models (E_O, E_P, E_S) are highly precise, with precision scores ranging from 93.5 (obtained by the PER entity) to

98.7 (obtained by the ORG entity). On the other hand, the recall scores were quite good (ranging from 74.4 to 92.4). Such results also reveal the balanced trade-off between precision and recall in all the classification models for LOC and PER entities. On the contrary, the classification models for predicting ORG entities obtained the highest precision among all entities, but also achieved the lowest recall scores.

RE, a more challenging task than NER, was once more confirmed by the relation models performance reported in Tab. 5. Similarly to the entity extraction models, the RE models (R_O, R_P, and R_S) are more precise but with lower recall: with precision scores ranging from 85.7 to 93.1, while recall scores range from 72.1 to 86.1. Although the results in Tab. 4 and 5 suggest OntoILPER preference of precision over recall, this is not a correct conclusion because OntoILPER can use other evaluation functions, such as the *recall evaluation* function [41], which prefers recall than precision during learning.

Table 4 Results for Entity Classification (All Models)

NER Model	**LOC**			**ORG**			**PER**		
	P	*R*	*F1*	*P*	*R*	*F1*	*P*	*R*	*F1*
E_O	95.9	92.4	**94.1**	98.7	79.2	**87.8**	93.7	91.2	**92.4**
E_P	95.2	92.0	93.5	97.5	76.5	85.7	93.5	89.0	91.3
E_S	96.0	88.4	92.0	97.0	74.4	84.3	94.8	87.5	91.0

Table 5 Results for Relation Classification (All Models)

RE Model	**located_in**			**work_for**			**orgBased_in**			**live_in**			**kill**		
	P	*R*	*F1*	*P*	*R*	*F1*	*P*	*R*	*F1*	*P*	*R*	*F1*	*P*	*R*	*F1*
R_O	90.5	78.6	**84.0**	85.7	86.1	**85.8**	88.7	82.5	85.4	87.4	76.9	**81.7**	92.3	78.0	**84.3**
R_P	91.1	78.0	83.9	87.2	80.8	83.8	91.5	84.0	**87.5**	85.7	72.1	78.2	91.5	77.6	83.9
R_S	91.2	75.9	82.6	93.1	72.9	81.7	88.4	77.0	82.2	92.5	67.4	78.0	97.5	73.7	83.8

Discussion. The overall F1 performance of the models for entities E_O and E_P was higher than the baseline performance of the E_S model for all entities (Tab. 4). Such results were expected as the E_O and E_P models are richer, i.e., they are more informed models than the E_S model.

For almost all the relation models in Tab. 5 (except *OrgBased_in*), the entity labels from the input corpus not only decrease the precision of the RO relation model, but can also contribute to improve its recall score. Thus, the correct entity labels enable the R_O extraction models to cover more examples in this case.

Interestingly, these last results might raise the following question which concerns the application scenario of having a new dataset in which all the entities are already defined and annotated: *what is the best RE model to learn: R_P or R_S?* According to the results summarized in Tab. 4 and 5, the pipeline models outperformed the separate models on both NER and RE tasks. However, especially for the RE models, there was a significant statistical difference in terms of F1 between the RP and RS relation models.

In conclusion, the distinctive feature of the OntoILPER learning process, i.e., its capability to employ rules learned in a previous learning stage, as additional BK predicates at a posterior learning stage, turns out to be very useful, as suggested by the above results over the TREC corpus.

Induced Rules. In the following, we show an induced rule of the R_P model for the *located_in* relation. This rule is expressed in terms of (*number of literals*), *(positive examples covered*), (*negative examples covered*), and the (*rule precision P*):

Rule: #Literals = 4, PosScore = 187, NegScore = 19, Prec = 90.8%

located_in(A,B):- t_ner(A,loc), t_next(A,B), t_ner(B,loc).

