Natural Language Engineering **8** (2/3): 167–191. © 2002 Cambridge University Press
DOI: 10.1017/S1351324902002863 Printed in the United Kingdom

Learning rules for information extraction

J. TURMO and H. RODRIGUEZ
Talp Research Center, Universitat Politècnica de Catalunya, Barcelona, Spain

(*Received 30 January 2002*)

Abstract

The growing availability of textual sources has lead to an increase in the use of automatic knowledge acquisition approaches from textual data, as in Information Extraction (IE). Most IE systems use knowledge explicitly represented as sets of IE rules usually manually acquired. Recently, however, the acquisition of this knowledge has been faced by applying a huge variety of Machine Learning (ML) techniques. Within this framework, new problems arise in relation to the way of selecting and annotating positive examples, and sometimes negative ones, in supervised approaches, or the way of organizing unsupervised or semi-supervised approaches. This paper presents a new IE-rule learning system that deals with these training set problems and describes a set of experiments for testing this capability of the new learning approach.

1 Introduction

Typically, information involved in knowledge-based systems has been manually acquired from experts on the subject of study. However, both the high cost of such a process and the existence of textual sources containing the required information have led to the use of automatic acquisition approaches. In the early eighties, Text-Based Intelligent (TBI) systems began to be used to automatically obtain the desired information by manipulating documents. These documents are usually highly structured when produced for computing use. However, documents have been frequently produced for human use and they consist of unrestricted Natural Language (NL) text. Sometimes, documents present a semistructured style where both chunks of NL text and structured pieces of information (meta-data) occur.

One major TBI area is IE. IE technology aims at locating and extracting the relevant content of a set of domain-specific documents. The definition of what content is relevant to be extracted is decided *a priori*, generally represented as a set of templates (typically named *scenario of extraction*). This fact implies a clear domain dependence of IE technology.

To obtain the desired information, IE systems involve several linguistic processes, and they are usually designed as *a cascade of transducers or modules that, at each step, add structure to the documents and, sometimes, filter relevant information, by means of applying rules* (Hobbs 1993). Most current systems follow this general architecture, although specific systems are characterized by their own set of modules,

https://doi.org/10.1017/S1351324902002863 Published online by Cambridge University Press

depending on the kind of documents they can deal with. Typical functionalities of such modules are: structuring the input texts, filtering textual zones that contain relevant information, attaching linguistic information by using linguistic processors (Part-of-Speech -POS- taggers, lexical analyzers, sentence splitters, syntactic parsers, word sense labellers, Named Entity -NE- recognizers), extracting information located within sentences and related to the scenario of extraction, solving referential relationships over all the extracted information among sentences, and generating the response in the desired format.

IE has consolidated itself as a useful technology for TBIS mainly through the Message Understanding Conferences (MUC[1]) efforts. These conferences, started in 1987, were sponsored by the United States Advanced Research Projects Agency (DARPA) under the TIPSTER project. The general goal of MUC conferences was the evaluation of IE systems developed by different research groups in order to extract information from restricted-domain free text. A different domain was selected for each conference (*Latin American Terrorism*, *Joint Ventures* and *Microelectronics*, *Management Successions*, and *Airline Crashes*). In general, extraction tasks involved were NE, Coreference, Template Element (i.e. set of entity properties), Template Relation (i.e. relationships between entities), and Scenario Template (i.e. events among entities). The evaluation metrics were based on the number of extracted template-slot fillers: *Recall* (R – percentage of slot fillers correctly extracted), *Precision* (P – percentage of extracted slot-fillers that are correct) and *F-score* (a consensus between R and P corresponding to their harmonic mean).

Beyond MUC competitions, the European Commission funded a number of projects under the LRE[2] program, which were devoted to develop tools and components useful to IE, such as automatically or semi-automatically acquiring and tuning lexicons from corpus, extracting NE, parsing in a flexible and robust way, and so on. Recently, research on IE technology has been included in the TIDES[3] program (Translingual Information Detection, Extraction and Summarization).

Besides these lines, IE technology has been applied to many domains in free text, as in Soderland, Fisher, Aseltine and Lehnert (1995) and Glasgow, Mandell, Binney, Ghemri and Fisher (1998). Other recent works refer to the application of IE technology to semistructured documents. Traditional IE systems do not fit to this documents because they are a mixture of grammatical and ungrammatical texts, sometimes telegraphic ones, and including meta-tags (e.g. HTML). Some examples are Soderland (1999), Craven (1999), Freitag (1998) and Califf (1998). In a complementary way, some researchers have applied IE technology to very structured documents, such as tabular ones within the Web (Kushmerick 2000; Muslea *et al.* 2000; Freitag and Kushmerick 2000).

[1] ftp://ftp.muc.saic.com/
[2] http://www.hltcentral.org/htmlengine.shtml?id=218#top
[3] http://tides.nist.gov/

https://doi.org/10.1017/S1351324902002863 Published online by Cambridge University Press

1.1 Machine learning for IE

The difficulty of exploiting IE technology is partly due to the difficulty of adapting IE systems to deal with new domains, new text styles and new languages. In order to handle these portability drawbacks, the use of Machine Learning (ML) techniques has increased, mainly due to the success of their application to some lower level NLP tasks. Borthwick, Sterling, Agichtein and Grisham (1998), Baluja, Mittal and Sukthankar (1999), Borthwick (1999) for NE recognition, Cardie, Daelmans, Nédellec and Sang (2000) for parsing, and McCarthy and Lehnert (1995), Aone and Bennett (1996), Cardie and Wagstaff (1999) and Mitkov (1998) for coreference and anaphora resolution are interesting examples of such approaches, while some surveys can be found in Young and Bloothooft (1997), Manning and Schütze (1999) and Mooney and Cardie (1999).

Most of the research effort has been devoted to apply symbolic inductive learning methods to acquire rules useful for different extraction tasks from a set of training documents. These rules are typically named *extraction patterns*, *scenario-specific patterns* or *IE rules*. Some of these efforts are under the propositional learning paradigm, such as PALKA (Kim and Moldovan 1995), CRYSTAL (Soderland *et al.* 1995), WAVE (Aseltine 1999), and TIMES (Chai, Biermann and Guinn 1999). Some others are under the relational learning paradigm, such as LIEP (Huffman 1995), SRV (Freitag 1998), RAPIER (Califf 1998), the approach in Roth and Yih (2001) and WHISK (Soderland 1999). Some of these approaches apply to free text while others apply to structured or semistructured texts. Most of them are supervised approaches, implying, thus, a hard task of manual annotation of the training documents.

Other approaches to inducing IE rules from free text try to reduce the cost of this manual annotation. For instance, HASTEN (Krupka 1995) and PET-PROTEUS (Yangarber and Grishman 1998) use a graphical interface to facilitate the manual selection and generalization of relevant examples within the training documents. In AutoSlog-TS (Riloff 1996), the user only has to annotate documents as relevant or non-relevant before learning. In Basili, Pazienza and Vindigni (2000) and Harabagiu and Maiorano (2000), approaches based on the introduction of syntactico-semantic biases are described. Bootstrapping techniques are used in Riloff and Jones (1999) and ExDISCO (Yangarber 2000).

Statistical modeling has been recently applied to IE tasks. Seymore, McCallum and Rosenfield (1999) and Freitag and McCallum (2000) use Hidden Markov Models (HMM), while in McCallum, Freitag and Pereira (2000), HMMs and Maximum Entropy Models are combined. In Freitag (1998), the usefulness of using a multistrategy approach is demonstrated within the framework of learning from semistructured documents on the Web.

