Exploiting ASP for Semantic Information Extraction

Massimo Ruffolo[1][3], Nicola Leone[1][4]*, Marco Manna[2], Domenico Saccà[1][2][3], and Amedeo Zavatto[2]

[1] Exeura s.r.l.
[2] DEIS - Department of Electronics, Computer Science and Systems
[3] ICAR-CNR - Institute of High Performance Computing and Networking of the Italian National Research Council
[4] Department of Mathematics
University of Calabria, 87036 Arcavacata di Rende (CS), Italy
e-mail: {ruffolo,leone,manna,sacca,zavatto}@exeura.it
WWW home page: http://www.exeura.it

Abstract. The paper describes HιLεX, a new ASP-based system for the extraction of information from unstructured documents. Unlike previous systems, which are mainly syntactic, HιLεX combines both semantic and syntactic knowledge for a powerful information extraction. In particular, the exploitation of background knowledge, stored in a domain ontology, allows to empower significantly the information extraction mechanisms. HιLεX is founded on a new two-dimensional representation of documents, and heavily exploits DLP^+– an extension of disjunctive logic programming for ontology representation and reasoning which has been recently implemented on top of DLV. The domain ontology is represented in DLP^+, and the extraction patterns are encoded by DLP^+ reasoning modules, whose execution yields the actual extraction of information from the input document. HιLεX allows to extract information from both HTML and flat text documents.

1 Introduction

Existing systems for storing unstructured information such as document repositories, digital libraries, and Web sites, are sources of a huge amount of digital contents organized as HTML pages or flat text documents. Such repositories tend to be practically useless because of the vastness of the information they hold combined with the lack of available powerful and expressive manipulation techniques. Moreover, they can only manage data, rather than the conveyed actual knowledge. Recognizing and extracting relevant information automatically from these rapidly changing sources according to their semantics is an important problem.

In the recent literature a number of approaches for information extraction from unstructured documents have been proposed. An overview of the large body of existing literature and systems is given in [1,2,3]. Existing systems are mainly purely syntactic, and they are not aware of the semantic of the information they extract.

* This work was supported by the European Commission under project IST-2001-37004 WASP.

In this work we present $\mathsf{H\imath L\varepsilon X}$, a logic-based system which combines both syntactic and semantic knowledge for a powerful information extraction from unstructured documents. Logic-based approaches for information extraction are not new [10,11], however, the approach we propose is original. Its novelty is due to:

- The two-dimensional representation of an unstructured document. A document is viewed as a Cartesian plane composed by a set of nested rectangular regions called *portions*. Each portion, univocally identified through the cartesian coordinates of two opposite vertices, contains a piece of the input document.
- The exploitation of an ASP-based knowledge representation language called DLP^+, extending DLP [4] with object-oriented features, including classes, (multiple) inheritance, complex objects, types, which is well-suited for representation and powerful reasoning on ontologies. This language is supported by the DLV^+ system [5], implemented on top of DLV [6,7,8,9].
- The use of an ontology, encoded in DLP^+, describing the domain of the input document. A concept of the domain is represented by a DLP^+ class; each class instance is a *pattern* representing a possible way of writing the concept.
- The employment of a new grammar, named $\mathsf{H\imath L\varepsilon X}$ grammar, for specifying the (above mentioned) patterns. $\mathsf{H\imath L\varepsilon X}$ grammar extends regular expressions for the representation of two-dimensional patterns (like tables, item lists, etc.), which often occur in web pages. The patterns are specified through DLP^+ rules, whose execution yields the *semantic information extraction*, by associating (the part of the document embraced by) each portion to an element of the domain ontology.

It is worthwhile noting that, besides the domain ontology, $\mathsf{H\imath L\varepsilon X}$ system uses also a *core ontology*, containing (patterns for the extraction of) general concepts (like, e.g., date, time, numbers, email, etc.) which are not bounded to a specific domain but occur generally.
The advantages of the $\mathsf{H\imath L\varepsilon X}$ system over other existing approaches are mainly the following:

- The extraction of information according to their semantic and not only on the base of their syntactic structure (as in the previous approaches).
- The possibility to extract information in the same way from documents in different formats. The same wrapper can be used to extract data from both flat text and HTML documents. Importantly, this is not obtained by a preliminary HTML-to-text translation; but it comes automatically thanks to higher abstraction due to the view of the input document as a set of logical portions.
- The possibility to obtain a “semantic” classification of the input documents, which is much more accurate and meaningful than the syntactic classifications provided by existing systems (mainly based on counting the number of occurrences of some keywords), and opens the door to many relevant applications (e.g., emails classification and filtering, skills classification from curricula, extraction of relevant information from medical records, etc.).

Distinctive features of the novel semantic approach to information extraction implemented in the $\mathsf{H\imath L\varepsilon X}$ system, summarized above, allows a better digital contents management and fruition in different application field such as: e-entertainment, e-health, e-commerce, e-government, e-business.