The above rule, in Prolog syntax, classifies an instance of the *located_in* relation in which its high precision score is mainly due to the high frequency of many phrases similar to "Perugia, Italy" in the learning corpus, indicating that the first argument (*A*) "Perugia" is followed by (predicate *next*) the second argument (*B*) "Italy", not considering the punctuation symbol between them. Other extraction rules for entities and relations are illustrated next.

```
% Induced rules for named entities
1: loc(A):- t_hasDep(prep_in,B,A)
2: loc(A):- t_pos(A,nnp), t_orth(A,upperinitial),t_bigPosBef(A,in-dt)

3: per(A):- t_hasDep(nsubjpass,B,A),t_pos(A,nnp),t_isHeadNP(A)
4: per(A):- t_isHeadNP(A),t_pos(A,nnp),t_trigPosAft(A,nn-in-dt)

5: org(A):- tok_length(A,medium),t_orth(A,uppercase),t_pos(A,nn)
6: org(A):- t_hasDep(conj_and,B,A),t_trigPosBef(B,nns-vbp-nnp)

% Induced rules for relations
7: located_in(A,B):- t_ner(A,loc),t_next(B,C),t_next(C,A),t_ner(B,loc)
8: located_in(A,B):- t_orth(B,upperinitial),t_next(A,C),t_next(C,D),
                       t_isHeadNP(A)

9:  work_for(A,B) :- t_isHeadNP(A),t_next(A,C),t_hasDep(prep_for,C,B)
10: work_for(A,B) :- t_next(B,C),t_orth(A,mixedcase),t_isHeadNP(A),
                       t_hasDep(nn,A,C)

11: live_in(A,B)  :- t_next(A,C),t_pos(C,vbz),t_next(C,D),t_next(D,B),
                       t_pos(B,nnp).
12: live_in(A,B)  :- t_orth(A,mixedcase),t_next(B,C),t_hasDep(nn,A,C),
                       t_pos(C,nnp), t_isHeadNP(A)
```

- *Rule 1* classifies LOC entities, *loc(A),* if there exists a grammatical dependency (preposition "in") between a token *A* and another token *B* in the same sentence;
- *Rule 5* identifies ORG entities, *org(A)*, if a token *A* is a noun in uppercase and having a medium number of characters according to the user BK predicates defined in Section 4.4.3;
- *Rule 7* identifies instances of the *located_in (A, B*) relation if both tokens *A* and *B* are recognized as LOC entities and there exists a given token *C* between them.
- *Rule 9* classifies *work_for (A, B)* relation instances when the token *A* is the head of a noun chunk, followed by a token *C*, and there is a prepositional dependency ("for") between the tokens *C* and *B*.

As already mentioned in Section 4.1.2, OntoILPER is based on ProGolem, a general bottom-up ILP learner that implements a global theory construction method. That is, this form of theory construction ensures that the theory (the final set of induced rules) is only constructed after the entire set of candidate rules have been generated, which completely avoids the generation of conflicting rules. As a result, ProGolem is not dependent of the order of examples during learning.

Another important aspect worth mentioning concerns the redundancy level of the set of rules learned by ProGolem. After inspecting the rules learned from the TREC dataset, we found that 5 to 8 percent of them cover most of the examples of another rule. However, no completely redundant rule was found, i.e., at least one example covered by a given rule R_i was not covered by the other rules R_j, $i \neq j$)

