# Ontology-enablement of a system for semantic annotation of digital documents

William J BLACK,[1] Simon JOWETT,[1] Thomas MAVROUDAKIS,[2] John McNAUGHT,[1] Babis THEODOULIDIS,[1] Argyrios VASILAKOPOULOS,[1] Gian-Piero ZARRI,[3] and Kalliopi ZERVANOU, [1]
[1]*Department of Computation, UMIST, PO Box 88, Manchester, UK*
[2]*Hellenic Ministry of National Defence, 151 Messogion Av., 15500 Athens, Greece*
[3]*LaLICC, University of Paris IV/Sorbonne, 96 boulevard Raspail – 75006 Paris, France*

**Abstract.** We describe the recent enhancement of the CAFETIERE formalism (Conceptual Annotation of Facts, Events, Terms, Individual Entities and RElations) with the ability to link natural language words and phrases in textual documents with instances and classes from a language-enabled ontology. The language-enabled ontology is one with an index from one or more natural language expressions to each concept (as in WordNet). In an information extraction application. the index, ontology and instance repository are consulted in place of the usual gazetteer prior to the application of the context-sensitive phrase structure rules of the CAFETIERE formalism. Information from the ontology and its instances is cached so that rules can be constrained by properties of objects and can in turn build representations using those properties. We describe the notational extensions to CAFETIERE and give examples of the extraction of event instances in the analysis of texts relative to a specific application ontology. Relevant background is given on the architecture and common annotation scheme of the Parmenides system (FP5 project), in the context of which this work has been done.

## Introduction

The vision of the Semantic Web implies that digital documents are enhanced with conceptual metadata that can support indexing and inference about the contents of the documents, as argued in [1, 2]. In the Parmenides project (IST project IST-2001-39023), we are also concerned with mining pre-analyzed texts to discover patterns of temporal relations between events[3]. Fully-automatic IR-based approaches to document indexing and search appeal because the alternative is to run up against the knowledge acquisition bottleneck, with its attendant need for expensive intellectual effort.

In Parmenides, we adopt the middle way of using automated analysis at a higher level than pure IR indexing, drawn from the body of Information Extraction techniques[4, 5]. These mechanisms, defined and refined in the MUC conferences,[1] involve intermediate-level natural language analysis techniques to identify the extent and referent class of proper names and other expressions in text, and building on that, extract relational and factoid information, filling slots in templates or predicate-argument structures.

Because of the inherent limits in the accuracy of information extraction, the Parmenides architecture prominently features an annotation editing tool which allows missing and spurious analyses to be corrected, while still benefitting from time savings compared with fully human-edited annotation.

[1]See `http://www.itl.nist.gov/iaui/894.02/related_projects/muc/proceedings/ie_task.html`.

The traditional IE system produces textual output, whereas the Parmenides requirement is to have not merely classified text spans, but rather to identify the knowledge-base instances denoted by each extracted phrase, and have the predicates and arguments in template representations identified with ontology classes and instances. In this paper, we show how this can be done both during manual/intellectual annotation and automatic information extraction.

Section 1 introduces the essence of the common annotation scheme which is a DTD defining the format that is used for inter-module communication in all phases.

Section 2 outlines the basic analysis pipeline of the Parmenides system and clarifies the role of the basic components, concentrating on the module responsible for looking up items in the knowledge base, and the module responsible for identifying phrases and structures based on a combination of syntactic analysis and the integration of information from different levels of analysis and sources of background knowledge. This discussion includes the role of the "common annotation scheme" as a lingua franca for structural and conceptual annotations. Section 2.2 explains essentials of the Cafetiere formalism which conducts a rule-based analysis to build annotations of spans and to fill templates. Section 3.3 shows how Cafetiere has been extended for ontology linkage to achieve this goal.

## 1 The Common Annotation Scheme

The Parmenides Common Annotation Scheme (CAS) is an XML representation which consists of three types of annotations as described in [6]

Structural Annotations: These define the structure of the document (head, body and further sections, paragraphs, sentences and tokens). These annotations are in-line annotations i.e. they contain the text spans they label.

Lexical Annotations: These identify lexical units of interest (entity instances), such as person's names, organizations, drug names, time expressions, etc. and are token-reference annotations, i.e. they do not contain textual spans but refer to unique token IDs instead.