As a complementary way, some work has been done on learning *wrappers*, i.e. procedures that can extract relevant information from Web pages to build databases for Internet applications. Such procedures look for delimiting slot-filler values by taking profit of meta-data (e.g. HTML tags). Relevant systems are WIEN (Kushmerick 2000), SoftMealy (Hsu and Dung 1998), STALKER (Muslea, Milton and Knoblock 2000), and BWI (Freitag and Kushmerick 2000).

https://doi.org/10.1017/S1351324902002863 Published online by Cambridge University Press

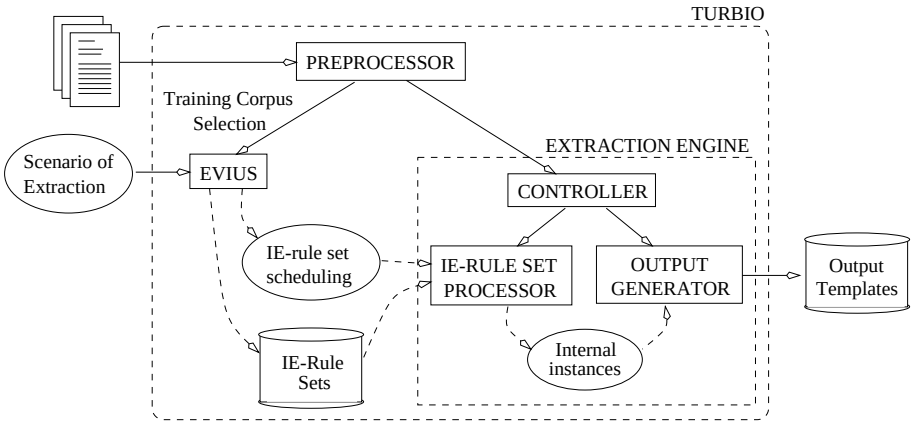


Fig. 1. General architecture of TURBIO.

Within this framework, this paper presents EVIUS, a new system to learn IE rules to extract information from free text. Most systems quoted previously are single-concept learning ones. They learn knowledge to extract instances of a concept within the scenario of extraction in an independent way. Instead, EVIUS is based on the fact that the scenario of extraction imposes some dependencies among concepts to be dealt with. When a concept depends on another one, knowledge about the former seems to be useful to learn to extract instances of the target one. EVIUS is a multi-concept learning system that incrementally learn to extract the whole scenario of extraction in a dependent manner.

The rest of the paper is structured as follows. Section 2 briefly describes TURBIO (Turmo, Catalá and Rodríguez 1999), the IE system in which EVIUS has been integrated. Section 3 presents the ML approach used by EVIUS. Section 4 describes and faces some problems related to the training set that affect the accuracy. A comparison of EVIUS to other similar IE rule learning system is described in section 5. Finally, conclusions are presented in section 6.

2 Architecture of TURBIO

TURBIO is an IE system that has been designed to be easily portable to new domains and languages, and which is able to work with free text.

As shown in Figure 1, the inputs of TURBIO are a scenario of extraction and a set of documents, from which information related to such scenario has to be extracted. TURBIO consists of three main blocks briefly described below: the Preprocessor, the Learning System, EVIUS and the Extraction Engine.

2.1 Preprocess of the input documents

The Preprocessor involves several components aimed at enriching the input documents with information useful to EVIUS and the Extraction Engine. These components involve different processes, as structuring documents in parts (ranking from

https://doi.org/10.1017/S1351324902002863 Published online by Cambridge University Press

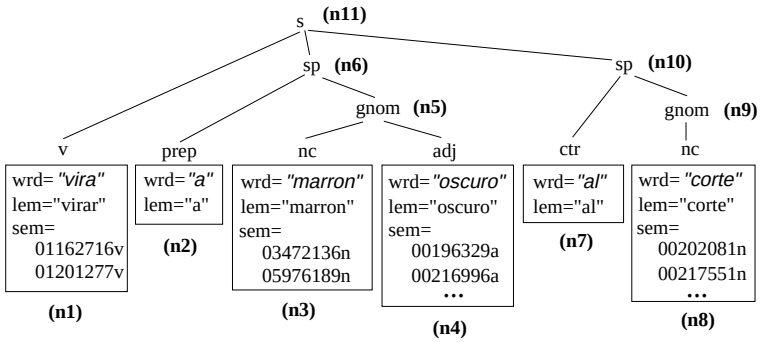


Fig. 2. Preprocessed training sentence.

free text to highly structured pieces), sentence splitting, POS tagging, parsing, and semantic interpretation. In particular, TURBIO enriches sentences occurring in free-text zones with morphological-syntactico-semantic information provided by different NLP tools and resources (http://www.lsi.upc.es/~nlp/tools.html). Semantics is provided by using EuroWordNet (http://nipadio.lsi.upc.es/wei.html), a lexico-semantic ontology in which semantic classes, represented as sets of synonym word-senses (synsets), are linked among them by different relations, such as hyponymy and hypernymy. Each sentence is structured as a set of parsed subtrees, and each word is linked to its corresponding EWN synsets.[4] Our aim has been to build highly reusable components that need only minor changes or tuning the knowledge and information sources when moving into a new domain (in the case of domain dependent components), or when using a new language. So, most of the Preprocessor components are general purpose ones. As a result of the Preprocessor performance, sentences are transformed into parse trees, such as the depicted in Figure 2 for the Spanish sentence *Vira a marrón oscuro al corte* (Changes to dark brown when cut).

2.2 The learning system

TURBIO needs EVIUS to be run only when moving to a new domain or language. In these situations, both the scenario of extraction and a training corpus are provided as the input of EVIUS. The training corpus consists of the free-text components occurring in a randomly selected set of preprocessed documents. The system learns IE-rule sets for the concepts within the scenario of extraction from the training corpus, as will be described in section 3. The learned rules are similar to Horn clauses and they can be easily translated into the rule formalism used by the Extraction Engine, which will be briefly described later in this section. Such IE-rule sets are stored within the IE-Rule Set DB as knowledge to extract from new documents containing free-texts similar to the training ones.

[4] With no attempt to disambiguating.

https://doi.org/10.1017/S1351324902002863 Published online by Cambridge University Press

2.3 Extraction engine

The Extraction Engine takes as input the preprocessed documents, those IE-rule sets learned by EVIUS, which are related to the scenario of extraction, and a scheduling of rule set activations computed by EVIUS. The Extraction Engine consists of two components, the IE-Rule Set Processor and the Output Generator, activated by a Controller module.

The Controller iteratively provides a sentence to the IE-Rule Set Processor. The aim of the IE-Rule Set Processor consists in applying all IE-rule sets involved in the scheduling to one this sentence. This is carried out by following the imposed order of execution. Each rule is represented as follows:

rule *Number*
ruleset *Rule_Set_Name*
<linguistic pattern> -> <action>

Rule_Set_Name and *Number* identify the rule. The condition part is a linguistic pattern in the form of a list of parse subtrees, containing morphological, syntactic and semantic information. When such a linguistic pattern matches the input sentence, the rule can be fired, and the action part can be executed to extract the relevant information.

As a result of the performance of the IE-Rule Set Processor, relevant information within each sentence is extracted. Once a whole document has been processed by the IE-Rule Set Processor, the Controller activates the Output Generator, which translate both structured information marked by the Preprocessor and internal objects extracted by the IE-Rule Set Processor into the desired output format.

3 Learning IE rules with EVIUS

The inputs of EVIUS are the scenario of extraction and a preprocessed training corpus. From these inputs, the aim of EVIUS is twofold:

- Computing the order in which concepts have to be learned.
- Learning an IE-rule set for each concept within the scenario in a supervised way. Positive training examples are annotated from the training corpus as explained in section 3.3.3.

We have selected the mycology domain for our experiments due to its conceptual richness and the variety of free-text expressions referred to its concepts. In particular, a set of 500 preprocessed mycology documents[5] has been used as training corpus, although only 68 of them have been annotated. These documents contain: (a) a heading describing the document identifier, and scientific and colloquial names of the mushroom to which the document refers, among others, (b) a Spanish description of the mushroom that consists of 11 sentences and 19 words per sentence in average,

[5] This document set has been granted by the Catalan Society of Mycology – http://www.micocat.org/publica.html.

https://doi.org/10.1017/S1351324902002863 Published online by Cambridge University Press

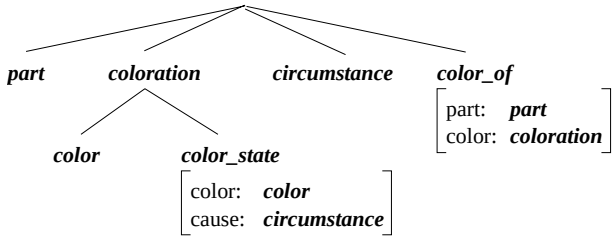


Fig. 3. Fragment of the scenario of extraction.

