Knowledge Media Institute

Template-Driven Information Extraction for Populating Ontologies

M. Vargas-Vera, J. Domingue, Y. Kalfoglou,
E. Motta and S. Buckingham Shum

KMI-TR-105

March, 2001

www.kmi.open.ac.uk/tr/papers/kmi-tr-105.pdf

IJCAI'01 Workshop on Ontology Learning (OL- 2001)
August 4, 2001, Seattle,USA.

Template-Driven Information Extraction for Populating Ontologies

M. Vargas-Vera, J. Domingue, Y. Kalfoglou,
E. Motta and S. Buckingham-Shum
Knowledge Media Institute (KMi),
The Open University,
Walton Hall, Milton Keynes, MK7 6AA, United Kingdom
{m.vargas-vera, j.b.domingue, y.kalfoglou, e.motta, s.buckingham-shum}@open.ac.uk

Abstract

We address the integration of information extraction (IE) and ontologies. In particular, using an ontology to aid the IE process, and using the IE results to help populate the ontology. We perform IE by means of domain specific templates and the lightweight use of Natural Languages Processing techniques (NLP). Our main goal is to learn information from text by the use of templates and in this way to alleviate the main bottleneck in creating knowledge-base systems that is "the extraction of knowledge". Our domain of study is "KMi Planet", a Web-based news server that helps to communicate relevant information between members in our institute [Domingue and Scott, 1999]. The raw input consists of e-mailed stories written by members of the laboratory. The main goals of our system are to classify the story, obtain the relevant objects within the story, deduce the relationships between them, and to populate the ontology. Furthermore, we aim to do this with minimal help from the user.

1 Introduction

A longstanding goal of the knowledge acquisition community is to be able to provide automatic support for the extraction of relevant information from unstructured text. One method, of course, is to simply aim for the full understanding of the text, however such in-depth understanding is still out of reach. Instead, "information extraction" (IE) systems have the more modest aim of focusing only on portions of the text that are relevant to a specific domain or task, and ignoring the rest. Essentially, IE can be seen as the task of pulling predefined relations from texts. Efforts have been made to apply IE to several domains, for instance, scientific articles, bibliographic notices [Proux and Chenevoy, 1997], and medical records [Soderland et al., 1995]. In designing an IE system for our KMi (Knowledge Media Institute) organisation, the system should be able to extract the name of KMi projects, KMi funding organisations, awards, dates, etc, and ignore anything not clearly relevant to these pre-specified categories. Ontologies can be used in IE systems to help them extract relations from semi or unstructured documents, statements or terms [Roux et al., 2000]. Also, recent work on semi-automatic ontology acquisition by means of IE, supported by machine-learning methods, is described in [Maedche and Staab, 2000; Kietz et al., 2000].

Most IE systems use some form of partial parsing to recognize 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. The robustness achieved by allowing useful work to be done from a partial parsing, is essential to deal with unstructured and informal texts (such as the e-mail messages we consider).

There are several approaches to IE. The most obvious is Natural Language Parsing: attempting to write a grammar that allows parsing of the desired texts. This approach is appealing but is fragile for general use. No current system can fully understand the range of language encountered. Another problem with the NLP approach is that it is very language and grammar dependent, requires a detailed parser, and is unlikely to be able to cope with slang or "bad" grammar.

At the other end of the spectrum is the Cyc approach [Lenat et al., 1986] which attempts to supplement the grammatical parsing with common-sense understanding. The Cyc Knowledge Server is an integration of a knowledge base of common sense with several components such as inference engine, representation language, NLP subsystem, etc. More generally, Cyc is intended to provide a "deep" layer of understanding that can be used by other programs to make them more flexible [1].

However, the pure syntactic approach, and deep semantic approaches have so far been too fragile for general use. An intermediate approach is the template-based approach of Riloff [Riloff, 1993]. She classifies text using extraction patterns and semantic features associated to slots in a predefined frames. For example, in the terrorist domain that she considers, the event "murder" is represented by the following frame:

[1] URL:http://www.cyc.com/

```
name-frame:
 event-type murder (active verb murdered)
slots
 victim:
        <subject> (human)
 perpetrator:
        <prepositional-phrase, by> (human)
 instrument:
        <prepositional-phrase, with > (weapon)
```

Each of the templates are triggered by the main verb, in this case "murder", in any tense. Trigger words can be reliably identified using linguistic rules like the ones described in [Riloff, 1996b]. For example, if the targeted information is the subject or the direct object of a verb then the best trigger word should be the main verb. The sentence is also matched using the extra words "by" and "with."

