Logic Programming for Natural Language Processing

Men Young Lee
TJHSST Computer Systems Lab
Mentor: Mr. Matt Parker
Mentoring Firm: Analytic Services, Inc.

March 1, 2006

Abstract

We propose a bifurcated paradigm for the construction of a Prolog knowledge base from a body of documents: first, an information extraction (IE) application that will annotate the corpus and output the annotated documents, and second, a Prolog knowledge base (KB) application that will transform the annotated documents into a KB (a set of facts).

The General Architecture for Text Engineering (GATE) was used as the platform for the development and execution of the IE application, which included most components of A Nearly New Information Extraction (ANNIE) system. Apart from the basic IE capabilities of ANNIE, the application featured additional high-level annotation grammars written in the Java Annotation Patterns Engine (JAPE) language and a trainable annotator that used the maximum entropy machine learning model, which were designed to annotate several biographies of well-known mathematicians. The Prolog KB application, programmed to be executed within the XSB System, was designed to receive the annotated text output by the IE application and produce a knowledge base, and it successfully creates a database of Prolog facts that can be intelligently queried through the XSB System. The KB utilizes the frame representation of facts, specifically by treating one document as an object to be represented as a frame, with each annotation type treated as a slot whose multiple values are whichever specific strings were annotated by the IE application. This transformation of extracted information into Prolog facts forms a link between IE, a recent development in Natural Language Processing, and logic programming with Prolog.

1 Introduction

1.1 Purpose

The purpose of the project was to link the recent advances in text processing and information extraction with classical logic programming in Prolog by first developing a modern information extraction application, which will process a corpus of documents and produce a structured output of extracted information, then developing a Prolog knowledge base application that will populate a knowledge base of facts by transforming the extracted information into a Prolog knowledge base, structured in a frame schema proposed here for representing annotated text.

1.2 Scope of Study

The corpus of documents to be processed will all be within the area of history of mathematics, specifically about two-thousand biographies of mathematicians available online on the MacTutor History of Mathematics archive at `http://www-groups.dcs.st-and.ac.uk/~history/`[8].

2 Background

2.1 Natural Language Processing

Language is the cornerstone of intelligence. Alan Turing stipulated that a machine may be considered sufficiently intelligent if it is able to converse with a human; the Turing Test for artificial intelligence (AI) requires a machine to converse in such a human-like manner as to be able to fool a person. This act of conversing, i.e. understanding and generation of texts written in a natural language such as English, is called natural language processing (NLP).

2.2 Information Extraction

The advent of the Internet and the subsequent explosion of the sheer volume of textual information that is readily available in electronic form, while presenting new opportunities for the exploitation of the available information, has created a difficult challenge, as it is now impossible for an expert to read and analyze all available documents and textual data that may contain possibly valuable knowledge or facts. Therefore it became necessary to develop intelligent systems that will automate at

least some of the tasks involved in the understanding of a document, so a certain restricted and a very practical subset of NLP arose, known as information extraction (IE).[3]

IE, in its most general form, is the transformation of information contained within an unstructured text document into a structured format that is previously defined, either to be used by another AI application such as an expert shell or to be simply presented to the analyst who can then utilize the information while spending a considerably reduced amount of time and effort towards understanding it. More narrowly, IE concerns itself with the identification of instances of certain prescribed classes of objects (e.g. persons) and the attributes of and the relationships of those instances that are relevant to the general objective of the analyst.[7]

The earliest, seminal development in IE was Zarri's RESEDA, a semantic metalanguage used to describe various historical figures and the relationships between them such as familial relationships and meetings.[11] Another much more recent work sought to extract information from personal websites of may university faculty members and and construct a knowledge base consisting of information regarding these faculty members and the relationships between them.[4] Due to the fact that there are many types of "things" that exist in nature, current IE research is entirely focused on carrying out extraction tasks for documents within a single domain of knowledge, wherein there can be a reasonably sized and sufficiently detailed ontology.

2.2.1 The Message Understanding Conference

The Message Understanding Conferences (MUC) were a series of competition-based conferences, where different information extraction systems were evaluated in their performance in extracting information from small news articles within some prescribed domain of knowledge. By its conclusion, the conference had defined five sub-tasks of information extraction:[5]

Named Entity Recognition (NE) , the identification and classification of entities in the text, such as identifying persons, dates, and organizations.

