Knowledge Extraction by using an Ontology-based Annotation Tool

Maria Vargas-Vera, Enrico Motta, John Domingue,
Simon Buckingham Shum and Mattia Lanzoni
Knowledge Media Institute (KMi),
The Open University,
Walton Hall, Milton Keynes, MK7 6AA, United Kingdom
{m.vargas-vera, e.motta, j.b.domingue, s.buckingham.shum, m.lanzoni}@open.ac.uk

ABSTRACT

This paper describes a Semantic Annotation Tool for extraction of knowledge structures from web pages through the use of simple user-defined knowledge extraction patterns. The semantic annotation tool contains: an ontology-based mark-up component which allows the user to browse and to mark-up relevant pieces of information; a learning component (Crystal from the University of Massachusetts at Amherst) which learns rules from examples and an information extraction component which extracts the objects and relation between these objects. Our final aim is to provide support for ontology population by using the information extraction component. Our system uses as domain of study "KMi Planet", a Web-based news server that helps to communicate relevant information between members in our institute.

Keywords

Ontology-based mark-up, Ontology population, Extraction of knowledge, Information extraction technologies.

1. INTRODUCTION

Semantic annotation has been focused in isolated annotations of web pages. However, semantic web tries to achieve the annotation of pages with semantic information. In other words, the aim is to enrich the content of web pages. Recent work on semantic annotation guided by an ontology is discussed in [14]. However, our approach has a different aim, we use the ontology as guider to the human annotator of the training set (ie. the user is presented with a set of possibles tags which could be used during the mark-up process), and then the system learns rules by using the semantic annotations, whilst in OntoAnnotate [14] the user selects the object identifier and the appropriate class for it from a hierarchy of classes. Then all the information which is in the Ontology for that particular object identifier is presented to the user. If the object identifier is not defined the user could create a new object or class relation.

One target of the system presented in this paper is to learn rules from texts by using a machine learning component called Crystal. To extract rules from text, we had developed an environment which allows user to perform four phases: browse, semantic annotation of pages, learning rules and information extraction (IE) of the web pages. Each of these phases are described as follows:

1. **Browse**

   This option could by used by the user to select the kind of browser in our case could be WebOnto or any other browser. WebOnto [3] provides web-based visualisation, browsing and editing support for the ontology. It allows easier development and maintenance of the knowledge models, themselves specified in OCML (Conceptual Modeling Language) [8].

2. **The Markup phase**. The activity of semantic tagging refers to the activity of annotating text documents (written in plain ASCII or HTML format) with an tags set defined on the ontology, in particular we work with the hand-crafted KMi ontology (ontology describing Knowledge Media Institute). The semantic annotation tool provides means to browse the event hierarchy (described in next section). In this hierarchy each event is a class and the annotation component extracts the set of possible tags from the slots defined in each class. In general mark-up process might be difficult but in our case the annotation component is guiding the user with the possible entities which could be marked in the text.

   Other approach related to our work is the SHOE Knowledge annotator which is a Java program that allows users to mark-up web pages with the SHOE ontology [5]. However, in SHOE there is not relation between the new annotations and the original text.

3. **Learning phase**. This phase uses the marked text as training set and learns relations from the stories. It uses crystal as a learning component. Crystal works using the bottom-up approach. It finds rules for specific instances and it generalises these rules.

4. **The information extraction phase**. The goal of a Information Extraction system (IE) is to extract specific types of information from text. For example, an IE system in the domain of KMi (Knowledge Media Institute) organisation, should be able to extract the name of KMi projects, KMi funding organisations, awards, dates, etc. The main advantage of IE task is that portions of a text that are not relevant to the domain can be ignored. Therefore text could be processed quickly.

Most IE systems use some form of partial parsing to recognise syntactic constructs without generating a complete parse tree for each sentence. Such partial parsing has the advantages of greater speed and robustness. High speed is necessary to apply the IE to a large set of documents.

IE has been used in several domains, for instance, scientific articles such as MEDLINE (it contains abstracts of biomedical journals) [2], bibliographic notices [9], and medical records [13]. Also, ontologies has been used in IE systems to help them extract relations from semi or unstructured documents, statements or terms [11]. Recent work on semi-automatic ontology acquisition by means of IE, supported by machine-learning methods, is described in [6, 4]. In similar lines there is the CMU's approach for extracting information from hypertext using machine learning techniques (Bayes classifier) and making use of an ontology [1]. However, we remark that we are not creating an ontology, we are supporting ontology population. The ontology population problem is an important issue to be addressed since it is difficult to keep up to date a hand-crafted ontology.

In our work, we had integrated the hand-crafted KMi Ontology into the information extractor. The main task of the ontology is to disambiguation of some extracted information. For instance, in the event conferring an award "X was granted Y amount of money". X could be instantiated to name of project or institution. In this case we make use of the ontology to clarify the type of X.