For our IE we use a template-driven approach similar to, and based upon, that of Riloff. We identify the sentences that contain keywords (target-string) encoded in our templates. For our specific event type we have a different set of keywords. By using this keywords we select only relevant sentences from the whole text. then we parse this sentences, and finally recognize the types of objects by syntactic features or by means of the KMi Planet ontology (defined in section 2).

In particular, the main aim of this paper is to describe the use of template-driven IE to populate an ontology. 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. Overall, our primary contribution is to integrate a template-driven IE engine with an ontology engine to supply the necessary semantic content.

The paper is organised as follows: Section 2 briefly describes our suite of tools in order to give some background. In Section 3 we present a solution to the problem of populating the ontology. Section 4 describes the classification process embedded in our system and the use of ontology to cope with the ambiguity in the identification of objects in the story, along with examples based on some of the event types we use. Finally, Section 5 gives conclusions and directions for future work.

2 The KMi Planet System and PlanetOnto

KMi Planet is a Web-based news server that facilities communication between members of KMi. To quote from [Domingue and Motta, 2000]:

> The authors integrated a suite of tools, called **PlanetOnto** that supports a speedy but high-quality publishing process, allows ontology-driven document formalization and augments standard browsing and search facilities with deductive knowledge retrieval.

Two primary components are the story library and the ontology library. The Story database contains the text of the stories that have been provided to Planet by the journalists. In the case of KMi Planet it contains stories which are relevant to our institute. The Ontology Library contains several existing ontologies, in particular the KMi ontology. OCML is the modeling language which allows the creation of classes and instances in the ontology.

The main architecture of PlanetOnto is shown in Figure 1. In PlanetOnto we identify three types of users: journalists who send stories to KMi planet, knowledge engineers who maintain the Planet ontology, and readers of the Planet stories.

PlanetOnto augmented the basic publish/find scenario supported by KMi planet, and supports the following activities [2].

1. **Story submission**. A journalist submits a story to KMi planet using e-mail text. Then the story is formatted and stored.
2. **Story Reading**. A Planet reader browses through the latest stories using a standard Web browser,
3. **Story annotation**. Either a journalist or a knowledge engineer manually annotates the story using Knote (the Planet knowledge editor),
4. **Provision of customised alerts**. An agent called Newsboy builds user profiles from patterns of access to PlanetOnto and then uses these profiles to alert readers about relevant new stories.
5. **Ontology editing**. A tool called **WebOnto** [Domingue, 1998] provides Web-based visualisation, browsing and editor support for the ontology. It allows easier development and maintenance of the knowledge models, themselves specified in OCML (Conceptual Modeling Language) [Motta, 1999].
6. **Story soliciting**. An agent called Newshound, periodically solicits stories from the journalists.
7. **Story retrieval and query answering**. The Lois interface supports integrated access to the story archive

Recently, two tools have been integrated in the architecture: **myPlanet** and an **IE tool**.

- **myPlanet** is an extension to Newsboy and helps story readers to read only the stories that are of interest instead of reading all stories in the archive. It uses a manually predefined set of cue-phrases for each of 'research areas' defined in the ontology. For