Coreference Resolution (CO) , the matching between different mentions of entities identified by NE as being in fact identical objects. For example, "USA", "United States", and "America" may all refer to the same country, while certain other appearances of the word "America" may in fact refer to something else, depending on the context.

Template Element Construction (TE) finds information that describe the named entities, i.e. attributes of the object instances.

Template Relation Construction (TR) is the recognition of relationships between different identified elements, for example familial relationships between persons.

Scenario Template Production (ST) builds the structured information describing events. For example, a birth event of a person has an associated father, mother, child, date, and location.

2.2.2 Metrics for Evaluation

The MUC evaluations also gave rise to unambiguous quantitative measures of success of an IE application, rather than a much more generic, nebulous concept of "understanding."[1]

Precision is the proportion of correct results out of all the answers produced by the IE system.

Recall is the proportion of the correct results of the IE system, out of the total number of correct responses contained within the document.

F-measure is the combination of precision and recall that requires an extra parameter β, which measures the relative importance of precision (P) and recall (R):

$$F = \frac{(\beta^2 + 1)PR}{\beta^2 P + R}.$$

Note that larger β leads to a larger relative importance of recall, as seen via this more intuitive formulation:

$$F^{-1} = \frac{\beta^2 P + R}{(\beta^2 + 1)PR} = \frac{\beta^2}{\beta^2 + 1} R^{-1} + \frac{1}{\beta^2 + 1} P^{-1}.$$

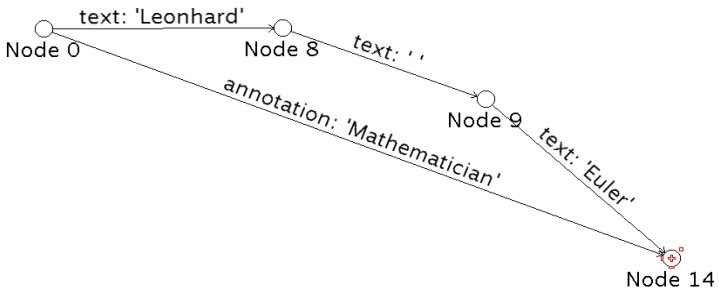
2.3 Logic Programming

Logic Programming is based on the idea of dealing with declarative, rather than imperative, sentences that directly represent knowledge. Through syntax and semantics that allow facts and rules to be represented, while having a full first-order reasoning capabilities, Prolog is the ideal language to deal with information that have direct, human cognitive-level semantics.

3 Development

3.1 Theory: Annotations and Frames

The intuitive representation of NE recognition output is to annotate the document, i.e. identify specific phrases as being an instance of a particular object type. Formally, an annotated text may be considered to be a directed acyclic graph, with a linear sequence of nodes that represent specific locations in the text, and where the literal text and the annotations are arcs pointing from a start node to an end node. Naturally, between two consecutive nodes there is a literal text arc, while an annotation arc leading from a start node to an end node means the annotation of the text between the two nodes, i.e. the concatenation of the sequence of text arcs that lead from the start to the end node.[10]



A frame is a data structure that represents some object, such as a person. It consists of slots, which may contain any kind of information, ranging from simple attributes such as a name, to referencing another object, such as the person's parent. Hence frames are used to structurally store knowledge, in terms of what the objects are, their features, and their relationships to other objects. Bratko gives a simple Prolog fact triple representation of frames: $Frame(Slot, Value)$. [2]

While traditionally frames have been used to describe fully understood knowledge, here we propose a compromise representation, document frames, to represent

what the machine knows about the text itself, i.e. the fact that a certain string in the document received some annotation by treating a document as an object, and annotations as slots. Hence, we give rise to a simple Prolog scheme for representing annotations: $Document(Annotation, Text)$. For example,

```
'Euler.html'('Mathematician', 'Leonhard Euler').
'Euler.html'('Father', 'Paul Euler').
```

There is reason to believe this representation has substantial advantages. First, this representation is transparent to the information extraction application that produces the annotations, i.e. the transformation of the annotation graph into this document frame representation does not need to take domain-specific knowledge into account. Another important advantage of these document frames is that when a corpus of multiple documents are processed, representation of textual understanding with direct reference to the document itself can greatly facilitate the acquisition and use of information regarding the relationships of the several documents to each other. Furthermore, the document-oriented representation of information can make explanation-generation easier for expert systems, by fundamentally and directly representing exactly where a specific piece of understanding comes from.

More generally, the transformation of annotations to Prolog facts opens the door to the application of first-order logic, specifically by applying subject-matter expert rules that pertain to the specific domain of knowledge to which the documents belong.