The remainder of this work is organized as follows: section 2 describes the DLP^+ knowledge representation language, section 3 shows the two-dimensional document representation idea and the HιLεX two-dimensional pattern specification grammar, section 4 describes how ontologies are represented in the HιLεX system, section 5 shows architecture and functionalities of the system, finally, section 6 describes the application of the semantic information extraction approach to web pages and flat text documents.

2 The Knowledge Representation Language DLP^+

The semantic data extraction approach implemented in the HιLεX system is based on the formal representation of information to be recognized and extracted from digital documents using the DLP^+ ontology representation language implemented on the DLV^+ system, a cross-platform development environment for knowledge modeling and advanced knowledge-based reasoning. The DLV^+ system allows to easily develop real world complex applications and allows to perform advanced reasoning tasks in a user friendly visual environment. The DLV^+ system seamlessly integrates the DLV system exploiting the power of a stable and efficient ASP solver.

A strong point of the system is its powerful language, extending DLP by object-oriented features. In particular, the language includes, besides the concept of **relations**, the object-oriented notions of **classes**, **objects** (class instances), **object-identity**, **complex-objects**, **(multiple) inheritance**, and the concept of modular programming by mean of **reasoning modules**.

A *class* can be thought of as a collection of individuals that belong together because they share some properties. An individual, or *object*, is any identifiable entity in the universe of discourse. Objects, also called class instances, are unambiguously identified by their object-identifier (oid) and belong to a class. A class is defined by a name (which is unique) and an ordered list of attributes, identifying the properties of its instances. Each attribute is identified by a name and can be of class type. This allows the specification of *complex objects* (objects made of other objects).

Classes can be organized in a specialization hierarchy (or data-type taxonomy) using the built-in *is-a* relation (*multiple inheritance*). Relationships among objects are represented by means of *relations*, which, like classes, are defined by a (unique) name and an ordered list of attributes (with name and type). Importantly, DLP^+ supports two kind of relations, *base relations*, corresponding to basic facts (that can be stored in a database), and *derived relations* corresponding to facts that can be inferred by logic programs.

As in DLP, logic programs are a set of logic rules and constraints. However, DLP^+extends the definition of logic atom introducing class and relation predicates and complex terms, allowing a direct access to object properties. This way, DLP^+ rules merge, in a simple and natural way, the declarative style of logic programming with the navigational style of the object-oriented systems. In addition, DLP^+ logic programs are organized in *reasoning modules*, taking advantage of the benefits of modular programming.

Moreover, DLV^+ offers several important facilities driving the development of both the knowledge base and the reasoning modules. Using DLV^+, developers and domain experts can create, edit, navigate and query object-oriented knowledge bases by an easy-to-use **visual environment**, enriched by a full graphic **query interface** à la QBE.

Classes can be declared in DLV^{+} by using the keyword `class` followed by the class name and by a comma separated list of attributes. Each attribute is a couple `(attribute-name:attribute-type)`.

The attribute-type is either a user-defined class name, or a built-in class name (in order to deal with concrete data types, DLP^{+} features two built-in classes *string* and *integer*). For instance, the class faculty with an argument of type string can be declared as follows:

```
class faculty(name:string).
```

Objects, that is class instances, are declared by asserting new facts. An instance for the class faculty, can be declared as follows:

```
#01:faculty(name:"Faculty of Science").
```

Here, the string "Faculty of science" values the attribute name; while #01 is the *object-identifier (oid)* of this instance (each instance is equipped by a unique oid).

The possibility to specify user-defined classes as attribute types allows for complex objects, i.e. objects made of other objects. The following declaration of class person includes, besides name and age, two attributes of type person, namely, father and partner.

```
class person(name:string,age:integer,father:person,partner:person).
```

Note that this declaration is "recursive" (father and partner are of type person). A couple of partners can be specified as follows:

```
#2:person(name:"Max", age:30, father:person(name:"Peter"),partner:#3).
#3:person(name:"Mary", age:28, father:#6, partner:#2).
```

Note that "Max" (identified by #02) is "Mary's" partner and vice versa ("Mary" is identified by #03). Moreover, arguments are identified by name, allowing for an easier way to access attributes. Instance arguments can be valued both by specifying object identifiers and by using a nested class predicate (complex term) which works like a function (e.g., Max's father is specified by a complex term above).

Classes can be organized in a taxonomy by using the *isa* relation. For example, the following student class extends the person class by two new attributes, code and enrol.

```
class student isa {person} (code:string,enrol:faculty).
```

Instances of the class student are declared as usual, by asserting new facts.

```
#6:student(name:"John",age:20,father:#7,partner:#7,code:0, enrol:#1).
#7:student(name:"Alice",age:20,father:#7,partner:#6,code:1, enrol:#1).
```

Like in common object-oriented languages with inheritance, each instance of a subclass becomes, automatically, an instance of all super classes (isa relation induces an inclusion relation between classes). Moreover, sub-classes inherit attributes from all super-classes.

The language provides a built-in most general class named *Object* that is the class of all individuals and is a superclass of all DLP^{+} classes.