In the construction of our IE component we had integrated several components (Marmot, Badger and Crystal) from the University of Massachusetts at Amherst (UMass) which are fully described in Rillof[10]. We remark that in our IE component the template matching itself is supported semantically by referring to the ontology, but also contains some lightweight NLP techniques in order to syntactically identify some *fragments* of the sentences. We believe it is important to mix the syntactic and semantic. The semantic checking is often necessary to resolve ambiguities, for example, ontologies can provide us with axioms of common sense knowledge such "if someone is visiting a place then this someone should be a person." Conversely, some grammar constructions (such as dates) can be recognized robustly.

Figure 1 illustrates the four phases. In particular la browse phase has been launched.

Our primary contribution is to integrate a template-driven IE engine with an ontology engine (including inference capabilities besides lexicons such as Wordnet) in order to supply the necessary semantic content and then to disambiguate extracted information and finally our second contribution is to provide support for the ontology population process.

The paper is organised as follows: In Section 2 we present a typology of two events as are defined in KMi ontology. Section 3 presents the mark-up phase. Section 4 shows the learning phase using Crystal. Section 5 presents the extraction of information using Badger. Section 6 describes the use of ontology to cope with the ambiguity in the identification of objects in the story. Section 7 shows the OCML [1] code generated after badger obtains template instantiations. Section 8 discusses the process of populating an ontology as an activity in the life cycle of the ontology construction. Finally, Section 9 gives conclusions and directions for future work.

[1] OCML is a language designed for knowledge modeling

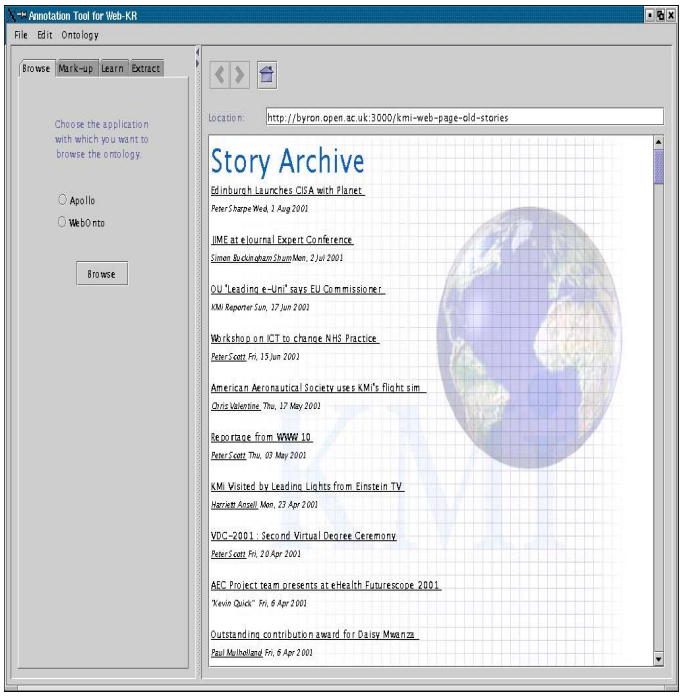

Figure 1: System overview

2. EVENT TYPOLOGY

KMi ontology consists of KMi projects, people in KMi, events, etc. In particular we will focus in a section of the KMi ontology called events (activities happening in our Institute). The events are defined formally in our ontology as classes. Currently, in our KMi ontology we have defined 40 different types of events. As the event typology is already defined in the KMi ontology. Then, for each event we already had defined the slots which might be instantiated by the IE component. Figure 2 shows a portion of the hierarchy of events as defined in KMi ontology.

For the sake of space, we only present the structure of three type of events from the event hierarchy: visiting-a-place-or-people, conferring-a-monetary-award and demonstration-of-technology.

```
Class Event 1: visiting-a-place-or-people

slots:

visitor (list of person(s))
people-or-organisation-being-visited
    (list of person(s) or organization)
has-duration (duration)
start-time (time-point)
end-time (time-point)
has-location (a place)
other agents-involved (list of person(s))
main-agent (list of person(s))
```

The structure of Event 1 (visiting-a-place-or-people) describes a set of objects which might be encountered in story describing an event visit, such as, visitor, people-or-organisation-being-visited, other agents-involved, etc.

```
Class Event 2: conferring-a-monetary-award
```