Semantic/Conceptual Annotations: These are also token-reference annotations referring to specific (already marked up as lexical annotations) entities via co-referential IDs. They mark entities, relationships and events.

## 2 A sketch of the Parmenides analysis pipeline

The analysis conducted in Parmenides is a pipeline in which each stage of analysis adds to the annotations of its predecessors. This is depicted in Figure 1 where the steps are numbered for convenience. Step 1 involves conversion from external formats to an XML document conformant with the Common Annotation Scheme DTD. Step 2 breaks the text into single word (and equivalent) tokens, and step 3 applies a part of speech tagger [7] to associate the contextually most likely part of speech tag for each token.

Step 4 is a necessary but not sufficient mechanism allowing phrases identified and classified in subsequent stages to be mapped to known classes or instances, i.e. to ground the textual annotations in the ontology. More information on this mechanism follows in Section 2.1.

Step 5 exploits any or all of the prior stages of analysis, together with syntactic rules, to build conceptual annotations representing entities, events and relations. This is discussed further in Section 2.2.

Step 6 allows the user to validate and correct or augment the analyses produced by the automated steps of the system pipeline. This is done using a custom-built annotation editor [8], since such a user may modify annotations but has no right to edit the underlying content.

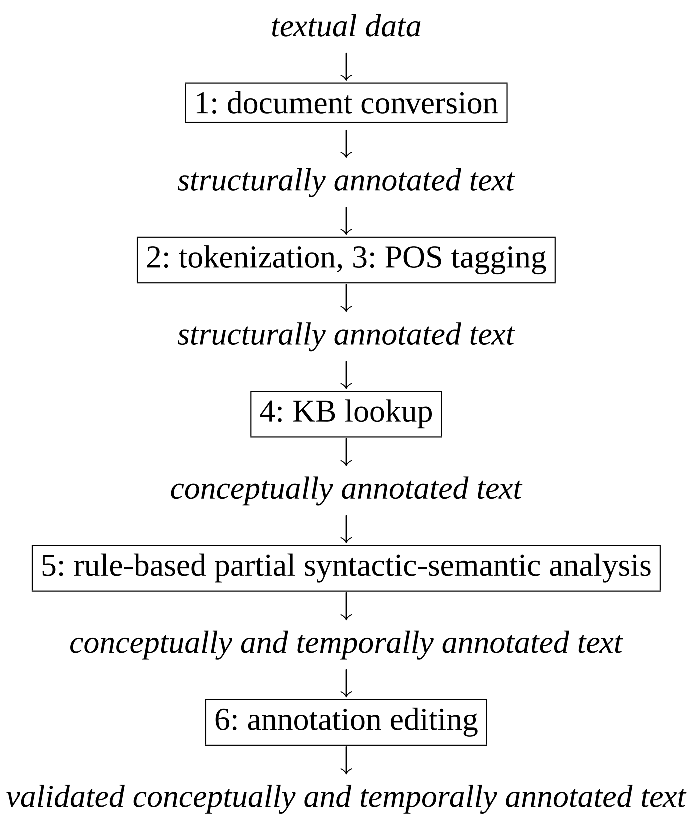
textual data
1: document conversion
structurally annotated text
2: tokenization, 3: POS tagging
structurally annotated text
4: KB lookup
conceptually annotated text
5: rule-based partial syntactic-semantic analysis
conceptually and temporally annotated text
6: annotation editing
validated conceptually and temporally annotated text

Figure 1: Essential steps in the Parmenides analysis pipeline

## *2.1 Lookup*

The lookup module consults an index that maps a word or phrase to a class label or instance identifier. Since the same string (e.g. “*Washington*”) can denote entities of different classes, the lookup annotation is a disjunction of possible *phrase* → *identifier* mappings. Even when singly-valued, the gazetteer entries are not relied on to annotate text spans, but provide additional evidence for the rule-based analysis phase about the concepts represented by text spans.

## *2.2 Rule-based partial syntactic-semantic analysis*

The Parmenides temporal text mining architecture uses the CAFETIERE [9] formalism to identify “basic semantic elements” from texts. CAFETIERE stands for “Conceptual Annotations for Facts, Events, Terms, Individual Entities, and RElations.”