Also multiple inheritance is supported. For example, class student-employee, declared next, inherits from both class student and class employee.

```
class employee isa {person} (salary:integer).
class stud_emp isa {student,employee} (spouse:employee).
```

Relations represent relationships among objects. Base relations are declared like classes and tuples are specified (as usual) asserting a set of facts (but they are not equipped with an oid). For instance, the base relation colleague, and a tuple asserting that “Alice” and ”Mary” are colleague, can be declared as follows:

```
relation colleague(p1:person,p2:person).
colleague(p1:person(name:"Mary"),person(p2:name:"Alice")).
```

Classes and base relations are, from a data-base point of view, the extensional part of the DLP^+ language. Conversely, derived relation are the intensional (deductive) part of the language and are specified by using reasoning modules. Reasoning modules, like DLP programs, are composed by logic rules and integrity constraints. DLP^+ reasoning modules allows one to exploits the full power of DLP. As an example, consider the following module, encoding the team-building problem (compute a team satisfying some project restrictions).

```
class project(numEmp:integer, numSk:integer, budget:integer, maxSal:integer).

module(teamBuilding){
 inTeam(E) v outTeam(E) :- E:employee().
 :- project(numEmp:N),not #count{E: inTeam(employee:E)}=N.
 :- project(numSk:S),not #count{Sk: E:employee(skill:Sk),inTeam(E)}>=S.
 :- project(budget:B),not #sum{Sa,E: E:employee(salary:Sa),inTeam(E)}<=B.
 :- project(maxSal:M),not #max{Sa:E:employee(salary:Sa),inTeam(E)}<=M. }
```

Eventually, in order to check the consistency of a knowledge base the user can specify global integrity constraints called axioms.

In the H$\imath$LεX system DLP^+ is exploited as ontology representation language for expressing the semantic of information to be extracted from unstructured documents. Moreover, reasoning modules providing modular programming facilities allows the definition of the logic-based pattern recognition approach implemented in the H$\imath$LεX system.

3 The H$\imath$LεX Two-dimensional Grammar

The semantic information extraction approach implemented in the H$\imath$LεX system is based on the main notion of two-dimensional structure of a document. This notion is founded on the idea that an unstructured document can be considered as a Cartesian plane composed by a set of nested rectangular regions called *portions*. Each region, univocally identified through the cartesian coordinates of two opposite vertices, contains a piece of the input document representing an element of the information to be extracted. Elements, organized in a document according to syntactic, presentation and semantic rules of a language recognizable by a human reader, can be: *simple* such as characters, words (classified using its part-of-speech tag recognized using natural language techniques), table cells; *complex* such as phrases, item lists, tables, paragraphs, text boxes, obtained as composition of other simple or complex elements.

Elements are represented in a DLP^+ ontology as classes whose instances hold extraction patterns useful to recognize them in a document. When an element is recognized the relative portion is annotated w.r.t. the ontology class.

To better explain the idea of portion in figure 1 is depicted a sequence of portions containing simple elements: the word ”knowledge” [(x_1, y_1),(x_2, y_2)], the character ”blank_char” [(x_2, y_1),(x_3, y_2)], the word ”management” [(x_3, y_1), (x_4, y_2)] and so on. Since portions can be nested, the portion containing the complex element representing the concept of ”knowledge management system” can be identified between the points [(x_1, y_1),(x_6, y_2)].

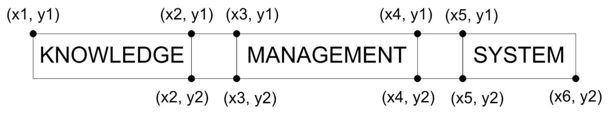


Fig. 1. Example of `portions`

Representing portions using the following DLP^+ class and relation:

```
class point (x: integer, y: integer).
relation portion (p: point, q: point, elem:
  element).
```

the *logic two-dimensional representation* of an unstructured document is obtained. Each instance of the relation *portion* represents the relative document region. It relates the two points identifying the region, expressed as instances of the class *point*, and an ontology element, expressed as instance of the class *element*. This DLP^+ encoding allows to exploit the two-dimensional document representation on which the semantic information extraction approach proposed in this paper is based on.