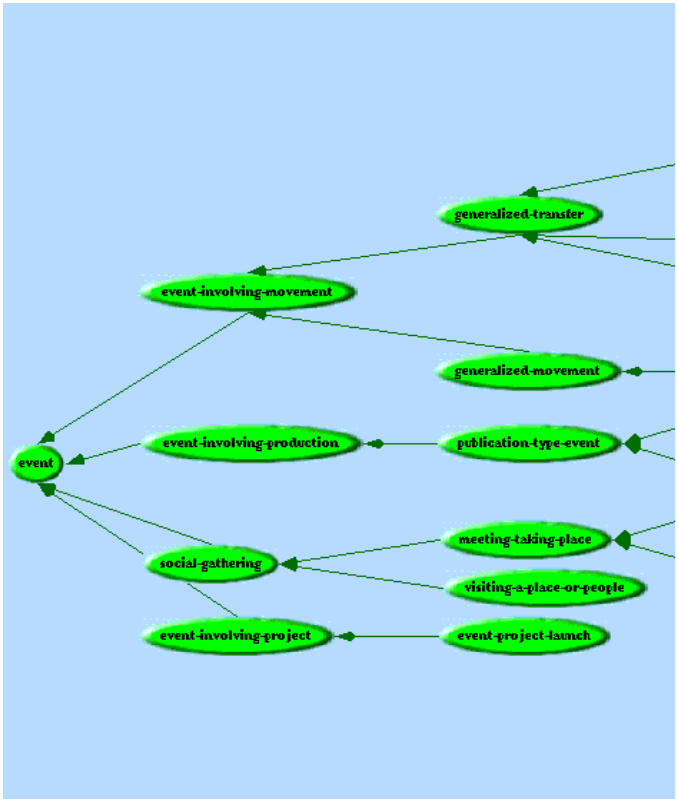

Figure 2: Event hierarchy

```
 slots:

monetary award (sum of money)
has-duration (duration)
start-time (time-point)
end-time (time-point)
has-location (a place)
main-agent (list of person(s))
other agents-involved (list of person(s))
location-at-start (a place)
location-at-end (a place)
awarding-body (an organization)
has-award-rationale (project goals)
```

In the event 2 the value for the slot **has-award-rationale** is extracted from text by using heuristics such as if the word **goal** appears in the story then the system will extract as rationale the sentence until it finds full stop. The reason for this is because is to general to be learned by an IE component. It does not follow any grammar rule about how the rationale could be expressed by a journalist who writes an story describing a project's award.

```
Class Event 3: demonstration-of-technology

technology-being-demostrated (technology)
has-duration (duration)
start-time (time-point)
end-time (time-point)
has-location (a place)
other agents-involved (list of person(s))
main-agent (list of person(s))
location-at-start (a place)
location-at-end (a place)
medium-used (equipment)
subject-of-the-demo (title)
```

Event 3 contains the structure for the event "demonstration-of-technology". Entities that need to be recognised are technology, place, etc.

3. MARK-UP PHASE

The mark-up component aims to help the manual annotation of web pages. In this component the ontology plays a important role guiding the mark-up process. The user does not know which is the relevant information which might be annotated. Therefore, we consider that is useful to have a such tool that presents user with possibles tags. An example of annotated story is shown in Figure 4. The user selects an specific class on the hierarchy of events, for example, "visiting-a-place-or-people". Then a set of possibles tags is presented to the user for the event "visiting-a-place-or-people". The set of tags are: has-duration, start-time, end-time, has-location, other agents-involved, main-agent, visitor, people-or-organisation-being-visited. From this set the user could select a subset of tags and then automatically a template for the event "visiting-a-place-or-people" is created. The created template is used later by the component which make instantiations of templates (Badger). Figure 3 shows the user selection. In this particular example the user only selects start-time, end-time, has-location and visitor.

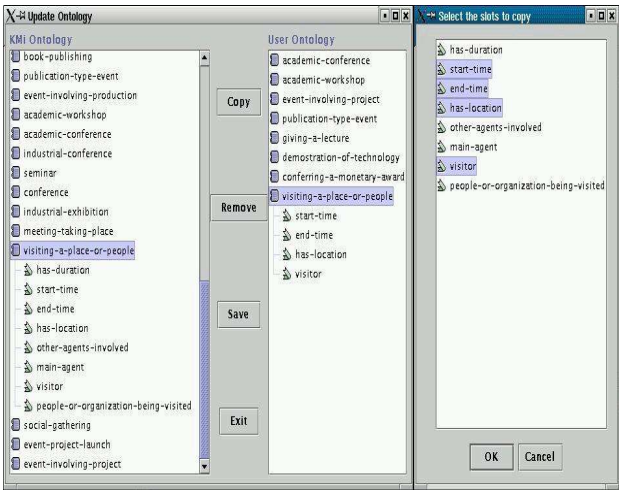

Figure 3: Selection of tags

For the sake of space, let us assume that the user annotates the story with two tags: visitor and place from the selected set. Figure 4 shows the semantic annotations which automatically are inserted in the text. In the story **David Brown** was annotated as visitor and **The OU** is annotated as place.

4. LEARNING PHASE

This phase was implemented by integrating two tools Marmot and the learning component called Crystal both from Umass.

A brief description of Marmot (a text preprocessor) is giving before the learning component Crystal is presented.