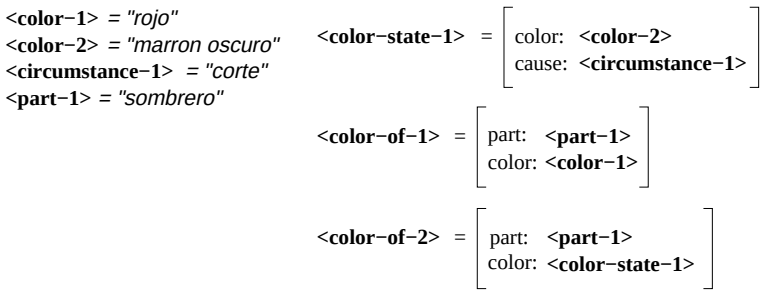


Fig. 4. Fragment of the scenario of extraction.

and (c) a Catalan description, which is not parallel to the Spanish one. Our aim is learning IE rules from the Spanish descriptions which consist of 12,914 different words and 4164 different lemmas. The fragment depicted in Figure 3 will be used as scenario of extraction along this paper. It refers to the color of the mushroom parts, which seems to be the most difficult concept to be extracted from the documents. Some text fragments within the Spanish descriptions related to this extraction task are *su color es de un blanco casi puro similar al del pie* (its color is almost-pure white similar to that of the stem), *rojo sangre hacia la base del pie* (blood red near the stem base), *color variando de blanco a amarillo* (color varying from white to yellow) and *su sombrero vira de rojo a marrón oscuro al corte* (its cap changes from red to dark brown when cut). For this last fragment, the desired output of out system is depicted in Figure 4.

EVIUS assumes that the domain independence cannot be guaranteed for any concept. In fact, each concept depends on the whole set of relevant concepts being represented within the specific scenario of extraction, in which some of them are related to others by means of the roles they play. Under such assumption, learning to extract an individual concept means learning to extract related concepts as well. For instance, taking our mycology scenario of extraction, concepts *part*, *coloration* and *color_of* are related by means of the attributes of the target one. Moreover, concepts *color* and *color_state* are related to concept *coloration* by means of relation *is_a*. How to extract all these related concepts has to be learned in due time.

Within this multi-concept situation, a multi-concept learning approach can be useful to incrementally learn to extract the whole scenario of extraction. In this process, the extraction of each concept can be learned by using a single-concept

https://doi.org/10.1017/S1351324902002863 Published online by Cambridge University Press

learning approach, and taking into account the already learned concepts, as useful background knowledge for further learning. For instance, for learning to extract *color_state*, knowledge about *color* and *circumstance* can be useful.

EVIUS is a multi-concept learning system based on a Multistrategy Constructive Learning (MCL) (Michalski 1993) approach and on a relational predicate model to represent the learning space. It is a supervised learning system, which needs a set of positive and negative examples for each concept. Following, the learning space, the MCL approach followed by EVIUS and the approach to learn to extract each concept within the MCL framework are described.

3.1 The learning space

As a preprocess of EVIUS, each preprocessed sentence within the input training corpus is translated into the relational predicate model. The set of predicates allows EVIUS to represent training sentences as complex structures, in which parse nodes are linked by multiple relations. Within this relational model, an example is defined as a set of nodes, as later explained in section 3.1.2. Being EVIUS based on a MCL approach, the learning space can change along the learning process by incorporating new background knowledge that is useful for further learning. As it will be explained in section 3.2, new nodes and relations among the resulting set of nodes will be added to the learning space.

3.1.1 The initial predicate model

The dynamic predicate model used by EVIUS consists of a set of attributive predicates, corresponding to features of nodes, and relational predicates, corresponding to relations among the nodes. The initial predicate model is described below.

Attributive Predicates. pos_X(A), word_X(A) and lemma_X(A), as morpho-syntactic predicates, and isa_X(A) and has_hypernym_X(A), as semantic predicates, where X is instantiated with categories related to node A of a parsed sentence. For instance, some of these predicates for the example in Figure 2 are:

word_marrón(n3)	lemma_marrón(n3)	pos_nc(n3)
isa_03472136n(n3)	has_hypernym_03464624n(n3)	...
isa_05976189n(n3)	has_hypernym_06270873n(n3)	...

being 03464624n (chromatic_color) and 06270873n (policeman) the first hypernym EWN synsets for both senses of *marrón*. All synsets through the hypernymy paths of the word are represented within the model.

Relational Predicates. distance_le_X(A,B) stating that there are X terminal nodes, at most, between nodes A and B, ancestor(A,B), where node B is the syntactic ancestor of node A, and brother(A,B), where node B is the right brother of node A sharing the syntactic ancestor. For instance, some relational predicates for the example in Figure 2 are:

distance_le_1(n3,n4)	distance_le_2(n3,n7)	ancestor(n3,n5)
ancestor(n4,n5)	brother(n2,n5)	...

https://doi.org/10.1017/S1351324902002863 Published online by Cambridge University Press

This initial predicate model is open, in fact, and other predicates could be taken into account.

3.1.2 Representing training examples

Examples used to learn a concept c can be modeled as tuples $<n_1, \ldots, n_k>$, where each n_i is a node within the parse tree. The meaning of n_i depends however on the kind of concept to be learned. From the representation of the scenario of extraction in Figure 3, two types of concepts are distinguished:

- *basic concepts*: terminal concepts within the scenario, concepts being represented as frames without attributes, as in the case of *part*, *color* or *circumstance* within our scenario of evaluation;
- *derived concepts*: concepts owning attributes, as in *color_state* and *color_of*.

If c is a basic concept, then $<n_1, n_2>$ is defined as the pair of terminal nodes being delimiters of a textual instance of c. For instance, example *marrón oscuro* (dark brown) for concept *color* is represented as <n3,n4>. If c is a derived concept, it owns a list of typed attributes, $<a_1{:}c_1, \ldots, a_k{:}c_k>$, where a_i is its ith attribute, and c_i refers to the type of values for a_i. Then, n_i is the value of the ith attribute of an example. For instance, let us consider that concepts *color* and *circumstance* have been learned and added to the background knowledge, as will be explained in section 3.2, then, example *marrón oscuro al corte* (dark brown when cut) for concept *color_state* will be represented as <n5,n9>, where n5 (*marrón oscuro*) corresponds to the value for the first attribute (color), and n9 (*corte*) to the value of the second one (cause).

3.1.3 Rule formalism

The resulting rules take the following form of first-order Horn clauses:

$$c(A_1, \ldots, A_n) :- < conjunction\ of\ predicates > \quad (1)$$

where <*conjunction of predicates*> is the precondition, and $c(A_1, \ldots, A_n)$ the conclusion. A_i is a variable representing either the limits of possible textual instances for a basic concept, or values of i-th feature of possible instances for a derived concept. Each of the predicates within <*conjunction of predicates*> may refer either to variables A_i, or to others related to them. For instance, some of the first rules learned for the concept *color* in our experiments are:

```
isa_color(A,A)  :-  pos_s_adj(A), has_hypernym_03464977n(A),
                    ancestor(A,C), pos_s_adj(C)
isa_color(A,B)  :-  has_hypernym_03460270n(A), has_hypernym_03492364n(B)
                    brother(A,B)
```

where synsets 03464977n, 02460270n and 03492364n refer to concepts *achromatic_color*, *visual_property* and *physical_property*, respectively.

https://doi.org/10.1017/S1351324902002863 Published online by Cambridge University Press

3.2 Multistrategy constructive learning approach

As a whole learning process, EVIUS is based on a MCL approach integrating *closed-loop learning*, *deductive restructuring* (Ko 1998) and *constructive induction*.

The use of closed-loop learning allows to incrementally learn to extract the whole scenario of extraction, by means of determining which concept to learn at each step. This scheduling, namely S, can be straightforwardly computed before learning, by using knowledge about the structure of the scenario: starting with S_0, which contains the basic concepts within the scenario of extraction, S_i is computed iteratively by adding those concepts which only involve concepts in S_{i-1}. In the case of the scenario of extraction depicted in Figure 3, S results as follows: $S_0 = \{color, circumstance, part\}$, $S_1 = \{color_state\}$, $S_2 = \{color_of\}$.