In the H$\imath$LεX system the extraction patterns are internally represented by means the H$\imath$LεX two-dimensional pattern representation grammar. This grammar is founded on two-dimensional grammars (also called picture languages) [12] and has an high expressive power allowing the definition of very expressive target patterns. By H$\imath$LεX grammar information to be extracted can be expressed in term of a pattern describing complex two-dimensional composition of elements defined in the ontology. The syntax of the H$\imath$LεX two-dimensional grammar is presented in the following.

```
NEW_ELEMENT → GENERALIZATION | RECURRENCE | CHAIN | TABLE
GENERALIZATION → GEN1 |GEN2 | GEN3
GEN1 → generalizationOf (arg: ARG1)
GEN2 → orContain_generalizationOf (arg: ARG1, inArg: ARG1, condition:
CND)
GEN3 → andContain_generalizationOf (arg: ARG1, inArg: ARG1, condition: CND)
CND → coincident | notCoincident
RECURRENCE → recurenceOf (arg: ARG3, range: RANGE, dir: DIR)
CHAIN → CHAIN1 (arg: ARG2, dir: DIR, sep: SEP)
```

```
CHAIN1 → sequenceOf | permutationOf
TABLE → TAB1 (arg: ARG2, range: RANGE, dir: DIR, sep: SEP)
TAB1 → sequenceTableOf | permutationTableOf
ARG1 → ARG2 | ARG3
ARG2 → [ LIST ]
ARG3 → BASE_ELEM
LIST → ARG3 , ARG3 LIST1
LIST1 → , ARG3 LIST1 | ε
RANGE → < NUM , NUM > | NUM | + | *
DIR → vertical | horizontal | both
SEP → ARG3 | null
```

Each NEW_ELEMENT is a pattern representing an instance of an element of the ontology obtained as a composition of other instances (BASE_ELEM) according to the semantic of the language constructs. Roughly speaking the semantic of each construct generating NEW_ELEMENT is:

– GENERALIZATION. A NEW_ELEMENT obtained from this operator constitutes a generalization of a single BASE_ELEM or a list of them;
– RECURRENCE. A NEW_ELEMENT obtained from the recurenceOf operator is composed by a concatenation of a fixed number of the same element in a given direction;
– CHAIN. A NEW_ELEMENT obtained from the sequenceOf operator is composed by a concatenation of at least two elements in a given direction which order is fixed, whereas, a NEW_ELEMENT obtained from the permutationOf operator is composed by a concatenation of at least two elements in a given direction, in this case elements are unordered;
– TABLE. A NEW_ELEMENT obtained from the sequenceTableOf operator is composed by elements having a fixed composition along a direction repeated a certain number of times along the other direction, whereas, a NEW_ELEMENT obtained from the permutationTableOf operator is composed by unordered elements along a direction repeated with the same structure a fixed number of times along the other direction. This construct allows to recognize table in HTML and text documents.

An example of HιLεX bi-dimensional pattern is:

```
knowledge_management_system → sequenceOf(arg:
  [knowledge, management, system], dir: horizontal,
  sep: blank_char).
```

In this pattern ”knowledge”, ”blank_char”, ”management”, and ”system” are the identifiers of class instances defined in the ontology. Intuitively, this pattern means that the concept of ”knowledge management system” can be recognized in a document when there are three portions distributed along the horizontal direction, the first containing the element ”knowledge”, the second containing the element ”management”, and the third containing the element ”system” separated by a portion containing one or more ”blank_char”. In order to extract this information the pattern is first translated into a set of DLP^+ rules exploiting the logic two-dimensional document representation, involved in a reasoning modules and then executed on a pattern recognizer implemented on the DLV^+ system.

4 HıLεX Ontologies

In the HıLεX system the semantic knowledge is encoded by means of DLP^+ ontologies. The ontologies are divided into *core ontology* and *domain ontologies*, and have a common root, called *element*:

```
class element (type: expression_type,
   expression: string, label: string).
```

The three attributes have the following meaning:

– `expression`: holds a string representing the pattern specified by regular expressions or by HıLεX two-dimensional language according to the `type` property. Patterns contained in this attributes are used to recognize the element in a document, in particular, patterns constituted by means of regular expressions characterize simple elements and are used during document preprocessing by a pattern matcher (Document Analyzer), whereas, patterns represented through HıLεX grammar characterize complex elements and are used by a pattern recognizer able to exploit their expressive power.
– `type`: defines the type of the expression (i.e. `regexp_type`, `hilex_type`).
– `label`: contains a description of the element in natural language.

In the following the structure of core and domain ontologies are described in details.

The Core Ontology. This ontology is composed of three sections. The first section represents general simple elements describing a language (e.g. alphabet symbols, lemmas, Part-of-Speech, regular forms such as date, e-mail, etc.). In this section lemmas of a language are represented using the same semantic relation used in WordNet [13]. The second section represents elements describing presentation styles (e.g. font types, font styles, font colors, background colors, etc.). The third section represents elements obtained from HTML structures (e.g. table cells, table columns, table rows, paragraphs, item lists, texture images, text lines, etc.). The core ontology is organized as the classes hierarchy shown in the following:

```
class linguistic_element isa {element}.
    class character isa {linguistic_element}.
        class number_character isa {character}.
        ...
    class regular_form isa {linguistic_element}.
        class float_number isa {regular_form}.
        ...
    class italian_lexical_element isa {linguistic_element}.
```

```
        class english_lexical_element isa {linguistic_element}.
            class english_lemma isa {english_lexical_element}.
            ...
        class spanish_lexical_element isa {linguistic_element}.
        ...
        class separator isa {linguistic_element}.
        ...
    class presentation_element isa {element}.
        class font_type isa {presentation_element}.
        ...
    class structural_element isa {element}.
        class table_cell isa {structural_element}.
        ...
```