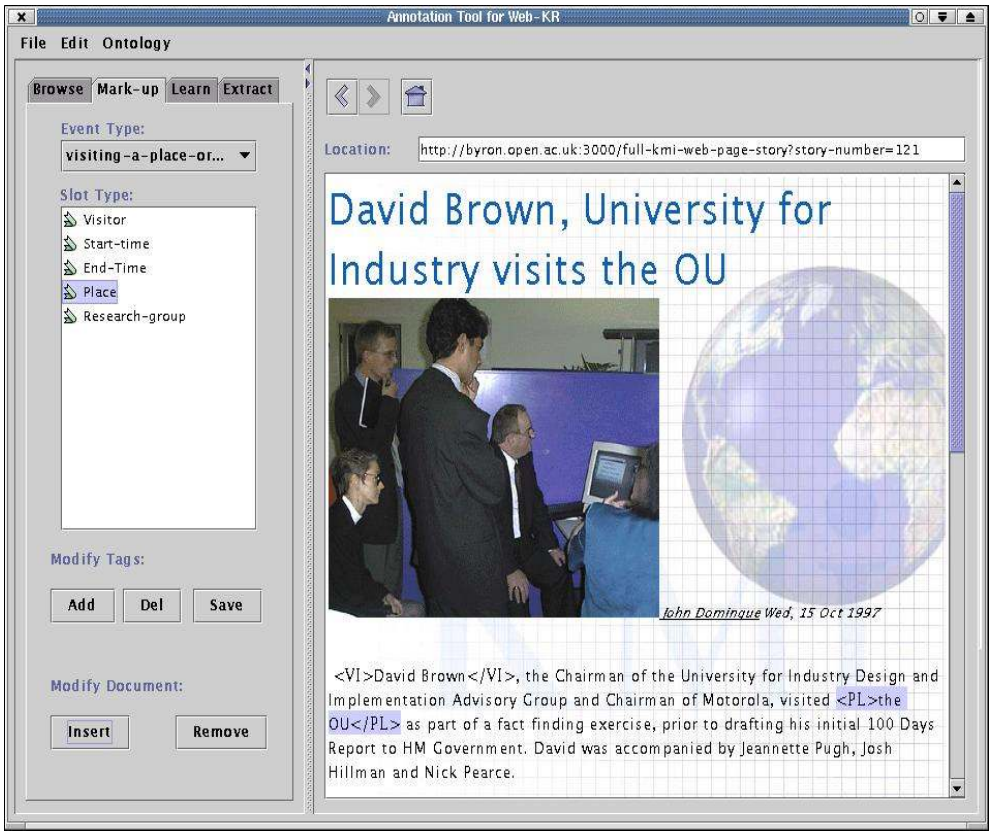

Figure 4: Annotated story

4.1 Marmot

Marmot (from UMass) is a natural language preprocessing tool that accepts ASCII files and produces an intermediate level of text analysis that is useful for IE applications. Sentences are separated and segmented into noun phrases, verb phrases prepositional phrases.

Marmot has several functionalities: preprocesses abbreviations to guide sentence segmentation, resolves sentences boundaries, identifies parenthetical expressions, recognises entries from a phrasal lexicon and replace them, recognises dates and duration phrases, performs phrasal bracketing of noun, preposition and adverbial phrases, finally scopes conjunctions and disjunctions.

We had defined our own verbs, nouns, abbreviations and tags in order to apply Marmot to our KMi domain. For the sake of space we would analyse only the first three sentences in the story given in Figure 5.

In the first sentence, Marmot recognised two entities firstly a subject (SUBJ) which is JOHN DOMINGUE and secondly a date. The latest is recognised and marked between the symbol "@". Dates are recognised robustly as regular expressions.

```
SUBJ(1): JOHN DOMINGUE
ADVP(2): @WED_%COMMA%_15_OCT_1997@
PUNC(3): %PERIOD%
```

In sentence number 2, DAVID BROWN is recognised as subject (SUBJ), a prepositional phrase (PP) "FOR INDUSTRY" is encounter, the verb (VB) VISITS is also found, OBJ1 takes the value of THE OU and finally a punctuation symbol (PUNC) is the full stop is encountered at the end of the sentence.

```
SUBJ(1): DAVID BROWN %COMMA% UNIVERSITY
PP  (2): FOR INDUSTRY
VB  (3): VISITS
OBJ1(4): THE OU
PUNC(5): %PERIOD%
```

In the same fashion, in sentence number 3, DAVID BROWN is recognised as subject, the word VISITED is recognised as verb and OBJ1 as THE OU.

```
SUBJ(1): DAVID BROWN %COMMA% THE CHAIRMAN OF
THE UNIVERSITY
PP  (2): FOR INDUSTRY DESIGN AND IMPLEMENTATION
ADVISORY GROUP AND CHAIRMAN OF MOTOROLA
PUNC(3): %COMMA%
VB  (4): VISITED
OBJ1(5): THE OU
```

4.2 Crystal

Crystal is a dictionary induction tool. It derives a dictionary of concept node (CN) from a training corpus. The first step in dictionary creation is the annotation of a set of training texts by a domain expert. Each phrase that contains information to be extracted is tagged (with SGML style tags).