The product of the analysis is a set of conceptual annotations as described in section 1. Unlike a ‘classical’ information extraction (IE) application, the annotations are linked to the classes and instances of an application-oriented ontology.

Since ultimately, the goal is the discovery of trends in the coincidence of event types, the units to be extracted are ultimately occurrences (or *facts*). These occurrences are classified relative to a hierarchy of event classes (the NKRL [10] H-TEMP), which is described further in 3.

In addition to classification, the temporal grounding of the *event* as indicated by verb tense and aspect, and by temporal adverbials are extracted as features of a lexical annotation. The representation of the occurrence needs the arguments (subject, object, etc.) of the verb (or event-denoting noun) to be identified, to complete a template instance, one of the classes of conceptual annotation supported by the system. The arguments themselves are either named *individual entities* or objects denoted by *terms* in the domain under analysis (identified via the ontology of entities – the H-CLASS in NKRL).

Some basic semantic elements identified in rule-based analysis are instances of concepts already in the domain ontology, although others are discovered during analysis.

All events whose instances are to be annotated must be in the ontology, but for other elements, the class can be determined heuristically from contextual clues.

Not all proper names need to be known to the system prior to analysis, because following the state of the MUC art, it is possible to classify names accurately from their textual occurrence and context. Similarly, not all unnamed entities need to be known beforehand. Common noun phrases can be analysed syntactically, or alternatively, annotations can be confined to those for which statistical evidence suggests domain termhood.

Rule-based analysis is used in creating all lexical annotations above the token level, and all conceptual annotations. Items found in the ontology lookup phase must be confirmed by rules, which may specify contextual constraints that will disambiguate when the same string can name or describe different objects.

The rule-based analysis formalism is essentially similar to that reported in [11], but enhanced to give various extensions to its expressive power, and now based on a compiled FST implementation.

Rules have the form *A*=>*B*\ *C*/*D*; *A* describes the text span if the rule succeeds, and *C* represents a sequence of one or more constituent phrases. The rule, being *context-sensitive*, requires elements *B* and *D* to be found to the left and right of *C* in order to label the constituents *C* as the phrase *A*.

Phrases and their constituents are described by a set of attribute-value pairs enclosed in square brackets; both negation and disjunction of values are supported; attributes range over orthographic, morpho-syntactic and semantic/conceptual properties; attributes are used as in HPSG-like linguistic formalisms both to constrain and to construct representations by means of feature unification (through Prolog-like named variables); there is a mechanism to identify longer-distance relationships such as anaphoric co-reference. Examples of rules are (1) and (2).

```
(1)  [syn=NP, sem=ORG, sector=EDU, loc=_LOC] =>
       \ [token="University"],
         [token="of"],
         [sem=LOC, token=_LOC] / ;

(2)  [syn=NNP, sem=PERSON] =>
        [sem=title]{1,2}
      \ [orth=capitalized],
        [orth=upperinitial]?,
        [orth=capitalized] / ;
```

The annotation being constructed is described on the first line of rule (1), by the features `syn`, `sem`, `sector` and `loc`. The first three of these features are ascribed in the rule, but the feature `loc` takes its value from the variable `_LOC`, which *shares* with the other instance which is the value of the `token` feature of the last word in the phrase. (Variables are recognizable to the system by having an initial underscore.) The symbols `\` and `/` mark the boundary between the phrase's constituents and its left and right contexts respectively. In (1) there are no contextual constraints, but in (2) the capitalized words with optional middle initial have to be preceded by a title for the phrase to be considered the name of a person.

### 2.3 Semi-automatic metadata annotation

Annotation is semi-automatic, which means that various levels of NLP processing are applied to the text, but because of the inherent limits to accuracy in such analyses, an editor is able to verify or correct the analysis in an annotation editing tool. As stated earlier, all annotations conform to a common annotation scheme defined in XML.

The annotation editing tool [8] is custom-built for the annotation scheme. We do not use a standard XML editor because the user does not change the underlying text, only the annotations on it.

The three levels of annotation fall in a strict order of precedence: Structural annotations must be present before lexical annotations are added, and the latter must be present before corresponding conceptual annotations may be added.

The user can edit any document that has been through at least the first phase of analysis (Stage 1 in Figure 1).