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

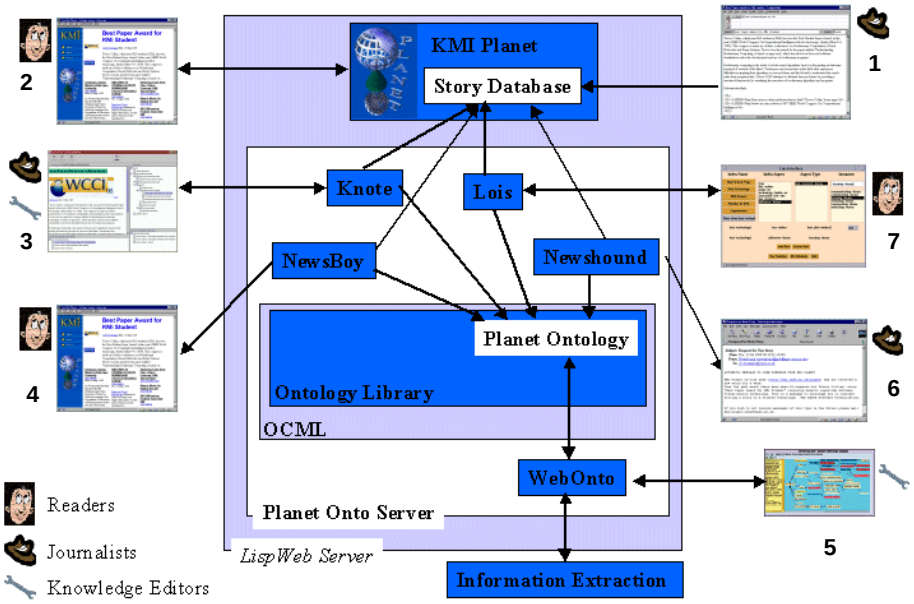

Figure 1: PlanetOnto architecture

example for genetic algorithms one cue-phrase is "evolutionary algorithms". Consider the following example: if someone is interested in *research* area *Genetic Algorithms*. A search engine will return all the stories that talk about that *research area*. myPlanet (by using the ontological relations) will also find all Projects that have research area Genetic Algorithms and then search for stories that talk about these projects, thus returning them to the reader even if the story text itself does not contain the phrase "genetic algorithms".

- **Information Extraction** is a tool which extracts information from e-mail text and it connects with webonto to prove theorems using the KMi-planet ontology.

3 Our Template-Based IE Methods

Our goal is to use templates and the ontology in order to be able to do enough IE to classify stories for use by myPlanet. That is, the overall process is to receive an story written by journalist in a e-mail message and store the story in the server and then the IE tool should

1. classify the story according an event type predefined in our KMi planet ontology
2. find the objects in the story and relations between these objects
3. produce OCML code [3] [Motta, 1999],
4. populate the ontology once the user has corrected/approved the extracted information.

Our domain (KMi planet) consists of email message, and hence is generally unstructured for example: the journalists use slang, break conventions for capitalization, incorrectly use punctuation, etc. Also, styles of writing in such an archive can vary wildly. All these make it difficult to apply heuristics to identify proper nouns. NLP approaches typically use heuristics; if a word is capitalised and not starting a sentence then it is a proper name. If a string contains "& Co" or "Ltd" then it can be tagged as a proper noun of type company. However, as discussed earlier we expect such methods to be too fragile for our domain, and so instead use templates.

The main process in the classification 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 domain-specific templates. If no extraction template applies to a sentence, then no information will be extracted; this means that irrelevant text can be processed very quickly.

Let us take the following example: Suppose that a journalist from KMi sent the story (presented in Figure 2) to the e-mail server

Our IE tool will take the story and produce the information shown in screen snapshot of Figure 3 In this snapshot the extracted information is presented to the knowledge engineer for confirmation. For each type of event a different form is presented from the information stored in the ontology. Figure 3 also has some fields that cannot be found in the story such as people-or-