Examples of instances of these classes are:

```
mining: english_lemma (type: regexp_type, expression:
   ‘‘mining’’).
knowledge: english_lemma (type: regexp_type, expression: ‘‘knowledge’’).
management: english_lemma (type: regexp_type, expression: ‘‘management’’).
system: english_lemma (type: regexp_type, expression: ‘‘system’’).

float_number: number (type: regexp_type, expression:
   "(\\d{1,3}(?>.\\d{3})*,\\d+)").
```

Domain Ontologies. These ontologies contain simple and complex elements related to a specific knowledge domain. The distinction between core and domain ontologies allow to describe knowledge in a modular way. When a user need to extract data from document regarding a specific domain he can use only the related domain ontology. The modularization improve the extraction process in terms of precision and overall performances. For example, elements representing concepts related to the information technology domain can be organized as follows:

```
class information_technology_domain isa {element}.
    class knowledge_management_domain isa
        {information_technology_domain}.
        class knowledge_audit isa
            {knowledge_management_domain}.
        class knowledge_management isa
            {knowledge_management_domain}.
        class knowledge_management_system isa
            {knowledge_management_domain}.
    class knowledge_discovery_domain isa
        {information_technology_domain}.
        class data_mining isa {knowledge_discovery_domain}.
        class log_mining isa {knowledge_discovery_domain}.
```

Examples of domain specific class instances are:

```
knowledge_management_system_00: knowledge_management_system (type:
   regexp_type, expression: "KMS").

knowledge_management_system_01: knowledge_management_system (type:
   hilex_type, expression: ‘‘ sequenceOf(arg: [knowledge, management,
   system], dir: horizontal, sep: blank_char)’’).
```

From the above examples is possible to note that patterns represented through the HιLεX grammar contain names of instance defined in the ontologies. This characteristic of HιLεX grammars allows the composition of elements aimed to obtain more complex concepts.

5 Architecture and Functionalities of the HιLεX system

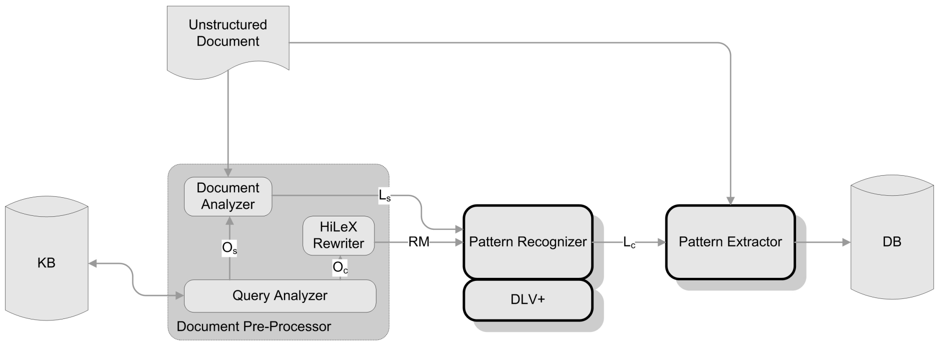


Fig. 2. The Architecture of the HιLεX System

The architecture of the HιLεX system is represented in figure 2. The Knowledge Base (KB) stores the ontologies described in the previous section by the DLV^+ system persistency layer. The information extraction process is executed in three main steps: document preprocessing, pattern recognition, and pattern extraction.

5.1 Document Preprocessor

The document preprocessor takes in input an unstructured document, and a query containing the class instances names representing the information that the user needs to extract. After the execution the document preprocessor returns the logic document representation and a set of reasoning modules constituting the input for the pattern recognizer. The document preprocessing is performed by the three sub-modules described in the following.

Query analyzer. This module takes in input the user query and explores the ontologies in order to identify patterns for the extraction process. The output of the query analyzer are two sets of couples (*class instance name*, *pattern*). The first set (O_s) contains couples in which instances are characterized by patterns represented through regular expressions (simple elements); the second set (O_c) contains couples in which patterns are expressed using the HιLεX pattern representation grammar (complex elements). The set O_s is the input of the document analyzer, whereas, the set O_c is the input of the rewriter module.

Document Analyzer. The input of this module is an unstructured document and the set of couples O_s. The document analyzer applies pattern matching mechanisms to detect simple elements constituting the document and for each of them generates the relative *portion*. At the end of the analysis the (*logic document representation* L_s) is returned as output. More formally, document analysis consists of a transformation expressible through the following function:

$$F : D \times O_s \rightarrow L_s \qquad (1)$$

In this function D is the original unstructured document viewed as a set of rectangular regions; O_s is the set of couples, obtained from the query on the ontologies, containing patterns expressing simple elements and their related class instance names; L_s is the resulting logical document representation containing a set of instances of the `portion` predicate. This function produces an homeomorphism between the document and its logic representation, in fact, for each portion contained in the logic representation is possible to identify univocally the corresponding document region and vice versa.