## 3 Ontology exploitation

In the project, four different applications are being developed, each supported by its own ontology developed by domain experts. Such an ontology needs an explicit mapping of words and phrases to concepts in order to be linked to information extraction rules. If all classes specify a multi-valued string property *synonym*, then it is straightforward to expect the ontology editors to add the synonymous natural language strings for an instance, e.g. “New York”, “NY”, “The Big Apple”. We are, however, interested in matching not just the proper names of known individual objects, but also domain terms, such as “phase III clinical trial” which are represented in natural language by indefinite and definite descriptions and not by names. Similarly, with our focus on events, we want to match natural language verbs and nouns with event or occurrence-denoting concepts. At the least, we need an ontology framework that allows natural language synonyms to be defined for concepts as well as instances, and an index to facilitate lookup via the synonym property.

Rule 3 shows a verb group or event-denoting noun being labelled semantically with the instantiation to the variable `_lp` that has been made by the lookup module (expressed by the condition `lookup=_lp`). The rule also passes on instantiations of the variables `_TNS`, `_POL` and `_ASP`, unpacked by a previous rule from the part of speech tag.

(3)
```
# A generic event
[sem=_lp, oid=_lp, id=_id, type=PEVENT,class=OCCURRENCE, tense=_TNS,
polarity=_POL, aspect=_ASP, rulid=event_gen1] =>
\
[syn=event_noun|event_phrase, lookup=_lp, lookup!=NIL, lookup<=event,
tense=_TNS, polarity=_POL, aspect=_ASP, id=_id]
/
;
```

When Rule 3 is applied, all occurrences mentioned in phrases syntactically analysed as event_noun or event_phrase, and which have synonyms defined in the ontology, will be visible in the annotation editor. The most important features illustrated by this rule are the three conditions `lookup=_lp, lookup!=NIL, lookup<=event`. The first of these has the effect of instantiating the variable `_lp` if the second condition is satisfied, that is, if there is a non-null result for lookup. The expression `lookup<=event` specifies that the lookup property of the phrase has to be the class event, *or any of its subclasses or subclass instances*. This simple extension of the rule language to exploit inheritance replaces many individual rules in the pre-ontology version.

### 3.1 Templates

An occurrence is not simply a text span in the same way that a name can be. The goal of information extraction is to find from a text the slot fillers for a template representation of occurrences of interest. An ontology that represents prototypical events in the same way can assist this process. Given an interest in management change events, a domain expert has defined an appointment as an occurrence with typed slots for the employer, employee and position, in addition to the time of occurrence.

When used manually following rule-based analysis, the Annotation Editor presents a slot representation of the occurrence to the user for completion, retrieving the names and filler types of each slot from the knowledge base. Candidate fillers for each slot, as found either by rules identifying names and other basic expressions, or by previous editing, are presented in drop-down lists. This ensures the integrity of all annotations, with respect to the ontology.

#### 3.1.1 Rule-based slot filling

So far, ontology linkage has not provided the means to fill slots automatically. Modifying rule (3) with the condition `lookup<=person-company-event` in place of `lookup<=event`, and specifying further constituents to be found in its right context, as in (4), allows the employee and role slots to be filled from the objects of the verb phrase or prepositional phrases modifying the event noun.

(4)
```
# Appointment event with person then role as objects
[syn=VP, sem=_lp, oid=_lp, id=_id, employee=_eeid,
c_position=_posid, type=PEVENT,class=OCCURRENCE, tense=_TNS,
polarity=_POL, aspect=_ASP, rulid=event_App1]    =>
\
[sem=event_noun|event_phrase, lookup=_lp, lookup!=NIL,
lookup<=company-person-event, tense=_TNS, polarity=_POL,
aspect=_ASP, id=_id]
/
[token="of"]?,
[sem=person, id=_eeid],
[token="of"|"as"|"to"]?,
[sem=position, id=_posid]
;
```

Similar rules for other event types will find slot fillers automatically, reducing but not eliminating the amount of annotation to be done by hand. However, the constraints on the slot fillers as recorded in the ontology's event templates have to be reproduced when writing each such rule.

The approach is suitable when only a small number of event types are of interest to the application. For application to the broader domain of the Semantic Web, the slot type constraints need to be expressed in the ontology, and not re-expressed in pattern-matching rules.