The set of concepts within each S_i can be independently learned in parallel. Once S_i has been learned, the learning space is updated by using deductive restructuring and constructive induction. These techniques allow adding more useful information for further learning.

3.2.1 Deductive restructuring

The use of deductive restructuring allows EVIUS to assimilate each training example of a learned basic concept as a new node within the learning space. This is done by shrinking the set of nodes involved in the training example.

Shrinkage consists in applying a set of rules to locate a unique node that will represent such example within the learning space, and will subsume all the involved information. For instance, using the training sentence in Figure 5,[6] once concept *color* has been learned, the training example *marrón oscuro* (dark brown) is shrunk. Node n5 is found to be the node representing the example, and nodes n3, n4 and n5 are restructured as a unique node, n5.

Shrinkage, however, cannot be used once a derived concept has been learned because nodes involved in a training example can also be involved in another one. For instance, training examples for concept *color_state* involve nodes representing examples of both concepts *color* and *circumstance*. However the same example of concept *color* could be associated with more than one example of the concept *circumstance*.

3.2.2 Constructive induction

The use of constructive induction allows EVIUS to add new predicates to the learning space. This is the case of a training example of the learned concept, after deductive restructuring has been applied.

For instance, in the example in Figure 5, and following the order in which concepts are learned, the following inductions are achieved, leading to a modification of the learning space:

[6] For the sake of simplicity, the training sentence has been represented here as a parse tree, although it is intended as the set of predicates involved.

https://doi.org/10.1017/S1351324902002863 Published online by Cambridge University Press

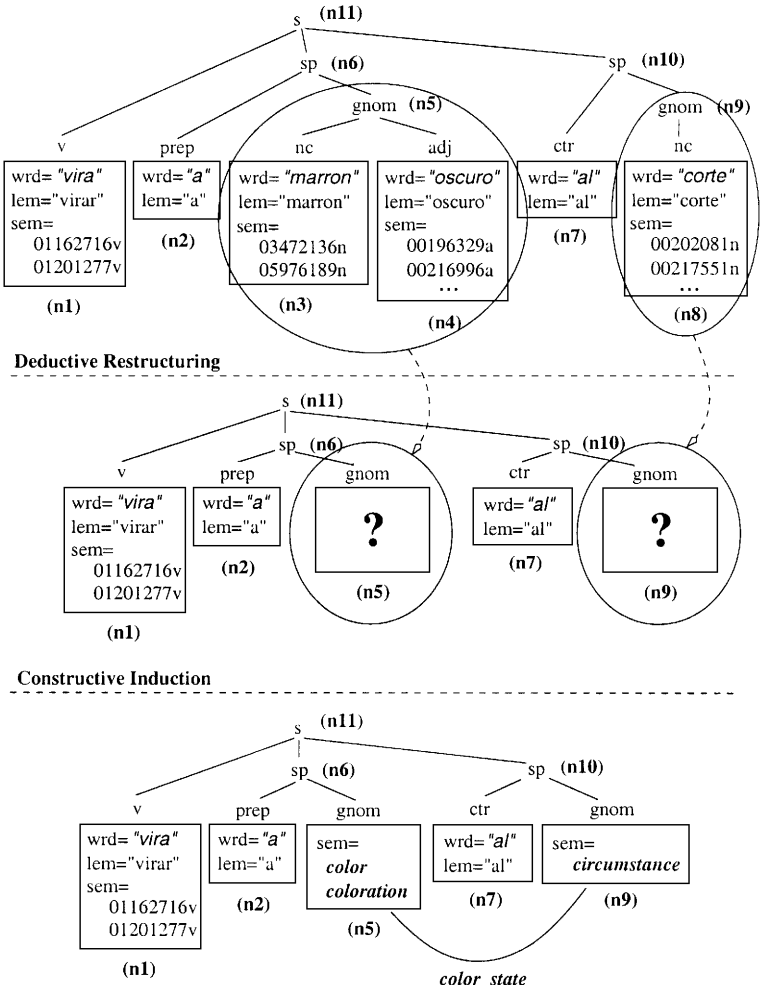


Fig. 5. Example of deductive restructuring and constructive induction represented as a parsed tree.

1. Once *marrón oscuro* is restructured into n5, predicates isa_color(n5) and isa_coloration(n5) are added to the learning space.
2. Once the training example *corte* of concept *circumstance* is restructured into n9, predicate isa_circumstance(n9) is added too.
3. Finally, once concept *color_state* has been learned, the predicate color_state-(n5,n9) is directly added to the learning space without applying deductive restructuring.

3.3 Learning to extract a concept

The MCL approach used by EVIUS implies an incremental learning of individual concepts. As explained below, each step is accomplished using FOIL (Quinlan 1990).

https://doi.org/10.1017/S1351324902002863 Published online by Cambridge University Press

3.3.1 Brief description of FOIL

FOIL is a well known relational learning system. It is aimed at inductively learning first-order rules from positive and negative examples, process being often referred to as Inductive Logic Programming (ILP). With minor differences, rules learned by FOIL (represented as in (1)) are similar to Horn clauses.

The input of FOIL is a set of extensional specifications corresponding to a set of relations. Some of these relations are selected to be learned as rule sets by taking into account the whole set of specifications as background knowledge. Basically, an extensional specification of a relation is defined as follows:[7]

- A header, which contains the relation name and the types of its arguments in the following form: $name(type_1, type_2, \ldots, type_n)$.
- A set of positive tuples of constants, $k_1, \ldots, k_n$, that accomplishes the relation, and, optionally, a set of negative ones for the relations to be learned. By default, the latter set is provided by FOIL using the *closed-world assumption*: all non-positive tuples are negative ones.

FOIL is based on a sequential covering algorithm. It learns rules one at a time, removing the positive examples covered by the latest rule before attempting to learn the next one. The process ends either when all positive examples have been covered, or when no rule can be learned from the remaining examples with an accuracy higher than a given threshold.

Each rule is learned by following a top-down strategy that is based on a hill-climbing, generate-and-test search. Briefly, starting with the most general rule, the *literal* (i.e. a variablized relation, a binding of variables, an instantiation of a variable, or the negation of any of the above, among others) that better specializes the partial rule is added, one at a time. For such a purpose, FOIL generates the set of possible literals to be added, and each one of them is tested over the training data to find the best one. This test is carried out by using a heuristic based on information theory, named *Foil_Gain*. This is a measure of the information gain achieved due to adding a literal to the partial rule. The literal achieving the highest information gain is selected to be added.

FOIL uses a minimum description length approach to halt the growth of a rule: in order to further grow the rule, its description length in literals must be smaller than the description length of the set of subsumed training data. On completion, a post-pruning step is performed to the learned rule. It consists in removing those literals that do not affect the accuracy of the rule. Once the covering algorithm has finished, redundant rules are removed from the whole set.

Complexity, accuracy and coverage of learned rules depend upon the set of parameters of FOIL. They impose some restrictions on the search of literals, for instance, only considering non-negative literals (false by default), setting the minimum gain of the best literal (80% by default), or setting the minimum accuracy of any rule (80% by default), among others.

[7] For the sake of simplicity, only relevant specifications have been detailed.

https://doi.org/10.1017/S1351324902002863 Published online by Cambridge University Press

3.3.2 Using FOIL within EVIUS

The main reasons to select FOIL among other ILP systems (cf. Mitchell 1997) to learn to extract individual concepts within EVIUS are the following:

- Systems using top-down strategies, like FOIL, often allow learning more general rules than bottom-up ones do, such as CIGOL and GOLEM. This is because bottom-up strategies generalize specific rules to cover positive training examples, and consequently, they may learn very specific rules that do not cover unseen positive examples.
- In generate-and-test search, as used in FOIL, the selection of each literal is based on testing possible partial rules over many examples, and consequently, the effects of noisy training data are minimized. On the contrary, systems using example-driven approaches, in which the refinement of a rule is based on individual examples, are less robust to errors in the training data.
- Other ILP programs in the state of the art of ML are based on FOIL.
- FOIL has been successfully applied in some other NLP problems.