HιLεX Rewriter. The input of this module is the set of couples O_c. Each pattern represented by the HιLεX two-dimensional grammar is translated in a set of logical rules organized in a DLP^+ reasoning module. The output of the rewriter is a set of Reasoning Module (RM) executable by the DLV^+ system. The translation is based on predefined operators able to manipulate portions. For example, the translation of the HιLεX expression presented in section 3 is reported below.

```
portion (p: P1, q: Q5, elem:
   knowledge_management_system_01) :-
   strictFollow (P1, Q1, knowledge, P2, Q2, blank_char),
   strictFollow (P2, Q2, blank_char, P3, Q3, management),
   strictFollow (P3, Q3, management, P4, Q4, blank_char),
   strictFollow (P4, Q4, blank_char, P5, Q5, system).
```

5.2 Pattern Recognizer

The pattern recognizer is founded on the DLV^+ system. It take in input the logic document representation (L_s) and the set of reasoning modules (RM) containing the translation of HiLεX pattern in term of logic rules and recognize new complex elements. The output of this step is the *augmented logic representation* (L_c) of a unstructured document in which new document regions containing more complex elements (e.g table having a certain structure and containing certain concepts, phrases having a particular mining, etc.) are identified. More formally, pattern recognition consists of a transformation represented in the following function:

$$G : L_s \times O_c \rightarrow L_c \tag{2}$$

In this function L_s is the logical document representation obtained by the document analyzer; O_c is the set of couples containing patterns representing complex elements translated in the reasoning modules; L_c is the augmented logical representation of the document. At the end of the pattern recognition L_c contains more portions than L_s. Added portions are related to complex elements obtained through composition of other elements recognized by the document analyzer or by other reasoning modules. This transformation preserve the homeomorphism.

5.3 Pattern Extractor.

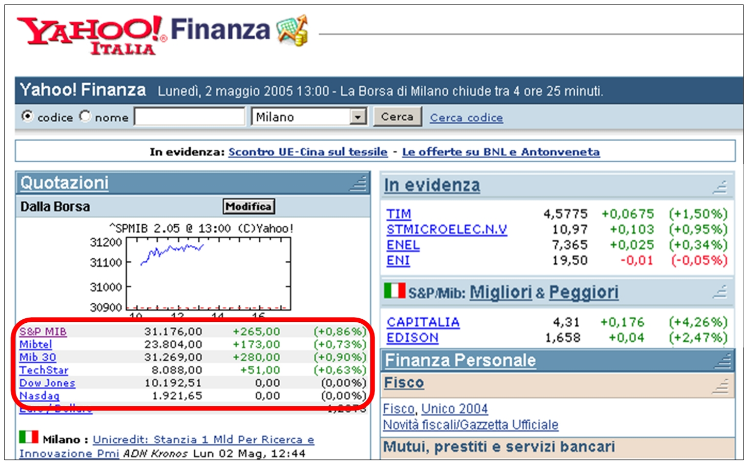
This module takes in input the augmented logic representation of a document (L_c) and allows the acquisition of element instances (semantic wrapping) and/or the document classification w.r.t. the ontologies classes. Acquired instances can be stored in a DLP^+ ontology, a relational database, a XML database. So, extracted information can be used in other applications; moreover, more powerful query and reasoning task are possible on them. For example, the classification of the documents w.r.t. the ontology can be exploited for document management purpose.

6 Application Example

As application example we shows the extraction of information about the stock exchange market contained in the Yahoo Italian financial portal depicted in figure 3. In this page the order of stock indexes change rapidly according to the their values. To acquire the market values whit a certain frequency is important to identify exactly the stock indexes.

The extraction process starts with the representation of knowledge concerning the stock exchange market. We first represent the following domain ontology:

```
class stock_market_domain isa {element}.
    class stock_index isa {stock_market_domain}.
        mibtel: stock_index (type: regexp_type,
        expression: ''Mibtel'').
        mib30: stock_index (type: regexp_type,
        expression: ''Mib30'').
        ...
    class stock_index_row isa {stock_market_domain}.
        stock_index_row_00: stock_index_row (type:
```


Yahoo! Finanza Lunedì, 2 maggio 2005 13:00 - La Borsa di Milano chiude tra 4 ore 25 minuti.

In evidenza: Scontro UE-Cina sul tessile - Le offerte su BNL e Antonveneta

Quotazioni

Dalla Borsa

S&P MIB	31.176,00	+265,00	(+0,86%)
Mibtel	23.804,00	+173,00	(+0,73%)
Mib 30	31.269,00	+280,00	(+0,90%)
TechStar	8.088,00	+51,00	(+0,63%)
Dow Jones	10.192,51	0,00	(0,00%)
Nasdaq	1.921,65	0,00	(0,00%)

Milano : Unicredit: Stanzia 1 Mld Per Ricerca e Innovazione Pmi *ADN Kronos* Lun 02 Mag, 12:44