Crystal initialises a CN dictionary for each positive instance of each type of event. The initial CN definitions are designed to extract the relevant phrases in the training instance that creates them

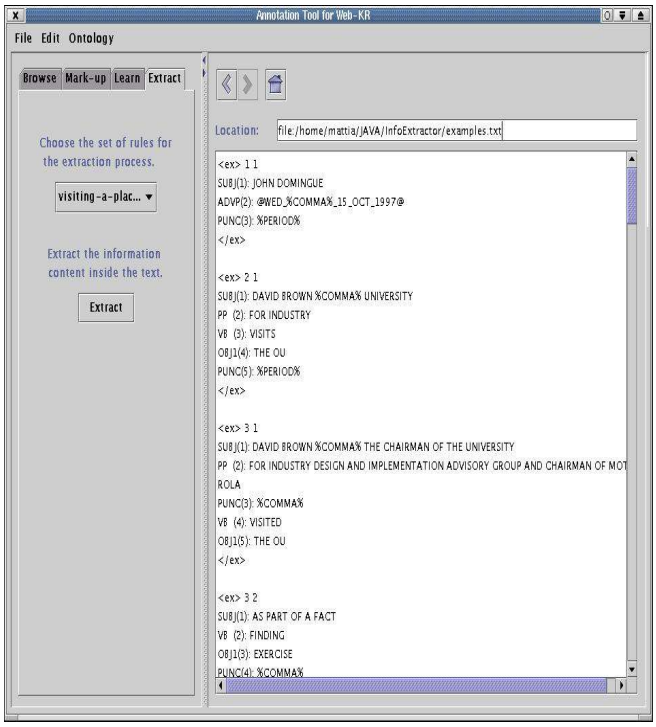

Figure 5: Marmot output

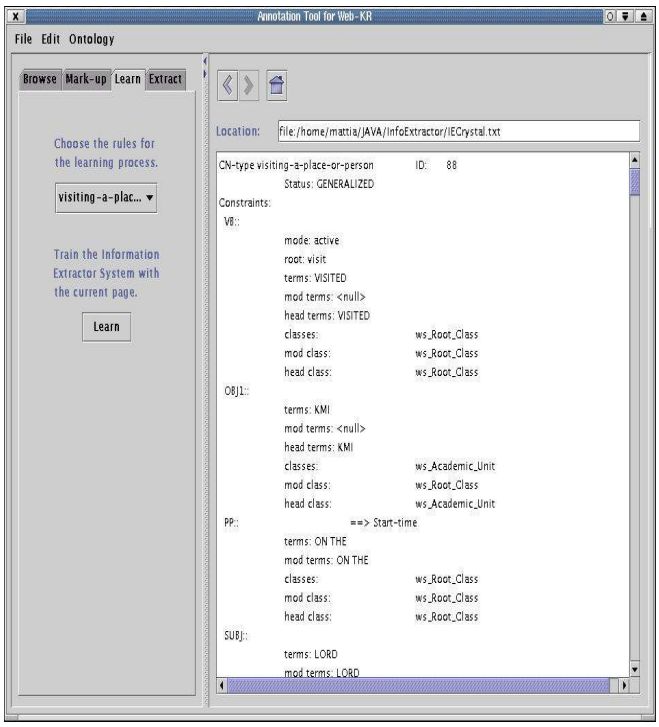

Figure 6: Crystal output

but are too specific to apply to a unseen sentences. The main task of Crystal is to gradually relax the constraints on the initial definitions and also to merge similar definitions.

Crystal finds generalisations of its initial CN definitions by comparing definitions that are similar. This similarity is deduced by counting the number of relaxations required to unify two CN definitions. Then a new definition is created with constraints relaxed. Finally the new definition is tested against the training corpus to insure that it does not extract phrases that were not marked with the original two definitions. This means that Crystal takes similar instances and generalises into a more general rule by preserving the properties from each of the CN definitions which are generalised.

The inductive concept learning in Crystal is similar to the inductive learning algorithm described in [7] a specific-to-general data-driven search to find the most specific generalisation that covers all positive instances. Crystal finds the most specific generalisation that covers all positive instances but uses a greedy unification of similar instances rather than breadth-first search.

Coming back to our example David Brown's story. We have that Crystal learns a conceptual node such as the one shown in Figure 7.

These conceptual node states that "X visited". So that in the future whenever the pattern "X visited " appears in the text the case frame will extract "X" as the visitor.

For the pattern X visited Y, we basically are extracting relations r(X,Y) from texts which could be interpreted as "X visited Y" and the Lexicon for relation r is the union of the lexicon(X) and lexicon(Y). If we find this relation in our texts then we find a instance for the event "visiting-a-place-or-people".

In this example we do not have the case that two different templates might apply to the same sentence. But it is possible to encounter these cases. Let us consider the following example from the MUC domain (the MUC domain is a set of documents describing terrorist activities in Latin America):

"A visitor from Colombia was hurt when two terrorists attempted to kill the major".

if **visitor from Colombia** is marked as victim **two terrorist** are marked as perpetrators and **major** as victim.

Crystal generates 3 frame cases that represents the following patterns:

If a text contains the expression "X was hurt" then the system extracts "X" as the victim.

If a text contains the expression "X attempted to kill" then the system extracts "X" as perpetrator.