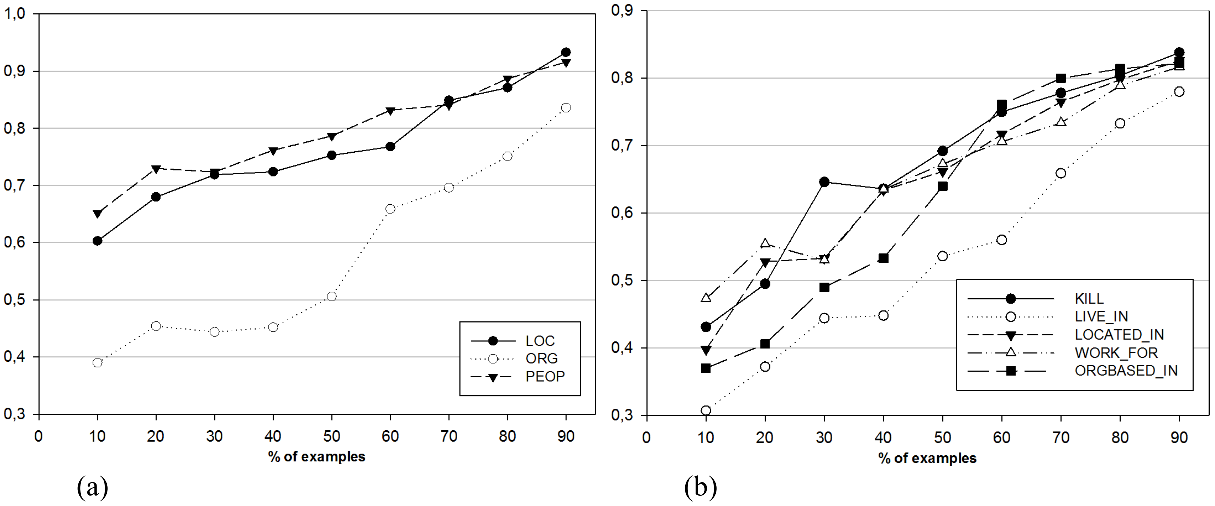
5.5 Learning Curves

Further evaluations of the E_S and R_S models were performed. It aims at investigating the effect of having limited training examples during learning. This is done by incrementally adding subsets of examples as training data to OntoILPER.

For that, nine experiments were conducted in which incremental portions of the training dataset, corresponding to 10% of the total number of examples each one, are added up to the previous subset of training data. Therefore, starting from a training dataset with only 10% of the total training examples, we generated other training datasets with 20%, 30%, 40%, and so on.

The learning curves in Fig. 11 relate the F1 score for each portion of the training dataset. It can be observed in Fig. 11 (a) that the classification models for LOC and PER entities yielded reasonable F1 scores (around 70%) with just 20% of the total number of training examples. This corresponds to 30 and 26 extraction rules in the final induced LOC and PER models, respectively. In contrast, for the ORG entity extraction model, more learning examples were necessary to attain the same performance.

In Fig. 11 (b), almost all the relations have increasing performance as more and more training data are available, with steadily increasing relation learning curves. However, *org_based_in* and notably the *live_in* relations had lower F1 scores for the 10%-40% of the training data, becoming rapidly higher for the rest of the training corpus.



(a) (b)

Fig. 11 Learning curves (F1) for (a) entity ES and (b) relations RS models

6 Comparative Evaluations

This section provides a comparative assessment of the NER and RE classification models generated by OntoILPER with the best ones presented in [39,13,22,2] over the TREC dataset. To the best of our knowledge, these are the only works that used this dataset and they are briefly presented next.

Giuliano et al. (2007) [13] propose an NER/RE system based on shallow linguistic features derived from tokenization, lemmatization, and POS tagging. Their solution relies on a combination of kernel functions, which uses two distinct information sources: (i) the *global contexts* where entities appear, and (ii) the *local contexts* around the interacting entities. The whole sentence (global context) is employed to discover the presence of a relation between two entities, while text windows of limited size centered on the entities (local contexts) provide clues to identify the roles played by the entities in a relation.

Roth and Yih (2007) [39] introduce an NER/RE system based on global inference or *joint extraction* of entities and relations. Their approach first identifies entities and relations in a sentence using separate classifiers learned from local information of the sentence. Then, it computes the most probable consistent global set of entities and relations using linear programming. The constraints induced from the dependencies among entity types and relations constitute a relational structure over the outcomes of the predictors (global inference).

Kate and Mooney (2010) [22] propose a joint extraction approach using a "card-pyramid" graph in which labelled nodes compactly encode all the entities and relations in a sentence. An efficient labelling algorithm that is analogous to parsing using dynamic programming constructs the card-pyramid graph. The advantage of this approach is that extraction from a part of the sentence is influenced by extraction from its subparts and vice-versa, thus leading to a joint extraction. Their implementation is based on the LIBSVM[13] software for building SVM classifiers.

Alicante and Corazza (2011) [2] employ tree kernels to the whole sentence parse tree and a linear kernel to a vector of binary features derived from the words surrounding each of the involved entities. The authors proposed the so-called barrier features that describe the syntactic context of tokens in entities, usually taking into account nouns or adjectives. For each candidate relation label, they create a binary SVM classifier taking as input both a feature vector and the parse tree of the whole sentence. The authors also included WordNet[14] sense tags and the hypernyms for each token denoting an entity.

6.1 Discussion

The comparative results of the aforementioned NER/RE systems are summarized in Tables 6 and 7. The results on NER (Tab. 6) show that the MC model had superior performance in terms of F1 compared to the other systems. However, this model uses many gazetteers for location, people's names, and organizations in its pre-processing phase, which certainly has a boosting impact on its NER performance.