[3] OCML is a language designed for knowledge modeling

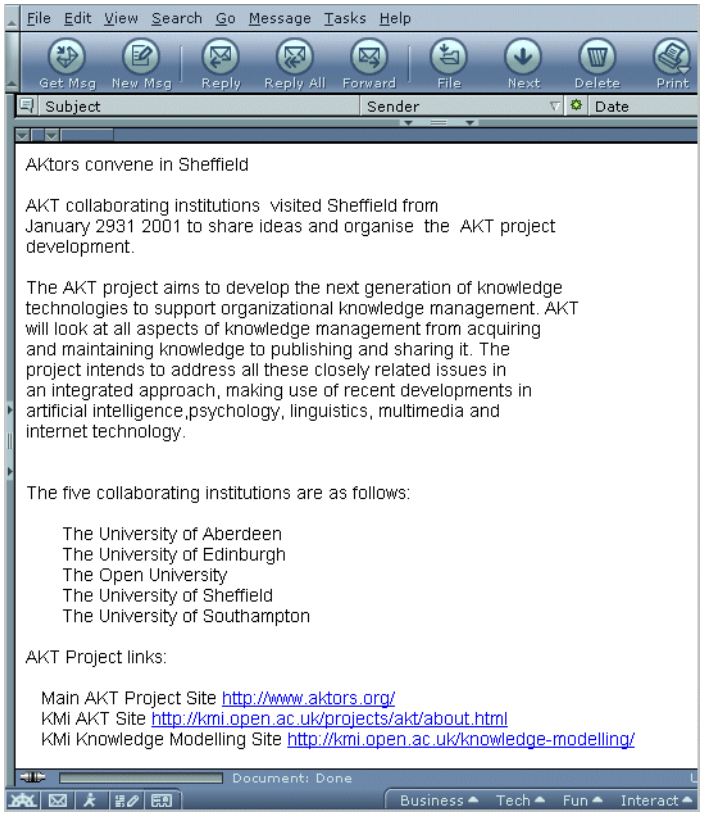

Figure 2: Email Story

organization-being-visited. Therefore, in the right hand side of the form there is a menu that presents to the user with possibles instantiations for each field. The suggested instantiations are obtained from the ontology.

4 Examples of Classification and object identification

We classify the story or document as belonging to any of the 40 types of events according with the type of the objects that are found. For each event type we have a predefined objects that should be found in the story. For the sake of space we do not show the type hierarchy.

Classification is performed in the following steps:

- pre-process the email-text: tokenise the text and join expressions such as dates into a single token
- find the objects in the story using further partial parsing
- identify each of the objects in the story. In our domain this means classify them as person, place, date, etc. This identification is done based partially on syntactic features, e.g. we assume titles such as "Dr." can be used to recognize some people. However, some other classifications are based upon semantic knowledge obtained by querying the server "Web-onto" for the KMi planet ontology. For example, project names will be recognized this way.

We have devised several type of events in KMi planet ontology. For example, "visiting-a-person-or-place", "academic-conference", "academic-workshop", "event-involving-project", etc. Each event has several templates used to recognize it.

The nature of the domain is very important in determining which kind of templates are necessary in our set of templates. Currently, in order to apply our system to a new domain the entire knowledge process of defining templates has to be repeated.

For the templates we use a prolog-like notation: the sentence is converted to a list of words (or tokens) and then matched against a template list containing some keywords, some variables which are allowed to match one or more words, and some underscores "_" that can match zero or more words. The underscores allow us to ignore unneeded parts of the sentence. There are also various constraints imposed on the variables, in order to give semantically sensible matches.

As an example, for the visiting-a-place-or-person type event we can have the following two templates:

1. [_ ,X, _ ,"visited", Y,_] matches the sentence word list. We demand that X is recognizable as an entity