We have used FOIL against the learning space defined for EVIUS, which has been described in section 3.1. Rules have been learned as Horn clauses because TURBIO is not able to manage rules owning negated predicates. This is why we have restricted the learning process to consider only non-negative literals.

When concept *c* has to be learned by FOIL, extensional descriptions of the predicates have to be provided from the dynamic learning space. Assuming that the only type of constant is *node*, the headers within the extensional descriptions take the form *name*(*node*, . . . , *node*). For instance, let us consider that the learning space of EVIUS owns the predicates generated from the example in Figure 2, and that concept *color* is required to be learned. Some of the headers within the extensional descriptions provided to FOIL are as follows:

pos_nc(*node*)	(e.g. nodes n3 and n4)
has_hypernym_03464624n(*node*)	(e.g. node n3)
ancestor(*node*,*node*)	(e.g. pairs <n1,n11>, <n2,n6>)

In addition, the predicate *isa_color*(*node*,*node*) is also provided as the relation between nodes to be learned by FOIL. However, the set of node tuples for this relation, which defines the set of training examples, has to be previously provided by EVIUS.

3.3.3 Providing FOIL with training examples

The set of positive examples has to be manually marked within the training corpus. The task of selecting positive examples can be very hard due to the complexity of some concepts within the scenario. The constructive learning approach followed by EVIUS allows optimizing the manual effort in the task of selecting positive examples to learn to extract derived concepts with FOIL.

On the one hand, when EVIUS selects a basic concept to be learned from the scheduling, EVIUS requires the user to annotate positive examples for the concept within the input training corpus, although some form of bootstrapping or active

https://doi.org/10.1017/S1351324902002863 Published online by Cambridge University Press

learning could be used to reduce the cost of such a manual task, by automatically searching for new examples from unnanotated corpus.

On the other hand, when a derived concept is selected by EVIUS to be learned, it automatically generates possible candidates to be positive examples of such a concept, and the user is only required to validate them. This is done by taking profit of the new knowledge added to the learning space by deductive restructuring and constructive induction techniques.

Let us suppose that concepts *color* and *circumstance* have already been learned, and that both deductive restructuring and constructive induction have been applied to their training positive examples. As a consequence, training examples *marrón oscuro* (dark brown) and *corte* (cut) have been restructured into nodes n5 and n9 (cf. Figure 5). A candidate for positive example of concept *color_state* can be generated as the combination <n5,n9>, assuming that <color:*color*, cause:*circumstance*> is the list of typed attributes owned by the concept, and that predicates isa_color(n5) and isa_circumstance(n9) exist in the background knowledge.

In general, being c the concept to be learned, and $<a_1{:}c_1, \ldots, a_k{:}c_k>$ the list of typed attributes owned by c, EVIUS generates candidates for examples of c as tuples $<n_1, \ldots, n_k>$. Each n_i within a tupla is a non repeated node within the same training sentence that accomplishes predicate isa_$c_i(n_i)$. Taking into account that the values of some attributes for an example may be elided, at least one n_i has to be instantiated. Finally, the user is required to validate candidates. When a candidate is asserted, all other candidates being subsumed by the validated one are dropped out.

Once the set of positive examples have been selected, negative examples can be automatically generated by closed-world assumption: a negative example is generated as a tuple of nodes within the same training sentence that has not been selected as positive example. However, the number of tuples generated under such an assumption becomes intractable by FOIL. This issue is discussed in the next section, which presents some automatic methods to reduce the cardinality of the set of tuples representing negative examples.

4 Dealing with training set problems

Assuming that positive examples selected as explained in the previous section are relevant for learning, two aspects can negatively influence the result of learning:

- *Relevance of negative examples*: Closed-world assumption can be used by FOIL only when dealing with small learning spaces. When the learning space consists of hundreds of predicates, as in our case, the resulting set of negative examples, $\mathscr{E}^-$, becomes intractable in practice. Moreover, only a small subset $\hat{\mathscr{E}}^- \subset \mathscr{E}^-$ is effectively relevant for learning to extract the concept.
- *Low cardinality of positive examples*: The size of the whole set of positive examples, $\mathscr{E}^+$, can be small to accurately learn an extractor for a concept within the scenario. This could be the case when dealing with some combinations of document style and domain, in which no more training corpus is available.

https://doi.org/10.1017/S1351324902002863 Published online by Cambridge University Press

Some strategies could be applied to select a set of relevant negative examples, $\hat{\mathscr{E}}^-$, from the whole set of negative examples ($\mathscr{E}^-$):

- *Use of empirical observations.* Sometimes, all positive examples share a set of properties. $\hat{\mathscr{E}}^-$ can be generated by selecting those non positive examples sharing these properties. For instance, taking strings of words as examples, (Freitag 1998) computes the minimum and the maximum lengths in words of the positive examples and takes as negative ones those other examples whose length is between these values.
- *Use of a distance measure.* A more general approach consists in defining a distance measure between examples and selecting the negative examples that are closest to each positive ones. However, as a consequence of the extremely large size of $\mathscr{E}^-$, the bigger $\mathscr{E}^+$ is, the higher the cost to compute all distance values between positive and negative examples is, until it becomes prohibitive.
- *Use of clustering techniques.* In order to reduce the cost, in practice, of the second strategy, we can take into account that some positive examples can be very similar to others. Then, applying clustering techniques to $\mathscr{E}^+$, we can take *medoids*[8] as the most representative positive examples to compute the distances explained in the second strategy.

Initial experiments using a baseline approach following Freitag's are presented in section 4.1. In section 4.2, a clustering method is used to deal with our instance relevance problem, and results are compared with the baseline approach. Finally, in section 4.4, a method to deal with the low cardinality of $\mathscr{E}^+$ is described and compared with the baseline approach.

4.1 Initial experiments with FOIL

Initial experiments on the mycology domain have been presented in Turmo and Rodriguez (2000) (see Turmo (2002) for results covering the whole scenario). A corpus consisting of 78 mycology documents has been used (covering 4183 different words corresponding to 1361 lemmas). From this set, 65 documents have been used as training corpus, and 13 have been kept for testing.

Several experiments have been carried out with different training set sizes (taking 5, 15, 25, 35, 45, 55 and 65 documents). For concept *color*, there were 33, 105, 206, 270, 328, 398 and 480 positive examples within the different training set sizes, respectively, and 102 ones within the testing set. All of them have been manually marked and negative examples have been computed by using Freitag's method as explained before. The results of such experiments are presented in Table 11, in which recall (R), precision (P) and F-score (F) are those used in MUC-7 conferences. The table shows that using a low number of training examples (aproximately more than 100) is enough to achieve an F value greater than 89%. Although no direct comparison to other systems is possible due to the domain and language used, this

[8] A medoid is an actual data point representing a cluster in a similar way to a *centroid*.

https://doi.org/10.1017/S1351324902002863 Published online by Cambridge University Press

Table 1. *Results from learning* color *concept from different training set sizes*

Training set	$\mathscr{E}^+$	R	P	F
5	33	66·67	93·94	77·99
15	105	87·81	97·30	92·31
25	206	82·08	98·65	89·57
35	270	82·96	97·33	89·57
45	328	82·42	98·68	89·82
55	398	81·32	98·67	89·16
65	480	84·27	94·95	89·29

result can be considered state-of-the-art regarding those reported by IE systems covering tasks with similar difficulty.

4.2 Relevance of negative examples

Freitag's approach to generate negative examples is quite straightforward, although generates a lot of irrelevant examples. This is due to the fact that other properties out of surface realization are useful to determine the relevance of an example. In fact, the relevance of the examples depends on the relevance of the features and relations from which examples are represented. As a consequence, a more powerful mechanism is needed for such a purpose. In Turmo and Rodriguez (2001), a clustering method is presented as a more general method that deals with our instance selection problem. It consists in: (a) finding clusters of positive examples, (b) classifying negative examples according to distances to these clusters, and (c) selecting those negative examples being relevant enough for each cluster.

Clustering methods require the definition of a distance measure between examples. In our case, examples are represented using a relational model, and defining a distance measure within such a kind of representation is difficult. This is why we start by translating positive examples into vectors of features.