OntoILPER E_S model obtained competitive results against the Separate w/Inf and Card-Pyramid models on NER, especially for LOC and PER entities. This model was also the most precise

[13] LIBSVM. A library for Support Vector Machines. https://www.csie.ntu.edu.tw/~cjlin/libsvm/

[14] WordNet. A lexical database for English. https://wordnet.princeton.edu

among the evaluated systems, but achieved lower recall. As a consequence, for those applications in which precision is more desirable than recall, the E_S model could be the best option, as it could avoid overloading users with too many false positives. Future work on feature engineering, especially for NER, can contribute to further improve OntoILPER results.

Table 6 Comparative results of the best models for NER. The highest (P/R/F1) scores are in bold

NER Model	*LOC*			*ORG*			*PER*		
	P	*R*	*F1*	*P*	*R*	*F1*	*P*	*R*	*F1*
MC (Giuliano et al., 2007) [13]	94.2	**94.4**	**94.3**	91.9	88.5	**90.2**	**94.8**	**96.6**	**95.7**
Separate w/Inf (Roth & Yih, 2007) [39]	91.8	88.6	90.1	91.2	71.0	79.4	90.6	90.5	90.4
Card-Pyramid (Kate & Mooney, 2010) [22]	90.8	94.2	92.4	90.5	**88.7**	89.5	92.1	94.2	93.2
E_S (OntoILPER)	**96.0**	88.4	92.0	**97.0**	74.4	84.3	**94.8**	87.5	91.0

Table 7 Comparative results of the best models for RE. The highest F1 score are in bold for each relation

RE Model	*located_in*			*work_for*			*orgBased_in*			*live_in*			*kill*		
	P	*R*	*F1*	*P*	*R*	*F1*	*P*	*R*	*F1*	*P*	*R*	*F1*	*P*	*R*	*F1*
MO|KSL (Giuliano et al., 2007) [13]	79.6	76.0	77.8	76.8	80.0	78.4	74.3	77.2	75.7	78.0	65.8	71.4	82.8	81.0	81.9
Omniscient w/Inf (Roth & Yih, 2007) [39]	61.9	62.9	59.1	79.2	50.3	61.4	81.7	50.9	62.5	63.9	57.3	59.9	79.9	81.4	79.9
Card-Pyramid (Kate & Mooney, 2010)[22]	67.5	56.7	58.3	73.5	68.3	70.7	66.2	64.1	64.7	66.4	60.1	62.9	91.6	64.1	75.2
Barrier Feat. (Alicante & Corazza,2011)[2]	70.0	75.4	72.6	76.4	86.2	80.9	86.6	77.7	81.9	74.7	73.4	74.3	92.4	75.6	83.2
R_O OntoILPER	90.5	78.6	**84.0**	85.7	86.1	**85.8**	88.7	82.5	**85.4**	87.4	76.9	**81.7**	92.3	78.0	**84.3**

The results on RE summarized in Tab. 7 show that OntoILPER outperformed all the other systems, according to statistical significance tests (*paired Student t*) for the difference among the F1 scores, at $\alpha = 0.05$ (95% confidence interval). The main reason is probably due to the richer sentence representation model employed by OntoILPER that takes into account structural information. In fact, in our graph-based model, any kind of relationships between entities in a sentence are represented using a formalism of representation (first order logic) which is more expressive than the propositional representation employed by the selected systems above. Furthermore, according to [18], kernel-based methods applied to RE are not able to fully exploit structural information. On the contrary, OntoILPER overcomes this shortcoming by providing a well-structured hypothesis space combining structural relations and node properties in a graph-based model that integrates lexical, syntactical, and semantic information.

A closer look at the results in Tab. 7 also reveals that, the Card-Pyramid model obtained the lowest F1 scores for the *located_in* and *kill* relations among all of the compared systems, whereas the Barrier Feature model yielded the second best F1 scores for almost all the relations, except for the *located_in* relation, in which the second best RE model was *MO|KSL*.

A final remark concerns the Text Preprocessing component in OntoILPER which is based on supervised models trained on the newswire domain. This fact might lead to the following question: "Can OntoILPER achieve state-of-art performance on another domain?" Indeed, due to its extensive range of relational features easily integrated into a carefully tailored hypothesis space for the RE task, OntoILPER has equally outperformed other state-of-the-art RE systems on the biomedical domain [26].