#### 3.1.2 NKRL: an event-template oriented knowledge representation framework

From the linguistic information extraction point of view, allowing the user complete freedom to name slots is not ideal, so we have considered a more disciplined approach to knowledge base construction and occurrence annotation, that of NKRL(Narrative Knowledge Representation Language)[10]. The most important innovation of NKRL with respect to similar knowledge representation tools (KRL, Conceptual Graphs, etc) consists in the addition of an ontology of events (i.e. a catalogue of standard, formalised representation of characteristic situations and events) to the usual ontology of concepts. Thus, the NKRL tool relies on two ontologies, a hierarchy of concepts (H_CLASS) and a hierarchy of events (templates, H_TEMP).

H_TEMP templates are NKRL predicative structures representing general classes of events they are the models of the predicative occurrences. The predicative occurrences are the NKRL representation of specific events: they instantiate the templates by replacing the variables with specific concepts from the hierarchy H_CLASS. In this way, occurrences describe the semantic contents of documents.

A template has a name, a parent template, a natural language description, a predicate, a set of roles (mandatory, forbidden or optional), a set of mandatory modulators, and a set of forbidden modulators. The predicates are: BEHAVE, EXIST, EXPERIENCE, MOVE, OWN, PRODUCE, and RECEIVE. The roles are SUBJ(ect), OBJ(ect), SOURCE, BEN(e)F(iciary), MODAL, TOPIC and CONTEXT. The SUBJ role is mandatory for every template. There are two classes of predicate arguments (role fillers): simple and complex. A simple argument can be a concept from H_CLASS, or a variable restricted to some values in H_CLASS. A complex argument is built using an AECS operator (ALTERN, ENUM, COORD or SPECIF) and a list of arguments that, again, can be simple or complex and must comply with the ”priority rule”: ALTERN (ENUM (COORD (SPECIF))). The roles SUBJ, OBJ, SOURCE, BENF may have a location associated with them. As an example, a template from a Greek MOD case study H_TEMP is shown below. (5) shows the concept and its hierarchical parent, (6) shows the constraints the event concept has on its arguments, (7) shows a text fragment to which this applies and (8) is a set of related filled templates analysing text fragment (7). In (6), ‘symbolic_label’ - an element of the “standard” ontology of concepts of NKRL, H_CLASS - is there to denote that the (“structured”) information to be transmitted is formed by a set of predicative occurrences, associated within a second order structure called a “binding occurrence”. In (7), ‘symbolic_label’ is then instantiated into ‘mod.c3’, the symbolic name of a specific binding occurrence stating that the content of the message transmitted by the Philippine Army consists of the two simultaneous - COORD(ination) - events represented by ‘mod3.c4’ and ‘mod3.c5’.

(5) Name: Move:StructuredInformation
Parent: Move:TransmitInformation
Description: ’Transmit an item of Structured Information’

(6)
```
MOVE SUBJ var1:[(var2)]
OBJ var3
[SOURCE var4:[(var5)]]
[BENF var6:[(var7)]]
[MODAL var8]
[TOPIC var9]
[CONTEXT var10]
{[ modulators ], ¬ abs}
var1 = <human_being_or_social_body>
var3 = <symbolic_label>
var4 = <human_being_or_social_body>
var6 = <human_being_or_social_body>
var8 = <artefact_>| <information_support>|<service>| <transmission_medium>
var9  <sortal_concept>
var10 = <situation_>| <symbolic_laebl>
var2, var5, var7 = <physical_location>
```

(7) ZAMBOANGA CITY: A son of a wealthy Filipino businessman was abducted by

armed members of the most violent Muslim rebel group in the southern Philippines, the military said yesterday. Robustiano Hablo, 30, was on his way home with his father when the Abu Sayyaf rebels blocked their way in a village south of Manila on Saturday.