If the text contains the expression "attempted to kill Y" then the system extracts "Y" as the victim.

In recent years had been great interest in annotated-based techniques for producing automatically dictionaries. The reason for this is that automatic creation of conceptual dictionaries is important factor for portability and scalability of an IE system.

Crystal has been tested on corpus of 300 KMi stories. Crystal was able to induce a dictionary of CN definitions for each event in KMi ontology.

5. EXTRACTION PHASE

A third component called Badger (from UMass) which was also integrated into our IE component.

Badger makes the instantiation of templates. The main task of badger is to take each sentence in the text (in our case a story written in a e-mail message) and see if it matches any of our CN definitions. If no extraction CN definition applies to a sentence, then no information will be extracted; this means that irrelevant text can be processed very quickly.

It might occurs that Badger obtains more than one type of event for an story. Then our IE system decides to classify the story ac-

Verb: visited (active verb)

Visitor: V (class_person)

Has−location: P (class_place)

Start−time: ST (class time_point)

End−time: ET (class time_point)

Has−duration: D (class duration)

Figure 7: Concept node for the visiting event

cording with the following criteria: how many feature for each type were encountered in the story.

Badger obtained a case frame instantiations for Place and Visitor using conceptual nodes defined in the dictionary constructed by Crystal. In the Badger's output the following conventions were used: the name of the slot appears in the left hand side of the arrow and the value for the slot on the right hand side of the arrow. In David Brown story, Badger instantiated Place to The OU and visitor to David Brown. The type of event is obtained from the value of Type and the document ID from docid.

The output shown in Figure 8 means that Badger had instantiated (using the CN definitions and domain lexicon) to a frame of the form:

```
Concept Node:
        CN-type: visiting-a-place-or-people
        Slots:
        Visitor   tag: VI
        Start-time  tag: ST
        End-Time    tag: ET
        Place     tag: PL
        Research-group tag: GR
```

Date is not stated in the story. So Start-time and End-time are instantiated to the date in which the story was written.

6. INFERENCE CAPABILITIES BY USING AN ONTOLOGY

An example of an story belonging to the type of event conferring-a-monetary-award is defined as follows. This example is described in this paper because shows the inference capabilities which could be obtained from using an IE component plus an ontology.

> IBROW has been awarded 1 million Ecu from the European Commission to carry out research in the area of knowledge-based systems.

The output from Badger is shown as below.

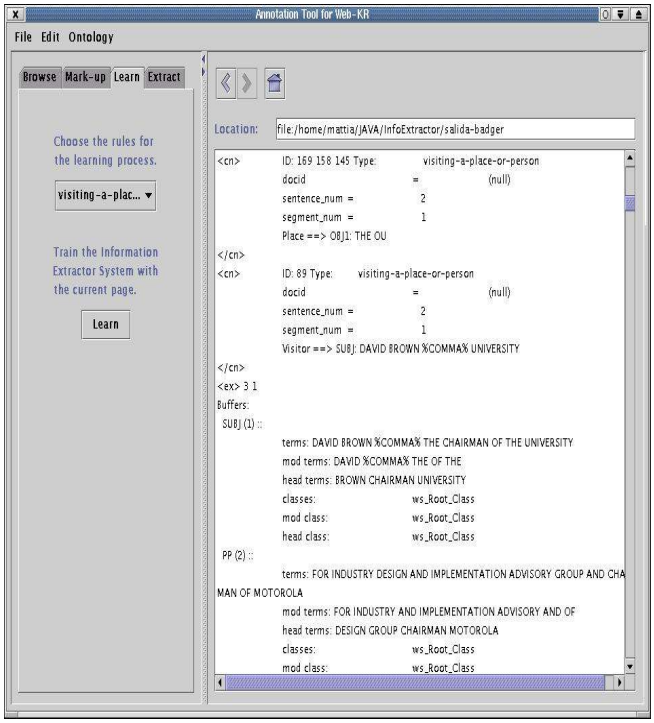

Figure 8: Badger output

```
<cn>ID: 80 Type: conferring-a-monetary-award
docid =  ibrow-story
sentence_num =    1
segment_num =    1
Funder ==> PP: FROM THE EUROPEAN COMMISSION
</cn>

<cn>ID: 106 Type:  conferring-a-monetary-award
docid =  ibrow-story
sentence_num =    1
segment_num =    1
Money ==> OBJ1: 1 MILLION ECU
</cn>

<cn>ID: 24 Type: conferring-a-monetary-award
docid =  ibrow-story
sentence_num =    1
segment_num =    1
Project-Institution ==> SUBJ: IBROW
</cn>
```

In this last example, we need to use the KMi planet ontology to find if Project-Institution is a **institution name** or **a project name**, and this is done by a simple traversal of the inheritance links in the ontology. Specifically, to remove ambiguity we sent a query to Web-onto asking for the set of all educational-organizations using the following query code.

```
web-onto display akt-kmi-planet-kb
ocml-eval(setofall ?x
        (educational-organization ?x))
```