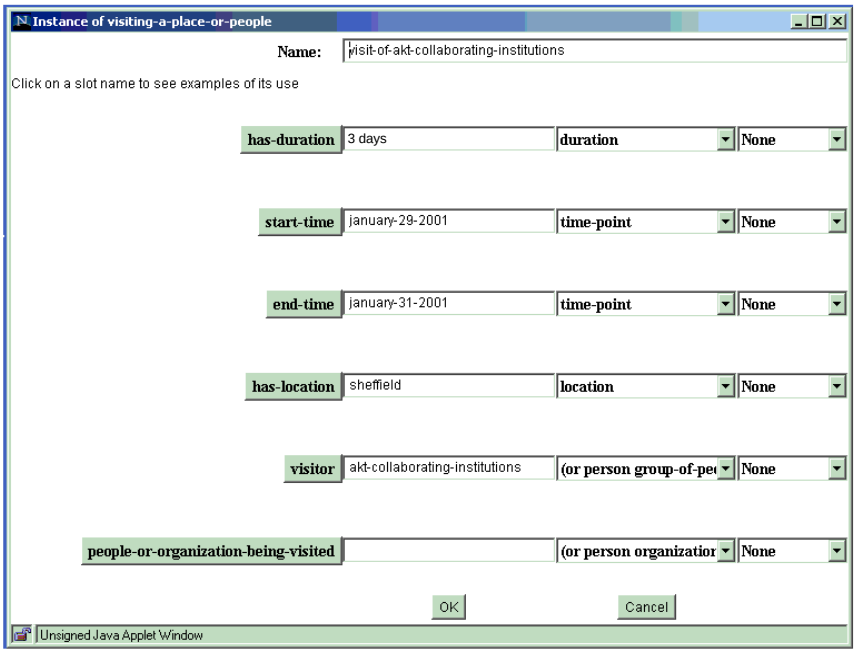

Figure 3: Extracted information

that is capable of visiting,"visitor(X)", and Y as a place, "place(Y)". Hence, for example, we require that Y does not contain a preposition.

2. [_ ,X, _ , "visited", Y, "from", Z , _] matches the sentence word list. Again, we demand visitor(X) & place (Y), but also require "date(Z)"; dates are automatically recognized by syntactic features of month, day, and year.

The templates and matching process are implemented in prolog, and the above "formalisation" is just an approximation of what the code does.

Consider the first sentence of the story in Figure 2. It matches Template 2 as follows

> AKT collaborating institutions
> visited Sheffield from January 29-31 2001 to share ideas and organise the AKT project development.

Note the absence of underscores around the "from" of template 2; this is to restrict spurious matchings.

Hence, the trigger 'visited' creates a visit frame with the following information:

visitor	'akt_collaborating_institutions'
place:	'sheffield'
date:	'january_29_31_2001'
duration-of-visit:	'3 days'

The duration of the visit is computed from the starting and ending date of the visit.

The robustness of the method shows up in that rearranging the sentence does not affect the outcome, e.g.

> Too share ideas the AKT collaborating institutions visited Sheffield from January 29-31 2001.

would also be a match against the template, and extract the same information. In contrast, the misspelled initial "Too" might cause a complete parser to fail entirely and extract no information.

Let us take another example of a candidate sentence for matching to the visited templates above::

> Lord Dalkeith, a Millennium Commissioner, visited KMi from September 2-3,2000.

We again match template 2 with the following instantiations

> X = 'Lord Dalkeith',
> Y = 'KMi'
> Z = 'September 2-3,2000'

The point of this example is to show that the phrase "a Millennium Commissioner" did not affect the matching to the template, although our program has no understanding of the meaning of this phrase.

Finally, a template for the conferring-a-monetary-award type event looks like

> [X, _ ,"has been awarded", Y, "from", Z, _]
> matches the sentence word list. We demand (institution(X) or project(X)) but also require money (Y) & funder-organisation(Z)

An example of an story belonging to the type of event conferring-a-monetary-award is defined as follows:

> IBROW has been awarded 22,500 Ecu from the European Commission to carry out research in the area of knowledge-based systems.

The output is shown as below.

```
Story classified as event_type:
       conferring_a_monetary_award

   project:  [ibrow]
   amount : [22500_ecu]
   funder : [european_comission]
```

In this last example, we need to use the KMi planet ontology to find if X 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 Z 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 (example 3) 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 22,500 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".

4.1 OCML code generated from our system

Once the objects have been identified, and the story classified then the system produces corresponding OCML. For the example story from Figure 2 we end up with a visiting-place-or-person event and produce the intermediate output:

```
(def-instance
    visit-of-akt-collaborating-institutions
    visiting-a-place-or-people
   ((has-duration '3 days')
    (start-time  january-29-2001)
    (end-time  january-31-2001)
    (has-location  sheffield)
    (visitor  akt-collaborating-institutions)
   ))
```

where an instance of the type event visiting-a-person-or-place has been defined with the name "visit-of-akt-collaborating-institutions."

In the above instance definition, each slot-value pair corresponds to one field-value pair as defined in the structure of the class (inside the ontology). Note that we are not restricted to have only one value in each slot. One slot can have several values. For example, has-location could have university-of-sheffield and also

fortress-hotel-sheffield if we assume the case that ak-collaborating-institutions meet in several locations then the slot has-location should be defined as (has-location university-of-sheffield fortress-hotel-sheffield).