(8)
```
mod3.c2) MOVE    SUBJ     PHILIPPINE_ARMY: (ZAMBOANGA_CITY)
                 OBJ      #mod3.c3
                 date-1:  21/11/1999
                 date-2:

Move:StructuredInformation (4.42)

mod3.c3) (COORD   mod3.c4  mod3.c5)

mod3.c4) PRODUCE SUBJ    (SPECIF GROUP_1 armed_): (VILLAGE_1)
                 OBJ     kidnapping_
                 BENF    ROBUSTINIANO_HABLO
                 date-1: 20/11/1999
                 date-2:

Produce:PerformTask/Activity (6.3)

mod3.c5) MOVE    SUBJ     (COORD1 ROBUSTINIANO_HABLO INDIVIDUAL_20): ()
                 OBJ      (COORD1 ROBUSTINIANO_HABLO INDIVIDUAL_20):(home_)
                 date-1:  20/11/1999
                 date-2:

Move:PersonDisplacement (4.31)
```

NKRL's limited set of role names in place of predicate-specific roles such as employer, employee and position is a positive benefit, from the point of view of making template filling rules sufficiently generic, but the choice to restrict predicates to a narrow set of primitives is not so compelling.

### 3.2 *PS-NKRL*

An implementation of a simplified variant of NKRL has been made by Wordmap, who provide commercial taxonomy management systems.[2] for use in Parmenides applcations. This variant allows for the import of the two NKRL hierarchies, but does not constrain class definitions to observe the restrictions either on predicate names or role names.

PS-NKRL has three aspects. The first is to define a constrained version of NKRL suitable for the needs of the analysis module. The second is to make the PS-NKRL ontologies available through a suitable navigation API and allow the manipulation of these through the WORDMAP Ontology Manager.

### 3.3 *Ontology extensions to the* CAFETIÈRE *rule formalism*

With a knowledge base in place of a gazetteer, the lookup stage of analysis can do more than before: As with the gazetteer, it supplies semantic classes (concepts) corresponding to words and phrases. It returns object identifiers and slot values for known instances, including where aliases and abbreviations name the same object. It retrieves the slots to be filled for anonymous instances of a class, including the types of slots of an event.

The rule formalism is extended with additional operators as follows:

[2]See `http://www.wordmap.com`

- The comparison operator `<=` which exploits inheritance at lookup-time, as explained above.
- The dot (.) operator between slot names, which allows access to the value of a slot of an object which is the filler of a slot in the current constituent.
  For example `capital.population=_pop` would instantiate `_pop` with the appropriate value, say 10000000, if the current constituent is an instance looked up from the string “United Kingdom”, which has as the value of a slot named country, an entity for whom the population slot has the value 10000000.

### *3.3.1 Obtaining event constraints from the ontology*

In (9), we see a general syntactic rule matching a simple subject-verb-object sequence that builds the semantic representation needed without the template-specific slot names that were used in (4).

(9)
```
[syn=_syn, sem=_event, subj=_s, eid=_event, obj=_o] =>
     [syn=np, lookup<=_sc, eid=_s]
  \ [syn=_syn, lookup<=event, lookup=_event, subjectclass=_sc,
      objectclass=_oc] /
     [lookup<=_oc, eid=_o]  ;
```

This rule will match an appropriate verbal constituent and fill its slots if the looked-up class has the slots subjectclass and objectclass and their respective values match the lookup values for the preceding and following constituents.

General syntactic rules like this can recognize and fill the slots for a wide range of event types, provided the slot constraints are expressed in these general terms and not by roles particular to the event type. However, such a policy is not suited to the outlook of application owners, who are not linguistically oriented, and who will be unable to map conceptual slots to abstract syntactic roles unaided.

## 4 Conclusions and Future Work

We have described a technical mechanism by which a rule-based information extraction system can be linked to an ontology and instance repository. This is necessary to produce semantic annotation of digital documents with the aid of natural language processing components. The mechanism is also supported by an ontology-enabled annotation editor. The ontology resource is an implementation of NKRL, embedded in an ontology management tool by Wordmap, although we are able to support other knowledge base formalisms, such as Protégé.

Further work is needed on enabling the needs of natural language ontology lookup to co-exist with that of the application owners to name slots as they see fit, and to attain generality of analysis without the writing of excessive domain-specific rules.