In evidenza

TIM	4,5775	+0,0675	(+1,50%)
STMICROELEC.N.V	10,97	+0,103	(+0,95%)
ENEL	7,365	+0,025	(+0,34%)
ENI	19,50	-0,01	(-0,05%)

S&P/Mib: Migliori & Peggiori

CAPITALIA	4,31	+0,176	(+4,26%)
EDISON	1,658	+0,04	(+2,47%)

Finanza Personale

Fisco

Fisco, Unico 2004
Novità fiscali/Gazzetta Ufficiale

Mutui, prestiti e servizi bancari

Fig. 3. Financial Yahoo Page

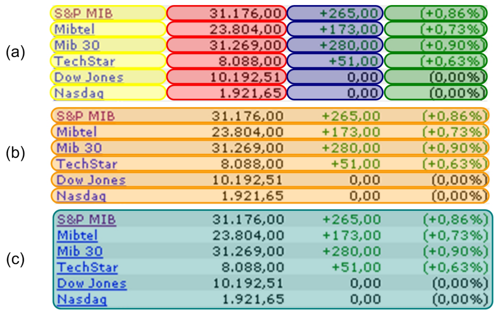
```
          hilex_type, expression: ‘‘sequenceOf (arg:
[@stock_index,
          unsigned_float_number, signed_float_number,
percentage],
          dir: horizontal, sep: sep001)’’).
  class stock_index_table isa {stock_market_domain}.
          stock_index_table_00: stock_index_table (type:
          hilex_type, expression: ‘‘recurrenceOf (arg:
          stock_index_row_00, range: <2, 8>, dir:
vertical)’’).
          ...
```

In the HιLεX extraction patterns expressed above `@stock_index` represent all the instances of the `stock_index` class and `sep001` is defined in the core ontology as:

```
sep001: separator (type: hilex_type, expression:
    ‘‘recurrenceOf (arg: blank_char, range: <0,10>, dir:
    horizontal)’’).
```

The result of the extraction process is graphically shown in figure 4. In particular, figure 4 (a) shows portions identified by the document preprocessor, figure 4 (b) and (c) shows the portion identified by the pattern recognizer.

It is worthwhile noting that: patterns are very synthetic and expressive and that extraction patterns haven’t any reference to the document structure. This last peculiarity implies that the wrapper, defined by the extraction patterns presented above, is more robust w.r.t. variations of the page structure than wrapper obtained from previous approaches. For example, the table containing the stock index could appear wherever in the page. Moreover, the wrapper is general and can be used, also, to extract information from the flat text having the structure depicted in figure 5. The result of the extraction process on flat text is depicted in figure 5 (a), (b), (c).

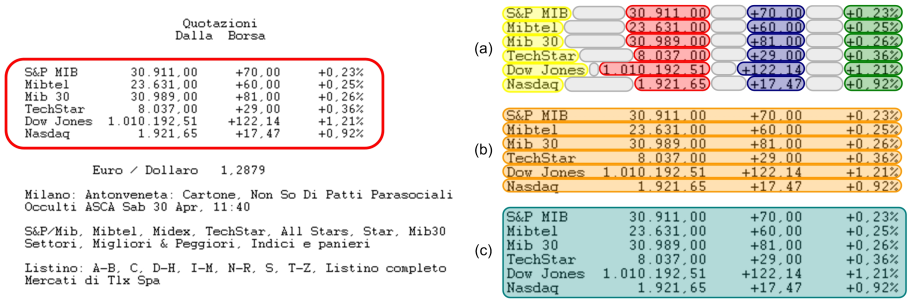


```
(a)
S&P MIB     31.176,00   +265,00   (+0,86%)
Mibtel      23.804,00   +173,00   (+0,73%)
Mib 30      31.269,00   +280,00   (+0,90%)
TechStar     8.088,00    +51,00   (+0,63%)
Dow Jones   10.192,51      0,00    (0,00%)
Nasdaq       1.921,65      0,00    (0,00%)

(b)
S&P MIB     31.176,00   +265,00   (+0,86%)
Mibtel      23.804,00   +173,00   (+0,73%)
Mib 30      31.269,00   +280,00   (+0,90%)
TechStar     8.088,00    +51,00   (+0,63%)
Dow Jones   10.192,51      0,00    (0,00%)
Nasdaq       1.921,65      0,00    (0,00%)

(c)
S&P MIB     31.176,00   +265,00   (+0,86%)
Mibtel      23.804,00   +173,00   (+0,73%)
Mib 30      31.269,00   +280,00   (+0,90%)
TechStar     8.088,00    +51,00   (+0,63%)
Dow Jones   10.192,51      0,00    (0,00%)
Nasdaq       1.921,65      0,00    (0,00%)
```