7 Conclusion and Future Work

This paper presented OntoILPER, a novel OBIE method for extracting entity and relation instances from natural language texts based on ILP. OntoILPER relies on an effective graph-based model of sentence representation that takes into account a condensed set of relational features which has been proved to be very effective for more complex IE tasks such as RE. Another major component in OntoILPER architecture is its ILP-based rule-learning component that employs the domain ontology as guidance during induction of symbolic extraction rules. Experiments conducted on the TREC dataset demonstrated OntoILPER effectiveness on both NER and RE tasks. In a comparative assessment, the yielded results also showed that OntoILPER is competitive on NER and outperforms other RE systems.

OntoILPER approach is based on a symbolic machine learning method, which combines several advantages. The first advantage resides in the fact that NER, RE, and ontology population tasks are treated at the same semantic level of the application domain, i.e., the semantic level is expressed by logical programs, regarded as extraction rules in first-order logic, which are very expressive. OntoILPER not only has the capability of integrating other semantic resources as BK, which promotes a higher level of adaptiveness to new application domains, but also allows for automatic reasoning mechanisms from the Semantic Web [3].

Future Work. Despite OntoILPER encouraging results, there is still room for improvement: (i) OntoILPER currently relies on shallow syntactic parsing, which does not take into account deeper semantic aspects of the sentences; (ii) the strategy of generating negative examples in OntoILPER can produce unbalanced distributions of positive and negatives training examples, which may hamper performance, as pointed out in [34]. To address the aforementioned shortcomings, we plan to: (i) integrate further BK into the preprocessing step, such as synonyms and hypernymys/hyponyms from WordNet, semantic role labeling [9], and word sense disambiguation [6], since these semantic resources have been proven to improve performance in many IE applications [11]; and, (ii) investigate the impact of undersampling techniques which would allow speed up the learning task by reducing the number of negative examples [34].

We will also investigate ILP-based rule induction from larger datasets aiming to promote OntoILPER scalability. Previous work for promoting scalability in ILP-based rule learning includes sampling techniques, for only selecting the most informative examples and removing the redundant ones [49]; and parallel ILP processing [50] [51] that can decompose the learning problem into smaller more manageable parts.

Directly related to the issue of applying OntoILPER over larger datasets, are the feature generation and selection steps. OntoILPER generates a wide range of features of different nature as BK. On the one hand, such a high number of features can describe several aspects regarding the nature of the data. On the other hand, this can produce a high dimensional space. To address this problem, we intend to apply dimensionality reduction techniques that not only can significantly reduce extra processing time during learning, but also avoid undesirable noise [52].

Finally, we will concentrate on how to adapt OntoILPER for performing Event Extraction (EE), the subfield of IE that aims at identifying n-ary relations [48]. In particular, we intend to deal with EE in the biomedical domain that refers to the change of state of one or more biomedical entities, including proteins, cells and chemicals [52]. In its textual realization, an event is usually denoted by a trigger expression that specifies the event and its type. Such triggers are typically expressed by verbal forms, while the entities (participants) involved in the event further specify the event. The Turku Event Extraction System (TEES) [48] detects events by using a rich feature set built from a graph-scheme for representing named entities and trigger words as nodes, and event arguments and relations as edges. The features generated from this graph are then transformed into a vector representation as input for SVM classifiers. TEES performs classification in two separate stages: trigger detection, and edge detection, which associates event triggers with their arguments. TEES has achieved state-of-the-art performance in several BioNLP shared tasks [48].

As OntoILPER employs a very similar graph-based representation as TEES, we can equally use both the graph nodes and edges as features for inducing event extraction rules. Actually, OntoILPER would have the advantage over TEES in the sense that it would jointly perform EE, i.e., it would learn the extraction rules in a single step. This has the potential of avoiding the small performance loss obtained by TEES, as discussed in [48].

Acknowledgement

The authors are grateful to Hilário Oliveira for his help in the development of some of the OntoILPER components. We also thank the National Council for Scientific and Technological Development (CNPq/Brazil) for financial support (Grant No. 140791/2010-8).