## Acknowledgments

The Parmenides project is co-funded by the European Commission (contract No. IST-2001-39023) and the project partners, and by the Swiss Federal Office for Education and Science (BBW/ OFES). Please see http:/www.crim.co.umist.ac.uk/parmenides for a detailed description of the project. The Parmenides consortium consists of the following partners (with responsible persons): Biovista (GR) Andreas Persidis; Ministry of Defence (GR) Thomas

Mavroudakis, Spiros Taraviras; Neurosoft (GR) Giorgos Orphanos; Otto-von-Guericke Universität Magdeburg (D) Myra Spiliopoulou; Coordinator: UMIST (UK) Babis Theodoulidis, William Black; Unilever (NL) Hilbert Bruins Slot, Chris van der Touw; University of Geneva (CH) Margaret King; University of Zurich (CH) Fabio Rinaldi; Wordmap (UK) Will Lowe.

## References

[1] Giam-Piero Zarri. Semantic Web and Knowledge Representation. In A. Min Tjoa and R.R. Wagner, editor, *Database and Expert Systems: Proceedings of 13th International Conference, DEXA'02*, Los Alamitos, CA, 2002. IEEE Computer Society Press.

[2] Fabio Rinaldi, Kaarel Kaljurand, James Dowdall, and Michael Hess. Breaking the Deadlock. In *Proceedings of the International Conference on Ontologies, Databases and Applications of SEmantics (ODBASE'03)*, Catania, Sicily, Italy, 2003.

[3] Myra Spiliopoulou, Fabio Rinaldi, William J. Black, Gian Piero Zarri, Roland M. Mueller, Marko Brunzel, Babis Theodoulidis, Giorgos Orphanos, Michael Hess, James Dowdall, John McNaught, Maghi King, Andreas Persidis, and Luc Bernard. Coupling Information Extraction and Data Mining for Ontology Learning in PARMENIDES. In *Proceedings of RIAO 2004*, Avignon, 2004.

[4] J. Hobbs. The Generic Information Extraction System. In *Proceedings of the Fifth Message Understanding Conference (MUC-5)*, pages 87–91, Baltimore, Maryland, 1993. Morgan Kaufmann, San Francisco, California.

[5] D. Appelt and J. Hobbs and J. Bear and D. Israel and M. Kameyama and A. Kehler and D. Martin and K. Myers and M. Tyson. SRI International FASTUS System: MUC-6 Test Results and Analysis. In *Proceedings of the Sixth Message Understanding Conference (MUC-5)*, pages 237–248, Columbia, Maryland, 1995. Morgan Kaufmann, San Francisco, California.

[6] Fabio Rinaldi, James Dowdall, Michael Hess, Jeremy Ellman, Gian Piero Zarri, Andreas Persidis, Luc Bernard, and Haralampos Karanikas. Multilayer Annotations in PARMENIDES. In *K-CAP2003 workshop on Knowledge Markup and Semantic Annotation*, page (to appear), Sanibel, Florida, USA, 2003.

[7] Argyris Vasilakopoulos. Improved Unknown Word Guessing by Decision Tree Induction for POS Tagging with TBL. In S. Clark and M. Osborne, editors, *6th Annual CLUK Research Colloquium*, Edinburgh, 2003.

[8] Argyris Vasilakopoulos, Michele Bersani, and William J. Black. A Suite of Tools for Marking Up Textual Data for Temporal Text Mining Scenarios. In *LREC 2004*, Lisbon, 2004. *to appear*.

[9] William J. Black and John M$^c$Naught and Argyris Vasilakopoulos and Kalliopi Zervanou and Babis Theodoulidis and Fabio Rinaldi. CAFETIERE: Conceptual Annotations for Facts, Events, Terms, Individual Entities, and RElations. Technical Report TR-U4.3.1, Department of Computation, UMIST, Manchester, 2003. `http://www.co.umist.ac.uk/w̃jb/parmenides/tr-u4.3.1.pdf`.

[10] G. P. Zarri. NKRL, a knowledge representation tool for encoding the meaning of complex narrative texts. *Natural Language Engineering*, 2/3(3):231–253, 1997.

[11] W J Black, L Gilardoni, F Rinaldi, and R Dressel. Integrated text categorisation and information extraction using pattern matching and linguistic processing. In *Proceedings of RIAO97*, pages 321–335, Montreal, 1997.

[12] J.R. Hobbs, M.E. Stickel, D.E. Appelt, and P. Martin. Interpretation as abduction. *Artificial Intelligence*, 63:69–142, 1993.