4.2.1 Representing positive examples as vectors of features

Positive examples $e^+ =< n_1, \ldots, n_k >$, where n_i is a terminal node, are transformed into vectors in the following form $\mathscr{P}(e^+) = (l_1, \ldots, l_{k-1}, \delta, \mathscr{L}, \mathscr{W}, \mathscr{S}_1, \ldots, \mathscr{S}_k)$, being:

- l_i the distance, in terminal nodes, between n_i and n_{i+1}.
- δ a number that codes parse subtrees and allows clustering them by taking only into account the syntactic tags involved (cf. Turmo *et al.* 1999). For instance, considering $\delta_{nc} = 2$, $\delta_{adj} = 7$ and $\delta_{gnom} = 53$, the parse trees subsumed by nodes n_5 and n_9 are coded as $\delta(n_5) = 742$ and $\delta(n_9) = 106$ by multiplying the δ codes of the involved tags (prime numbers).
- $\mathscr{L}$ is the set of lemmas in nodes n_i.
- $\mathscr{W}$ is the set of words in nodes n_i.

https://doi.org/10.1017/S1351324902002863 Published online by Cambridge University Press

– $\mathscr{S}_i$ is the set of all possible senses and hypernyms for the lemma occurring in n_i, preserving ambiguities.[9]

4.2.2 Defining a distance measure between examples

Due to the different nature of these attributes, an Heterogeneous Overlap-Euclidean Metric (HOEM) (Randall 1997) has been used with the aim of measuring the distance between examples. It has been defined as the euclidean distance among the following $2k + 2$ components:

$$\forall r \in [1, k-1] : d_r(\mathscr{P}(e_i), \mathscr{P}(e_j)) = |l_r^i - l_r^j| \tag{2}$$

$$d_k(\mathscr{P}(e_i), \mathscr{P}(e_j)) = \frac{max(\delta_i, \delta_j)}{min(\delta_i, \delta_j)} \tag{3}$$

$$d_{k+1}(\mathscr{P}(e_i), \mathscr{P}(e_j)) = 1 - \frac{|\mathscr{L}_i \cap \mathscr{L}_j|}{|\mathscr{L}_i \cup \mathscr{L}_j|} \tag{4}$$

$$d_{k+2}(\mathscr{P}(e_i), \mathscr{P}(e_j)) = 1 - \frac{|\mathscr{W}_i \cap \mathscr{W}_j|}{|\mathscr{W}_i \cup \mathscr{W}_j|} \tag{5}$$

$$\forall r \in [1, k] : d_{k+2+r}(\mathscr{P}(e_i), \mathscr{P}(e_j)) = min_{s \in \mathscr{S}_r^i,\ t \in \mathscr{S}_r^j}\{cd(s,t)\} \tag{6}$$

$|\cdot|$ represents the absolute value in (2) and the cardinality function in (4) and (5). Function $cd(s,t)$ in (6) is the conceptual distance between two synsets s and t in EWN defined in Agirre and Rigau 91996) as:

$$cd(s,t) = \sum_{c \in shortestpath(s,t)} \frac{1}{depth(c)} \tag{7}$$

from which the lower the concepts are in EWN, the closer they are. Both d_{k+1} and d_{k+2} are derived from Jaccards' coefficient of similarity between sets, defined as the quotient between intersection and union cardinalities.

4.2.3 Clustering positive examples

The most representative positive examples are found by applying an agglomerative clustering technique based on medoids. The result is a partition of positive examples into the set $\mathrm{E} = \{< \mathscr{E}_i^+, m_i >\}$, where $\mathscr{E}_i^+$ is the set of positive examples included in the i-th cluster, and m_i its medoid.

4.3 Classifying and selecting negative examples

Using the distance measure described above, distances between medoids and negative examples are computed. In this sense, negative examples are being multi-classified into clusters $\mathscr{E}_i^+$, and distances can be seen as measures of relevance.

To select the set of relevant negative examples, $\hat{\mathscr{E}}^-$, two guidelines could be

[9] This is why $\mathscr{S}_1, \ldots, \mathscr{S}_k$ are not clustered into a single set.

https://doi.org/10.1017/S1351324902002863 Published online by Cambridge University Press

applied: (a) the more similar the medoids are, the fewer negative examples have to be selected, and (b) the larger the size of $\mathscr{E}_i^+$ is, the larger is the number of negative examples from $\mathscr{E}^-$ to be selected. From the former guideline, the following set of negative examples can be taken as relevant enough for the ith cluster:

$$\hat{\mathscr{E}}_i^-(\alpha_i) = \{e^- \in \mathscr{E}^- \mid d(e^-, m_i) \leq \alpha_i\} \tag{8}$$

where α_i is the maximum distance value to consider a negative example relevant for the ith cluster. Different α_i-formulas have been experimented (cf. Turmo and Rodriguez 2001). Some of them define general measures of α_i for all clusters, such as the the distance average. Other formulas define specific measures of α_i for each cluster, such as the following one, which achieves the best results:

$$max_{j \leq |\mathrm{E}|}\{d(m_i, m_j)\} \tag{9}$$

From this background, two approaches have been studied to generate $\hat{\mathscr{E}}^-$:

1. Generating the union of sets $\hat{\mathscr{E}}_i^-(\alpha_i)$.
2. Computing a number of relevant negative examples N_i for each cluster, according to the size of $\mathscr{E}_i^+$, as follows:

$$N_i = \beta\,|\mathscr{E}_i^+| \qquad \beta = \frac{\sum_{i \leq |\mathrm{E}|} |\hat{\mathscr{E}}_i^-(\alpha_i)|}{|\mathrm{E}|} \tag{10}$$

where β is the average number of more relevant negative examples for any medoid, and by extension, for any positive example. Consequently, for each positive example within $\mathscr{E}_i^+$, we are taking β different negative examples from those closest to the i-th cluster.

The first approach combined with α_i-formula (9) achieved the best results, and it has been used to select $\hat{\mathscr{E}}^-$.

4.3.1 Results

$\hat{\mathscr{E}}^-$ for concept *color* has been computed for different sizes of training corpus as explained above. The results of learning from them are shown in Table 2. They demonstrate that such a technique improves Freitag's method in some aspects. A much smaller set of relevant negative examples is selected by using our method (an average reduction of 55.63%), and consequently, the learner runs faster. Moreover, the resulting F values show that better extractors are learned. Finally, comparing the number of rules, the extractors learned by using our approach are slightly more compact than those learned by using the baseline method (an average reduction of 15%).

4.4 Low cardinality of training sets

As explained in section 3.3, FOIL is based on a covering approach that stops learning either when no positive training examples remain to be covered, or when no rule can be learned from the remaining examples with enough accuracy. When this latter

https://doi.org/10.1017/S1351324902002863 Published online by Cambridge University Press

Table 2. *Test results from both Freitag's baseline and the selected method*

	Freitag's baseline					Selected method				
Size	$\lvert\hat{\mathscr{E}}^-\rvert$	R	P	F	Rules	$\lvert\hat{\mathscr{E}}^-\rvert$	R	P	F	Rules
5	2790	66·67	93·94	77·99	11	1155	89·89	94·12	91·96	8
15	7553	87·81	97·30	92·31	15	3501	90·24	94·87	92·50	12
25	13116	82·02	98·65	89·57	33	6363	82·02	100	90·12	25
35	18036	82·96	97·33	89·57	37	8657	85·23	96·15	90·36	33
45	29523	82·42	98·68	89·82	39	12465	84·62	97·47	90·59	37
55	36285	81·32	98·67	89·16	38	15241	83·70	97·47	90·06	35
65	42382	84·27	94·95	89·29	43	17846	88·89	98·77	93·57	40

condition occurs, the remaining examples are considered by FOIL as noisy data, and they become uncovered by the learned IE-rule set. However, these examples could be relevant evidences of the concept with little evidence on the training set. When this happens, we could use a bootstrapping approach by providing FOIL with more examples from new unnanotated training corpus, and retraining. Sometimes, however, no more corpus is available for training. Our bootstrapping approach is based on artificial examples, as described in Turmo and Rodriguez (2000). The algorithm is as follows:

1. Add a number of artificial examples to $\mathscr{E}^+$ and learn a new IE-rule set.
2. Drop out those new IE-rules that are repeated, those empirically subsumed[10] and those that only extract error values from the training set. Append the filtered new rules to the current ones.
3. Repeat from (1) until all positive examples are covered by the resulting rule set, or the F-score gain, measured with the training data, is smaller than or equal to a given threshold α.