4.2 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 in the ontology and then determining the type of an object is not possible. Let us consider the following example:

Maria Vargas-Vera visited Dave Robertson

In this case by using the ontology we could validate that *Maria Vargas-Vera* is a person, The answer from the ontology will be true (assuming that Maria Vargas-Vera is an instance defined in the ontology). Then the system will try to validate "if *Dave Robertson* is a place." The system might return false because *Dave Robertson* is not a place or because *Dave Robertson* is not defined as an instance in the ontology. We would like that the system produce an answer like "unknown" like in a open world (OWA). Therefore, in future we would consider to convert OCML to an OWA language.

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

4.3 Experiments

The classification process shows the following results from our experiments. We found that the type event visiting-a-place-or-person is one of the most well defined in the sense that with our set of templates for this type of event we can classify 94% correctly. In others type of event we obtain a performance of 88%. The main reason for this is that our archive of stories has more stories of the type visiting-a-person-or-place than the other type of events. We are currently enhancing the templates for the other event types.

5 Conclusions and future directions

Our system consists of manually encoded templates written as Prolog rules for each event type.

Currently the set of templates needs to be provided by a knowledge engineer but in the future we hope that these will extracted automatically by the system. Our system will be trained using the archive of stories that we had collected in KMi [4] The training step should be performed using typical examples of stories belonging to each of the type events defined in the ontology.

One possible approach to the learning mechanism is to learn our templates using Genetic Programming (GP) since templates are simple programs. Since the early 90's GP has as a goal the automatic generation of computer programs. Currently, this is one of the most promising paradigms for fast, productive software development.

Another approach in a different direction is the use of machine learning algorithms integrated with IE as in the latest version from Riloff called AutoSlog-TS ([Riloff, 1993; Riloff, 1996b; Riloff, 1996a]). It generates extraction patterns using only a preclassified training corpus (one set of texts that are relevant to the domain and one set of texts that are irrelevant). AutoSlog-TS generates extraction patterns by making two passes over the corpus. During the first pass AutoSlog-TS uses heuristics to generate a set of patterns that collectively extract every noun phrase in the corpus. In the second pass AutoSlog-TS computes statistics to determine which extraction patterns are most strongly correlated with the relevant training sets. A ranked list is then presented to a user and they decide which patterns should be kept.

The marriage of IE and Machine learning technology seems promising as described in [Riloff, 1996b]. But even more promising is the combination of IE, Machine Learning and Ontological inference capabilities.

6 acknowledgments

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

References

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

[Domingue and Motta, 2000] Domingue, J. and Motta, E. (2000). PlanetOnto: From News Publishing to Integrated Knowledge Management Support. *IEEE Intelligent Systems*, 15(3):26–32.

[Domingue and Scott, 1999] Domingue, J. and Scott, P. (1999). Kmi planet: putting the knowledge back into media. *The Knowledge Web*, pages 173–184.

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

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

[Lenat et al., 1986] Lenat, D. B., Prakash, M., and Shepherd, M. (1986). Cyc: Using Common Sense Knowledge to Overcome Brittleness and Knowledge-Adquisiton bottlenecks. *AI Magazine*, 6:65–85.

[Maedche and Staab, 2000] Maedche, A. and Staab, S. (2000). 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.

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

[Proux and Chenevoy, 1997] Proux, D. and Chenevoy, Y. (1997). 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.

[Riloff, 1993] Riloff, E. (1993). Automatically Constructing a Dictionary for Information Extraction Tasks. In *Proceedings of The 11th National Conference on Artificial Intelligence (AAAI-93)*, pages 811–816. AAAI press.

[Riloff, 1996a] Riloff, E. (1996a). Automatically generating extraction patterns from untagged text. In *Proceedings of The 13th National Conference on Artificial Intelligence (AAAI-96)*, pages 1044–1049. AAAI press.

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

[Roux et al., 2000] Roux, C., Proux, D., Rechenmann, F., and Julliard, L. (2000). 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).*

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