References

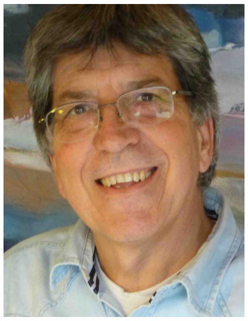
1. Airola A, Pyysalo S, Björne J, Pahikkala T, Ginter F, Salakoski T (2008) All-paths graph kernel for protein–protein interaction extraction with evaluation of cross corpus learning, BMC Bioinformatics, 9:S2
2. Alicante A, Corazza A (2011) Barrier Features for Classification of Semantic Relations. In: Proceedings of the International Conference Recent Advances in Natural Language Processing (RANLP) 2011, September, Hissar, Bulgaria, pp. 509-514
3. Baader F, Horrocks I, Sattler U (2008) Description Logics. Handbook of Knowledge Representation. Elsevier. Atlanta
4. Baeza-Yates R, Ribeiro-Neto B (1999) Modern Information Retrieval. Addison-Wesley, Boston.
5. Brown M, Kros JF (2003) Data Mining and the Impact of Missing Data. Industrial Management and Data Systems, 103(8), pp. 611-621
6. Ciaramita M, Altun Y (2006) Broad-coverage sense disambiguation and information extraction with a supersense sequence tagger. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP '06), Association for Computational Linguistics, Stroudsburg, PA, USA, pp. 594-602
7. Choi SP, Jeong CH, Choi YS, Myaeng SH (2009) Relation extraction based on extended composite kernel using flat lexical features, JKIISE: Software Application, 36:8
8. Choi SP, Lee S, Jung H, Song S (2013) An intensive case study on kernel-based relation extraction. In: Proceedings of Multimedia Tools and Applications, Springer, US, pp. 1-27
9. Christensen J, Mausam, Soderland S, Etzioni O (2010) Semantic role labeling for open information extraction. In: Proceedings of the NAACL HLT, First International Workshop on Formalisms and Methodology for Learning by Reading (FAM-LbR '10), ACL, Stroudsburg, PA, USA, pp. 52-60
10. De Marneffe M-C, Manning CD (2006) Stanford typed dependencies manual. Technical Report. Department of Computer Science, Stanford University
11. Dou D, Wang H, Liu H (2015) Semantic data mining: A survey of ontology-based approaches. IEEE International Conference on Semantic Computing (ICSC), 2015, Anaheim, CA, pp. 244-251
12. Fürnkranz J, Gamberger D, Lavrac N (2012) Foundations of Rule Learning, Springer-Verlag, Berlin

13. Giuliano C, Lavelli A, Romano L (2007) Relation Extraction and the Influence of Automatic NER. ACM Transactions on Speech and Language Processing, vol 5, no.1, ACM
14. Gruber T (1993) Towards Principles for the Design of Ontologies used for Knowledge Sharing. International Workshop on Formal Ontology in Conceptual Analysis and Knowledge Representation, Kluwer Academic Publishers, Deventer, the Netherlands
15. Gutierrez F, Dou D, Fickas S, Wimalasuriya D, Zong H (2015) A Hybrid Ontology-based Information Extraction System. Journal of Information Science, 2015, pp. 1-23
16. Hitzler P, Krötzsch M, Parsia B, Patel-Schneider PF, Rudolph S (2009) OWL 2 Web Ontology Language Primer. W3C Working Draft. http://www.w3.org/TR/owl2-primer
17. Horvath T, Paass G, Reichartz F, Wrobel S (2009) A Logic-based Approach to Relation Extraction from Texts. In: Proceedings of the 19th international conference on Inductive logic programming (ILP'09), Luc De Raedt (Ed.), Springer-Verlag, Berlin, Heidelberg, pp. 34-48
18. Jiang J (2012) Information Extraction from Text. C.C. Aggarwal and C.X. Zhai (eds), Mining Text data, pp. 11-41
19. Jiang J, Guan Y, Zhao C (2015) WI-ENRE in CLEF eHealth Evaluation Lab 2015: Clinical Named Entity Recognition Based on CRF. Conference and Labs of the Evaluation forum Toulouse, France, September 8-11, CLEF (Working Notes)
20. Jiang J, Zhai CX (2007) A systematic exploration of the feature space for relation extraction. Annual Conference of the North American Chapter of the Association for Computational Linguistics, NAACL-HLT’2007, Rochester, NY, USA, pp. 113–120
21. Karkaletsis V, Fragkou P, Petasis G, Iosif E (2011) Ontology Based Information Extraction from Text. Paliouras G. et al. (Eds.) Multimedia Information Extraction, LNAI 6050, pp. 89–109
22. Kate RJ, Mooney RJ (2010) Joint Entity and Relation Extraction using Card-Pyramid Parsing. In: Proceedings of the 14th Conference on Computational Natural Language Learning (CoNLL-2010), Uppsala, Sweden, July, pp. 203-212
23. Kohavi R, John GH (1995) Automatic parameter selection by minimizing estimated error. 12th International Conference on Machine Learning, San Francisco, Morgam Kaufman
24. Lavrac N, Dzeroski S (1994) Inductive Logic Programming: Techniques and Applications. Ellis Horwood, New York
25. Li M, Munkhdalai T, Yu X, Keun HR (2015) A Novel Approach for Protein-Named Entity Recognition and Protein-Protein Interaction Extraction, Mathematical Problems in Engineering, vol. 2015
26. Lima R, Batista J, Ferreira R, Freitas F, Lins R, Simske S, Riss M (2014) Transforming graph-based sentence representations to alleviate overfitting in relation extraction. In: Proceedings of the 2014 ACM symposium on Document engineering (DocEng '14), ACM, New York, NY, USA, pp. 53-62
27. Lima R, Espinasse B, Freitas F (2015) Relation Extraction from Texts with Symbolic Rules Induced by Inductive Logic Programming. In: Proceedings of the IEEE International Conference on Tools with Artificial Intelligence, IEEE-ICTAI 2015, Vietri sul Mar, Italy, pp. 194-201
28. Lima R, Espinasse B, Oliveira H, Pentagrossa L, Freitas F (2013) Information Extraction from the Web: An Ontology–Based Method using Inductive Logic Programming. In: Proceeding of the IEEE International Conference on Tools with Artificial Intelligence, IEEE-ICTAI 2013, Washington DC, USA, pp. 741-748
29. Muzaffar AW, Azam F, Qamar U (2015) A Relation Extraction Framework for Biomedical Text Using Hybrid Feature Set. Computational and Mathematical Methods in Medicine, vol. 2015
30. Muggleton S (1991) Inductive Logic Programming. New Generation Computing 8 (4): 29
31. Muggleton S (1995) Inverse entailment and Progol. New Generation Computing, 13, pp. 245-286
32. Muggleton S, Fen C (1990) Efficient induction of logic programs. 1st Conference on Algorithmic Learning Theory Tokyo, pp. 368-381
33. Muggleton S, Santos J, Tamaddoni-Nezhad A (2009) ProGolem: a system based on relative minimal generalisation. 19th International Conference on ILP, Springer, Leuven, Belgium, pp. 131-148