Fig. 4. Portions Extracted from the Yahoo Page



```
                Quotazioni
              Dalla  Borsa

S&P MIB        30.911,00      +70,00     +0,23%
Mibtel         23.631,00      +60,00     +0,25%
Mib 30         30.989,00      +81,00     +0,26%
TechStar        8.037,00      +29,00     +0,36%
Dow Jones  1.010.192,51     +122,14     +1,21%
Nasdaq          1.921,65      +17,47     +0,92%

         Euro / Dollaro    1,2879

Milano: Antonveneta: Cartone, Non So Di Patti Parasociali
Occulti ASCA Sab 30 Apr, 11:40

S&P/Mib, Mibtel, Midex, TechStar, All Stars, Star, Mib30
Settori, Migliori & Peggiori, Indici e panieri

Listino: A-B, C, D-H, I-M, N-R, S, T-Z, Listino completo
Mercati di Tlx Spa
```

```
(a)
S&P MIB        30.911,00      +70,00     +0,23%
Mibtel         23.631,00      +60,00     +0,25%
Mib 30         30.989,00      +81,00     +0,26%
TechStar        8.037,00      +29,00     +0,36%
Dow Jones  1.010.192,51     +122,14     +1,21%
Nasdaq          1.921,65      +17,47     +0,92%

(b)
S&P MIB        30.911,00      +70,00     +0,23%
Mibtel         23.631,00      +60,00     +0,25%
Mib 30         30.989,00      +81,00     +0,26%
TechStar        8.037,00      +29,00     +0,36%
Dow Jones  1.010.192,51     +122,14     +1,21%
Nasdaq          1.921,65      +17,47     +0,92%

(c)
S&P MIB        30.911,00      +70,00     +0,23%
Mibtel         23.631,00      +60,00     +0,25%
Mib 30         30.989,00      +81,00     +0,26%
TechStar        8.037,00      +29,00     +0,36%
Dow Jones  1.010.192,51     +122,14     +1,21%
Nasdaq          1.921,65      +17,47     +0,92%
```

Fig. 5. Flat Text Version of the Yahoo Page

7 Conclusions and future works

The HιLεX system is a technology based on the novel ontology representation language called DLP^+ having solid theoretical foundation. Many functions that will be available in the future ”semantic web” technologies are turning into reality already today with HιLεX. Currently HιLεX system is under development and their theoretical foundation are under investigation and improvement. HιLεX allows a concrete and powerful semantic approach to information extraction and represents a decisive enhancement in this field. The semantic approach to information extraction presented in this paper can be used to implement a new generation of semantic wrapper able to deal with both HTML and flat text documents.

Future works will be focused on:

– The consolidation and extension of the HιLεX two-dimensional grammar.
– The investigation of computational complexity issues from a theoretical point of view.
– The extension of the approach to pdf and other document formats.
– The exploitation of disjunctive rules in the HιLεX extraction patterns.

References

1. Eikvil L. Information extraction from world wide web: a survey. Technical Report 945, Norwegian Computing Center, Oslo, Norway, July 1999.
2. Alberto H. F. Laender, Berthier A. Ribeiro-Neto, Altigran S. da Silva, and Juliana S. Teixeira. A brief survey of web data extraction tools. ACM Sigmod Record, 31(2), June 2002.
3. Tredwell R. and Kuhlins S. Wrapper generating tools. Published on http://www.wifo.uni-mannheim.de/ kuhlins/wrappertools/ 2002.
4. Gelfond M., Lifschitz V. Classical Negation in Logic Programs and Disjunctive Databases. NGC, vol. 9, pg. 365-385, 1991.
5. Dell'Armi T. and Leone N. Il Linguaggio DLP+. Exeura Internal Report, June 2004.
6. Eiter T., Faber W., Leone N. and Pfeifer G. Declarative Problem-Solving Using the DLV System Logic-Based Artificial Intelligence, Kluwer Academic Publishers, pg. 79-103, 2000.
7. W. Faber and G. Pfeifer. DLV homepage. www.dlvsystem.com, since 1996.
8. Dell'Armi T., Faber W., Ielpa G., Koch C., Leone N., Perri S., Pfeifer G. System Description: DLV. Logic Programming and Nonmonotonic Reasoning - 6th International Conference, LPNMR'01, Vienna, Austria, September 2001.
9. Eiter T., Leone N., Mateis C., Pfeifer G. and Scarcello F. A Deductive System for Non-Monotonic Reasoning Proc. of the 4th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'97), J. Dix, U. Furbach and A. Nerode, editors, 1265, pp.364-375, Springer, LNAI 1997.
10. Baumgartner R., Flesca S., Gottlob G. Visual Web Information Extraction with Lixto. Proceedings of the 27th VLDB Conference, Roma, Italy, 2001.
11. Baumgartner R., Flesca S., Gottlob G. Declarative information extraction, web crawling and recursive wrapping with lixto. In Logic Programming and Nonmonotonic Reasoning, 6th International Conference, LPNMR 2001, Vienna, Austria, Proceedings, pages 21-41. Springer, September 2001.
12. Giammarresi D. , Restivo A. Two-dimensional languages. Handbook of Formal Languages G. Rosenberg, A. Salomaa Eds. Vol. III, pag. 215-268. Springer Verlag, 1997
13. Miller A. and Fellbaum C. WordNet a lexical reference system. http://wordnet.princeton.edu/