Artificial examples are created in order to artificially increase the evidence of some properties of uncovered examples, so that general rules covering these examples will be more frequently learned than very specific ones. We have applied a method inspired by the generation of *convex pseudo data* (Breiman 1998), in which a process similar to *gen-combination* in genetic algorithms is used. This process, detailed in Turmo and Rodriguez (2000), can be paraphrased as follows: for each example uncovered by the current IE-rule set, a set of artificial examples is generated. Taking examples represented as vectors (as described in previous section), each artificial example is built by combining feature values from both the uncovered example vector and a randomly selected covered one.

Merging IE-rule sets by appending new rules allows improving recall, although

[10] A rule r_1 is empirically subsumed by another one, r_2, if examples covered by r_1 are also covered by r_2.

https://doi.org/10.1017/S1351324902002863 Published online by Cambridge University Press

Table 3. *Results from adding artificial examples to the initial training set with 35 documents*

Training set	$\mathscr{E}^+$	$F_{training}$	R	P	F_{test}
35_0	270	94·51	82·96	97·33	89·57
35_1	410	97·13	87·50	97·47	92·22
35_2	485	97·53	87·50	97·47	92·22

precision could decrease. This is why small sets of artificial examples are preferred to be added at each iteration.

We have applied such an incremental learning approach to improve the results achieved by the baseline one for concept *color*. This was tested with the training set corresponding to 35 documents, so as to allow better comparisons with higher training set sizes. Results are shown in Table 3. Using the baseline approach, 28 positive examples remained uncovered. Taking α=1 (a relatively high value), and generating five artificial examples per uncovered case, two iterations were executed, 35_1 and 35_2. The results for the testing set were improved in 4.54 points for recall, 0.14 for precision, and 2.65 for F. Moreover, comparing them with those in Table 1, F improves the results achieved by learning from 480 real training examples. Taking into account that we are learning from artificial examples, the results are considerably good.

5 Comparisons with related learning systems

Direct comparisons among systems are possible when they have been tested against the same set of extraction tasks and the same testing set, as in MUC competitions. EVIUS has been evaluated in a different domain and language. This fact makes difficult to compare our system to others. In spite of this, similarities among systems may be found by comparing their methodologies. Following, general comparisons among IE-rule learning systems are described from different viewpoints.

5.1 Document styles

AutoSlog, CRYSTAL, PALKA and LIEP can learn IE rules from free text. On the contrary, RAPIER, SRV and WHISK can deal with more structured documents, such as Web pages. Webfoot extends CRISTAL in order to learn from Web pages. EVIUS can be also used to learn from this documents. This is due to (1) the Preprocessor of TURBIO also allows the partition of documents into HTML-tagged fragments, and (2) EVIUS uses an open predicate model, like SRV, which is independent from the learning methodology, so HTML tags can be represented with new predicates, as in the case of syntax and semantics, among others.

https://doi.org/10.1017/S1351324902002863 Published online by Cambridge University Press

5.2 Applicability of learned rules

RAPIER, SRV and AutoSlog only allow learning rules that extract values for a slot (single-slot rules). In MUC terms, this fact implies that the extraction of relations among entities (relationships and events) cannot be learned by such systems. However, LIEP only learns rules that extract values for a set of slots within a template (multi-slot rules), while WHISK, CRYSTAL and PALKA can learn both kinds of rules, as EVIUS does. EVIUS learns single-slot rules for basic concepts within the scenario, and multi-slot rules for derived concepts.

5.3 Generality of learned rules

EVIUS learns rules that extract exact slot-filler values, as RAPIER, SRV, WHISK and LIEP do. However, AutoSlog, PALKA and CRYSTAL locate syntactic constituents containing the exact value, but some further process is required to correctly extract them. Moreover, when EVIUS learns to fill slots, it can impose semantic constraints (e.g. belonging to a EWN synset) not only on the nodes aimed at filling slots, but on the context of these nodes. This is also the case of CRYSTAL, SRV, RAPIER and WHISK. However, they differ from the kind of semantic class generalization they perform in:

- SRV and RAPIER allow semantic class generalization by using a wide coverage semantic hierarchy (WordNet), as EVIUS does by using EWN. However, unlike EVIUS, these systems work by previously disambiguating word senses. In both cases, the most frequent word sense, according to WordNet, is selected. This selection is a strong bias for the learning system, and it can be a wrong strategy because (a) dealing with restricted domains, it is often the case that the correct word sense is very infrequent with respect to a general context, and (b) the same word can occur in different documents with different senses.
- CRYSTAL and PALKA use a specific semantic hierarchy for the domain to perform the semantic generalization. This fact implies that probably a new semantic hierarchy has to be defined when dealing with a new domain.
- In WHISK and AutoSlog, no semantic generalization is performed. This means that the resulting rules can only match new examples owning the same semantic constraints as those used to learn, and consequently a lower coverage could be achieved. This fact occurs also with the semantic generalization performed by LIEP, which is based on defining disjunctions of semantic constraints.

Finally, AutoSlog, PALKA and CRYSTAL always require syntactic information about constituents, like LIEP and EVIUS, but they do not allow syntactic generalizations. All three systems work within the context of propositional learning, and use a form of propositionalization of syntactic relations among constituents involved in training examples. It means that the syntactic contexts of the slots to be extracted are fixed, and they cannot be generalized.

https://doi.org/10.1017/S1351324902002863 Published online by Cambridge University Press

5.4 On the supervision of the learning methodology

On the one hand, IE rule learning systems need a set of positive examples to learn IE rules for a concept within the scenario of extraction. Traditionally, they are manually annotated within the training documents. Only a few research groups have investigated less expensive approaches to achieve such positive examples (e.g. HASTEN, AutoSlog-TS, and the approaches in Basili *et al.* (2000) and Yangarber (2000)). Our approach is based on taking profit of constructive learning to reduce the annotation task. When learning single-slot rules for a concept, EVIUS needs the manual annotation process to select positive examples. However, when multi-slot rules have to be learned, which is frequent in many scenarios, the MCL approach used by EVIUS allows reducing the manual annotation process to a manual validation process, which requires less effort than traditional one.

On the other hand, increasing the number of positive examples with artificial ones to improve results does not seem to have been approached before this work. This kind of examples are generated in an unsupervised way.

Finally, EVIUS requires also negative examples to proceed, as SRV does. Freitag generates them by using a supervised method based on empirical observations, within the framework of SRV. However, EVIUS uses a more effective, unsupervised approach based on a clustering method.

6 Conclusions and future work

Most existing systems to learn rules for IE are single concept learning systems. As a consequence, each concept within the scenario of extraction is learned in an independent manner, as a rule set. This paper has focused on the description of EVIUS, a multi-concept relational learning system to incrementally learn to extract the whole scenario of extraction from free text. It is based on a multistrategy constructive learning approach in which FOIL is involved. Our approach (1) takes into account new knowledge related to already learned concepts for further learning, and (2) reduces the annotation of positive examples within training corpus to only some concepts, and requires the user to validate automatically generated candidates to be positive examples for the remaining concepts.

In addition, different techniques have been experimented and integrated within EVIUS to achieve a better training set for learning to extract a concept with FOIL. On the one hand, we have demonstrated that incrementally learning by adding artificial examples improves the accuracy of some IE-rule sets that have been learned from insufficient size of available training corpora. On the other hand, we have experimented with different approaches to provide FOIL with a tractable and appropriate set of negative examples. The resulting set from applying the closed-world assumption becomes intractable to FOIL. We have compared the Freitag's selection approach, based on empirical observations from the positive examples, with our proposal, based on clustering positive examples. The results demonstrate that our approach allows FOIL to learn more accurated and compact IE-rule sets than Freitag's, with a lower number of negative examples.

https://doi.org/10.1017/S1351324902002863 Published online by Cambridge University Press

Future work on these techniques will experiment on both measuring the number of artificial examples needed to better learn at each iteration, and applying other instance selection approaches. We continue working on the mycology domain extending the extraction to all information occurring in the documents. In addition, we are also interested in applying TURBIO to other domains and languages. In particular, we have started working on extracting information from Web pages related to the domain of seminar announcements.