This gives a list containing all educational-organizations:

```
 to give @(the-open-university
         ...
         org-knowledge-media-institute)
```

IBROW does not match any of these, however, we also send a query to Web-onto asking for the set of all kmi-projects:

```
web-onto display akt-kmi-planet-kb
  ocml-eval(setofall ?x
            (kmi-project ?x))
```

yielding

```
to give @(project-d3e
          ...
          project-kmi-planet
          ...
          project-ibrow
          ...
          project-heronsgate-mars-buggy)
```

and hence a match of "IBROW" to project-ibrow

In a similar fashion a query is sent to webonto in order to find if Funder is a valid funder body.

```
web-onto display akt-kmi-planet-kb
   ocml-eval(setofall ?x
             (awarding-body ?x))

    to give @( ...
               org-european-commission
               org-british-council)
```

At same time some **semantic relations** could be obtained by using the KMi planet ontology. For our example about IBROW we can derive the following semantic relations:

"ibrow is KMi project" and "KMi is part-of the Open-University"

The OCML query to derive that KMi is part of the open university is as follows:

```
web-onto display akt-kmi-planet-kb

   ocml-eval(setofall ?x
             (organization-unit-part-of ?x
               the-open-university))


 to give @(knowledge-media-institute
   acad-unit-department-of-earth-science
   acad-unit-department-of-statistics-ou
   acad-unit-faculty-of-maths-and-computing-ou
   ...
   org-office-for-technology-development)
```

therefore we could conclude that:

"the Open-University has been awarded 1 million Ecu from the European Commission"

In a future implementation we will be interested in finding more complex relations by using our KMi Planet ontology.

Finally, we remark that OCML (the query language used by webonto) has adopted the closed world assumption (CWA), in the same fashion as Prolog, and so facts that are not provable are regarded as "false" as opposed to "unknown".

7. OCML CODE GENERATED FROM OUR SYSTEM

Our goal is to use the information obtained by Badger and KMi ontology in order to be able to populate our KMi ontology with new instances of classes. In order to accomplish this task we had plugged another component which is a translator from Badger's output to OCML code. The main function of this translator is to tokenise the Badger output and then find the CN definitions (cn markers) and extract all the objects encountered in the story. The name of each slot in the frame case corresponds to the name of the field in the class definition and the value for the field is the extracted information.

For the example David Brown's story we end up with a visiting-a-place-or-people event and produce the intermediate output:

```
(def-instance  visit-of-david-brown-
     the-chairman-of-the-university
     visiting-a-place-people
  ((has-duration 1-day)
   (start-time  wed-15-oct-1997)
   (end-time  wed-15-oct-1997)
   (has-location  the-ou )
   (visitor  david-brown-the-
     chairman-of-the-university)
  )
)
```

where an instance of the type event visiting-a-place-or-people has been defined with the name "visit-of-david-brown-the-chairman-of-the-university".

8. POPULATING THE ONTOLOGY

Building domain-specific ontologies often requires time-consuming expensive manual construction. Therefore we envisage IE as a technology that might help us during ontology maintenance process. During the population step our IE system has to fill predefined slots associated with each event, as already defined the ontology. Our goal is to automatically fill as many slots as possible. However, some of the slots will probably still require manual intervention. There are several reasons for this problem:

- there is information that is not stated in the story,
- none of our templates match with the sentence that might provide the information (incomplete set of templates)

We note that there are some cases when the instances are not defined in the ontology and then determining the type of an object is not straightforward. This has to be derived from a proof. Currently, we still looking to this aspect of our research.

Figure 9 shows the extracted information from David Brown story.

Once the system had extracted the information the user will presented with all extracted information even the one that cannot be categorized as belonging to a type of object defined in our domain. Therefore, before populating the ontology we will require that a person check/complete the extracted information.

9. CONCLUSIONS AND FUTURE DIRECTIONS

We had built a tool which extracts knowledge using an ontology, an IE component and OCML translator. Currently, our system had