34. Nitesh V, Chawla, Kevin W, Bowyer, Lawrence OH, Philip KW (2002) SMOTE: synthetic minority over-sampling technique. Journal of Artificial Intelligence Research, 16, 1, pp. 321-357
35. Patel A, Ramakrishnan G, Bhattacharya P (2010) Incorporating Linguistic Expertise Using ILP for Named Entity Recognition in Data Hungry Indian Languages, LNCS, vol. 5989, Springer Berlin Heidelberg, pp. 178-185
36. Petasis G, Karkaletsis V, Paliouras G, Krithara A, Zavitsanos E (2011) Ontology Population and Enrichment: State of the Art. In: G. Paliouras et al. (Eds.): Multimedia Information Extraction, LNAI 6050, pp. 134–166
37. Plotkin G (1971) A note on inductive generalization. Machine Intelligence 5 1971, pp. 153-163
38. Ramakrishnan G, Joshi S, Balakrishnan S, Srinivasan A (2008) Using ILP to Construct Features for Information Extraction from Semi-structured Text. In: Proceedings of the 17th International Conference on Inductive Logic Programming, LNAI 4894, Berlin, Springer, pp. 211-224
39. Roth D, Yih W (2007) Global Inference for entity and relation identification via a linear programming formulation. Introduction to Statistical Relational Learning, L. Getoor and B. Taskar, the MIT Press, Cambridge
40. Roth D, Yih W (2004) A Linear Programming Formulation for Global Inference in Natural Language Tasks. CoNLL (2004), pp. 1-8
41. Santos J (2010) Efficient Learning and Evaluation of Complex Concepts in Inductive Logic Programming, Ph.D. Thesis, Imperial College University
42. Seneviratne MD & Ranasinghe DN (2011) Inductive Logic Programming in an Agent System for Ontological Relation Extraction. International Journal of Machine Learning and Computing, vol. 1, no. 4, pp. 344-352
43. Smole D, Ceh M, Podobnikar T (2011) Evaluation of inductive logic programming for information extraction from natural language texts to support spatial data recommendation services. International Journal of Geographical Information Science, 25, pp. 1809-1827
44. Tang J, Hong M, Zhang D, Liang B, Li J (2007) Information Extraction: Methodologies and Applications. Emerging Technologies of Text Mining: Techniques and Applications, Idea Group Inc., Hershey, USA, pp. 1-33
45. Wimalasuriya DC, Dou D (2009) Ontology-Based Information Extraction: An Introduction and a Survey of Current Approaches, Journal of Information Science, JIS-0987-v4, pp. 1–20
46. Wimalasuriya DC, Dou D (2010) Components for Information Extraction: Ontology-Based Information Extractors and Generic Platforms. CIKM’10, October 26–30, Toronto, Ontario, Canada
47. Zhou G, Zhang M, Ji D-H, Zhu Q (2007) Tree Kernel-based Relation Extraction with Context-Sensitive Structured Parse Tree Information. Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, Prague, pp. 728–736
48. Björne J, Salakoski T (2015). TEES 2.2: Biomedical Event Extraction for Diverse Corpora. BMC Bioinformatics 16. Suppl 16 (2015): S4. PMC. Web. 1 Nov
49. Byrd R, Chin G M, Nocedal J, Wu Y (2012). Sample size selection in optimization methods for machine learning". Journal of Mathematical Programming. Volume 134-1, pp.127-155.
50. Camacho R, Ramos R, Fonseca N (2014). AND Parallelism for ILP: The APIS System. Inductive Logic Programming: 23rd International Conference, ILP 2013, Rio de Janeiro, Brazil, August 28-30, 2013, Revised Selected Papers. Springer Berlin Heidelberg. pp. 93–106
51. Srinivasan A, Faruquie T, Joshi S (2012). Data and task parallelism in ILP using MapReduce. Journal of Machine Learning, vol 86-1, pp. 141-168.
52. Xia J, Fang, A C, Zhang X (2014) A novel feature selection strategy for enhanced biomedical event extraction using the Turku system. BioMed Research International, vol. 2014, Article ID 205239