References

Agirre, E. and Rigau, G. (1996) Word sense disambiguation using conceptual density. *Proceedings 16th International Conference on Computational Linguistics*, Copenhagen, Denmark.

Aone, C. and Bennet, W. (1996) Evaluation automated and manual acquisition of anaphora resolution. In: Riloff, E., Wermter, S. and Scheler, G. (eds.), *Lecture Notes in Artificial Intelligence 1040*. Springer-Verlag.

Aseltine, J. H. (1999) WAVE: An incremental algorithm for information extraction. *Proceedings AAAI Worshop on Machine Learning for Information Extraction.*

Baluja, S., Mittal, V. and Sukthankar, R. (1999) Applying machine learning for high performance named-entity extraction. *Proceedings International Conference of Pacific Association for Computational Linguistics.*

Basili, R., Pazienza, M. T. and Vindigni M. (2000) Corpus-driven learning of event recognition rules. *Proceedings ECAI Workshop on Machine Learning for Information Extraction.*

Borthwick, A., Sterling, J., Agichtein, E. and Grishman, R. (1998) Exploiting diverse knowledge sources via maximum entropy in named entity recognition. *Proceedings Sixth ACL Workshop on Very Large Corpora.*

Borthwick, A. (1999) A maximum entropy approach to named entity recognition. PhD thesis, Computer Science Department, New York University.

Breiman, L. (1998) Arcing classifiers. *Ann. Stat*. **26**: 801–849.

Califf, M.E. (1998) Relational Learning Techniques for Natural Language Information Extraction. PhD thesis, University of Texas at Austin.

Cardie, C. and Wagstaff, K. (1999) Noun phrase coreference as clustering. *Proceedings Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora.*

Cardie, C., Daelemans, W., Nédellec, C. and Sang, T. S. E. (2000) *Proceedings Fourth Conference on Computational Natural Language Learning*.

Chai, J. Y., Biermann, A. W. and Guinn, C. I. (1999) Two dimensional generalization in information extraction. *Proceedings 11th AAAI National Conference on Artificial Intelligence.*

Craven, M. (1999) Learning to extract relations from MEDLINE. *Proceedings AAAI Workshop on Machine Learning for Information Extraction.*

Freitag, D. (1998) Machine Learning for Information Extraction in Informal Domains. PhD thesis, Computer Science Department, Carnegie Mellon University.

Freitag, D. and Kushmerick, N. (2000) Boosted wrapper induction. *Proceedings ECAI Workshop on Machine Learning for Information Extraction.*

Freitag, D. and McCallum, A. (2000) Information extraction with HMM structures learned by stochastic optimization. *Proceedings 21st National Conference on Artificial Intelligence.*

Glasgow, B., Mandell, A., Binney, D., Ghemri, L. and Fisher, D. (1998) MITA: An information extraction approach to analyses of free-form text in life insurance applications. *Artif. Intell.* **19**:59–72.

Harabagiu, S. M. and Maiorano, S. J. (2000) Acquisition of linguistic patterns for knowledge-based information extraction. *Proceedings Third International Conference on Language Resources and Evaluation.*

https://doi.org/10.1017/S1351324902002863 Published online by Cambridge University Press

Hobbs, J. (1993) The generic information extraction system. *Proceedings 5th Message Understanding Conference*, pp. 87–92.

Huffman, S. (1995) Learning information extraction patterns from examples. *Proceedings IJCAI Workshop on New Approaches to Learn for NLP*.

Hsu, C.-N. and Dung, N.-T. (1998) Learning semistructured Web pages with finite-state transducers. *Proceedings Conference on Automated Learning and Discovering*.

Kim, J. and Moldovan, D. (1995) Acquisition of linguistic patterns for knowledge-based information extraction. *IEEE Trans. Knowl. & Data Eng.*

Ko, H. (1998) Empirical assembly sequence planning: A multistrategy constructive learning approach. In: Michalsky, R. S., Bratko, I. and Kubat, M. (eds.), *Machine Learning and Data Mining*. Wiley.

Krupka, G. R. (1995) Description of the SRA system used for MUC-6. *Proceedings Sixth Message Understanding Conference*.

Kushmerick, N. (2000) Wrapper induction: Efficiency and expressiveness. *Artif. Intell.* Extract, PhD Thesis, University of Washington.

Manning, C. and Schütze, H. (1999) *Foundations of Statistical Natural Language Processing*. MIT Press.

McCallum, A., Freitag, D. and Pereira, F. (2000) Maximum entropy Markov models for information extraction and segmentation. *Proceedings 17th International Conference on Machine Learning*.

McCarthy, J. F. and Lehnert, W. G. (1995) Using decision trees for coreference resolution. *Proceedings 14th International Join Conference on Artificial Intelligence*.

Michalski, R. S. (1993) Towards a unified theory of learning: Multistrategy task-adaptive learning. In: Buchanan, B. G. and Wilkins, D. (eds.), *Readings in Knowledge Acquisition and Learning*. Morgan Kauffman.

Mitchell, T. (1997) *Machine Learning*. McGraw-Hill.

Mitkov, R. (1998) Robust pronoun resolution with limited knowledge. *Proceedings Joint 17th International Conference on Computational Linguistics* and *36th Annual Meeting of the Association for Computational Linguistics*, pp. 869–875. Montreal, Canada.

Mooney, R. J. and Cardie, C. (1999) Symbolic machine learning for natural language processing. *AAAI Workshop on Machine Learning for Information Extraction.*

Muslea, I., Milton, S. and Knoblock, C. (2000) Hierarchical wrapper induction for semistructured information sources. *Autonomous Agents and Multi-Agent Systems.*

Quinlan, J. R. (1990) Learning logical definitions from relations. *Machine Learning* **5/3**: 239–266.

Randall, D. and Martínez, T. (1997) Improved heterogeneous distance functions. *Artif. Intell.* **6**: 1–34.

Riloff, E. (1996) Automatically generating extraction patterns from untagged texts. *Proceedings 13th National Conference on Artificial Intelligence*.

Riloff, E. and Jones, R. (1999) Learning dictionaries for information extraction by multi-level bootstrapping. *Proceedings 16th National Conference on Artificial Intelligence.*

Roth, D. and Yih, W. (2001) Relational learning via propositional algorithms: An information extraction case study. *Proceedings 15th International Conference On Artificial Intelligence.*

Seymore, K., McCallum, A. and Rosenfeld, R. (1999) Learning Hidden Markov Model structure for information extraction. *Proceedings 20th National Conference on Artificial Intelligence.*

Soderland, S. (1999) Learning information extraction rules for semi-structured and free text. *Machine Learning* **34**: 233–272.

Soderland, S., Fisher, D., Aseltine, J. and Lehnert, W. (1995) CRYSTAL: Inducing a conceptual dictionary. *Proceedings 14th International Joint Conference on Artificial Intelligence*, pp. 1314–1321.

Turmo, J., Català, N. and Rodríguez, H. (1999) An adaptable IE system to new domains. *Appl. Intell.* **10**: 225–246.

https://doi.org/10.1017/S1351324902002863 Published online by Cambridge University Press

Turmo, J. and Rodríguez, H. (2000) Learning IE-rules for a set of concepts. *Proceedings IV Conference on Computational Natural Language Learning*, pp. 115–118.
Turmo, J. and Rodríguez, R. (2001) Selecting a relevant set of examples to learn IE-rules. *Proceedings IEA/AIE Conference*.
Turmo, J. (2002) An information extraction system portable to new domains. PhD thesis, Universitat Politècnica de Catalunya. Spain. Forthcoming.
Yangarber, R. (2000) Scenario customization of information extraction. PhD thesis, Courant Institute of Mathematical Sciences, New York University.
Yangarber, R. and Grishman, R. (1998) Description of the PROTEUS/PET system as used for MUC-7. *Proceedings Seventh Message Understanding Conference*.
Young, S. and Bloothooft, G. (eds.) (1997) *Corpus-Based Methods in Language and Speech Processing*. Kluwer Academic.

https://doi.org/10.1017/S1351324902002863 Published online by Cambridge University Press