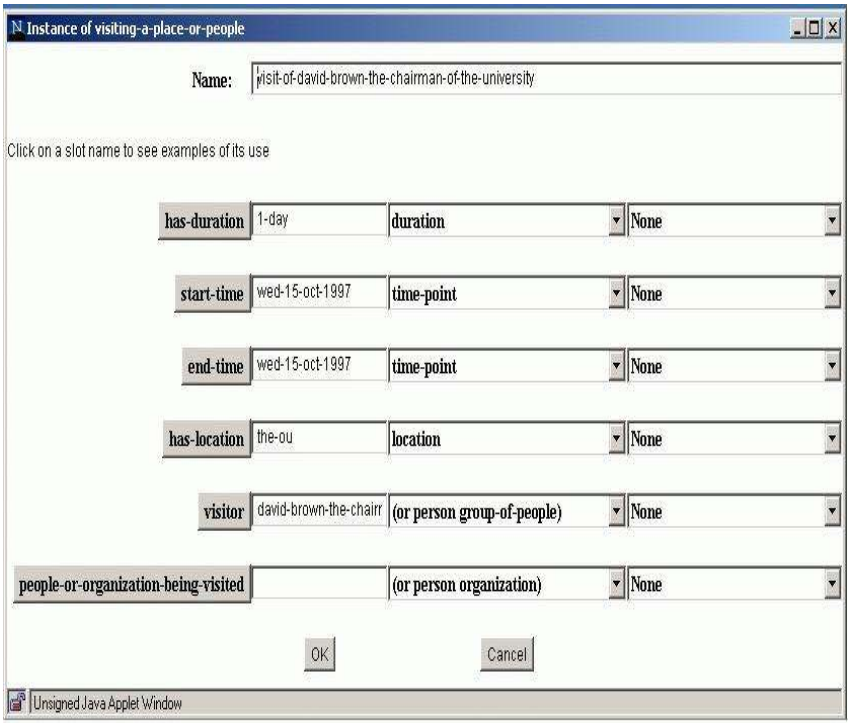

Figure 9: Extracted information

been trained using the archive of 300 stories that we had collected in KMi. [2] The training step was performed using typical examples of stories belonging to each of the different type events defined in the ontology. We obtained results over 95% using the IE component in KMi stories. However, in the future we would like to use the IE component in a different domain. We are interested in using our system in companies project reports, Curriculum Vitae (CV's), or application of jobs.

Another possible direction that we would like to explore is to incorporate into the IE component a different Machine Learning algorithm such as described in [12]. in order to compare performance between them.

As medium term goal, we would like to have access to a library of IE methods and to activate these over a web page or a collection of web pages.

Besides the above issues, Badger could be extended in order to save its output in XML (Extensible Markup Language). This will increase the portability of our IE system as XML is the universal format for structured documents and data on the Web.

Finally, we would like to integrate our IE component with visualisation component. This visualisation component will allow visualisation of all entities extracted.

10. ACKNOWLEDGMENTS

The research described in this paper is supported by (EPSRC) under the project name: Advanced Knowledge Technologies (AKT).

11. REFERENCES

[1] M. Craven, D. DiPasquo, D. Freitag, A. McCallum, K. Nigam T. Mitchell, and S. Slattery. Learning to Construct Knowledge Bases from the World Wide Web. *Artificial Intelligence*, 1999.

[2] M. Craven and J. Kumlien. Constructing Biological Knowledge Bases by Extracting Information from Text Sources. In *Proceedings of The 7th International Conference on Intelligent Systems for Molecular Biology (ISMB-99)*, 1999.

[3] J.B. Domingue. Tadzebao and WebOnto:Discussing, Browsing and Editing Ontologies on the Web. In *Proceedings of the Knowledge Acquisition Workshop*, 1998.

[4] J-U. Kietz, A. Maedche, and R. Volz. A method for semi-automatic ontology acquisition from a corporate intranet. In *Proceedings of the EKAW'00 Workshop on Ontologies and Text, Juan-Les-Pins, France*, oct 2000.

[5] S. Luke, L. Spector, D. Rager, and J. Hendler. Ontology-based Web agents. In *Proceedings of the First International Conference on Autonomous Agents*, pages 59–66. ACM, 1997.

[6] A. Maedche and S. Staab. Semi-automatic engineering of ontologies from texts. In *Proceedings of the 12th International Conference on Software Engineering and Knowledge Engineering, SEKE2000, Chicago, IL, USA*, pages 231–239, jul 2000.

[7] T. Mitchell. Generalization as search. *Artificial Intelligence*, 18:203–226, 1982.

[8] E. Motta. *Reusable Components for Knowledge Modelling*. IOS Press, Netherlands, 1999.

[9] D. Proux and Y. Chenevoy. Natural Language Processing for Book Storage: Automatic Extraction of Information from Bibliographic Notices. In *Proceedings of The Natural Language Processing Pacific Rim Symposium (NLPRS'97)*, pages 229–234, 1997.

[10] E. Riloff. An Empirical Study of Automated Dictionary Construction for Information Extraction in Three Domains. *AI Journal*, 85:101–134, 1996.

[11] C. Roux, D. Proux, F. Rechenmann, and L. Julliard. An Ontology Enrichment Method for a Pragmatic Information Extraction System gathering Data on Genetic Interactions. In *Proceedings of The 14th European Conference on Artificial Intelligence (Workshop on Ontology Learning ECAI-2000)*, 2000.

[12] S. Soderland. Learning Information Extraction Rules for Semi-structured and free Text. *Machine Learning*, 34:1–44, 1999.

[13] S. Soderland, D. Aronow, D. Fisher, J. Aseltine, and W. Lehnert. Machine Learning of Text Analysis Rules for Clinical Records. Tr 39, Center for Intelligent Information Retrieval, 1995.

[14] S. Staab, A. Maedche, and S. Handschuh. An annotation framework for the semantic web. In *Proceedings of the First Workshop on Multimedia Annotation*, 2001.

[2] URL:http://kmi.open.ac.uk/planet/