Author Biography

Rinaldo Lima is an adjunct Prof. at the Rural Federal University of Pernambuco (UFRPE), Recife, Brazil. He graduated in Computer Science at the Federal University of Pernambuco (UFPE), and received a PhD in Computer Science from this university in 2014. From 2012-2016, he worked as a research fellow for Hewllet-Packard on several projects on Automatic Text Summarization and text mining applications. He collaborates with colleagues from the Aix-Marseille University in several research projects for more than 8 years. More recently, he has been working as a consultant researcher on research projects related to Alternative and Augmented Communication for people with special communication needs. He published several papers in international journals and conferences on Information Extraction, Automatic Text Summarization, and Semantic Web. His major research fields include ontologies, Text Mining, Machine Learning, and Semantic Web.



Bernard Espinasse is a Full Professor of Computer Science at the Aix-Marseilles University, Marseilles, France. He has been an Associate Professor at the Laval University, Québec, Canada (1983-1987). He received an engineer diploma from the Ecole Nationale Supérieure d'Arts et Métiers (1977) of Paris, and a Ph.D. and D.Sc. (habilitation) degrees in Computer Science in 1981 and 1990 respectively, from the Aix-Marseilles University. He has been a team leader at LSIS UMR CNRS, a research laboratory in computer sciences in Marseilles during fifteen years. He is the author of numerous publications in selective journals and conferences in information systems and artificial intelligence. His current research focuses on information extraction (text mining) and decision support systems, using machine learning, ontology, software agents, and semantic Web technologies.

Fred Freitas is associate professor at the Federal University of Pernambuco (UFPE), Recife, Brazil. He received his PhD in Electrical Engineering from the Federal University of Santa Catarina, Brazil, in 2002. He stayed in a sabbatical leave for over a year in 2010 at the University of Mannheim, Germany. He published in conferences and workshops, like IJCAI, ACM and IEEE, and in the Description Logic, Ontology Modularization and Biomedical Life Science Ontologies' workshops, as well as in high impact peer-reviewed Journal of Web Semantics, and Oxford Bioinformatics. He also co-edited special issues in journals as Journal of Brazilian Computer Science, Journal of Universal Computer Science, and Elsevier's Information Systems Journal. He has co-chaired two workshop series on Ontologies and their applications in Brazil, and on Building Applications with Ontologies for the Semantic Web in Portugal. His interest areas comprise ontologies, semantic web, description logic reasoning, knowledge representation and text mining.