3.2 Design Criteria

Following the theory described above, there are two design objectives:

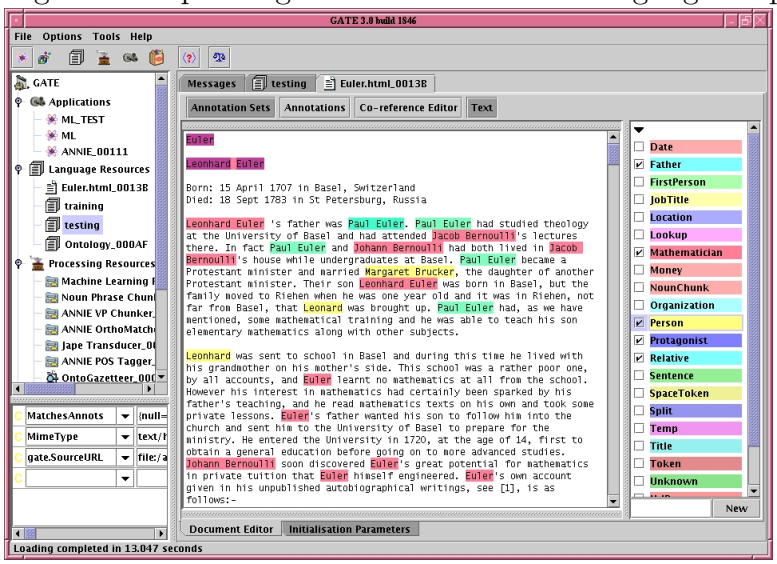
1. Develop an information extraction system that will annotate text. This information extraction system must annotate instances of mathematicians occurring inside biographies of various mathematicians, as well as identify the protagonist of the biography and his parents, in addition to recognizing persons, locations, and dates.

2. Develop a Prolog KB application that will transform the annotated text into a set of facts, organized into the structure specified above.

3.3 Materials—the Software

3.3.1 General Architecture for Text Engineering

The General Architecture for Text Engineering (GATE), first proposed by Cunningham, is a comprehensive architecture for the development and execution of NLP applications. GATE's Document class represents a document with annotations as an annotation graph. NLP applications are created as pipelines, i.e. a string of smaller tools known as Processing Resources (PRs) that perform a particular task, such as string tokenising or performing gazetteer lookup. Several useful built-in tools include A Nearly New Information Extraction (ANNIE) system, and a machine learning PR that trains a maximum entropy model for automatically creating annotations. See a sample screenshot of the GUI, where text is highlighted, with highlights corresponding to annotations on the highlighted portions of the text.[10]



3.3.2 A Nearly New Information Extraction system

The ANNIE system is a generic-cross domain IE system, with rule-based implementations of language parsing, i.e. parts of speech tagging and noun and verb phrase identification, along with rudimentary NE and CO capabilities. ANNIE comes with the capability of identifying persons, locations, and dates, and it can be extended to perform more specific IE tasks through the addition of more JAPE grammars that will specify other context patterns for which annotations should be created.[6]

3.3.3 The XSB System

The XSB System is an extended Prolog system that implements tabling, also known as memoization, that caches results of queries so that they do not have to be recomputed.[9] The major advantage of XSB lies in its efficiency and scalability. The Prolog application will be written specifically to be run by XSB.

3.4 Procedures

3.4.1 The Corpus

The corpus of documents to be annotated was obtained from the MacTutor History of Mathematics archive at `http://www-groups.dcs.st-and.ac.uk/~history/`. A quick, small script in python retrieved almost two thousand biographies of well-known mathematicians.

3.4.2 The IE Application

The IE Application used most of the components of the ANNIE system, while also having extended its gazetteer list to identify names of mathematicians and added JAPE grammars `mathematician.jape` and `mathematician_context.jape`, which identify instances of mathematicians and the protagonist's parents in a biography. In addition, the MaxEnt machine-learning annotator, which uses the maximum entropy model to learn which texts to annotate, was trained to recognize the protagonist. Once the annotations were made, the annotated document was outputted in an XML format.

3.4.3 The Prolog KB Application

The creation of the document frame facts were divided into three stages of processing:

1. The transformation of GATE/ANNIE's XML output into a Prolog structure, by the `transform/2` predicate. defined in `transform.P`,

2. The building of the annotation graph from the document structure, by the `build/2` predicate. defined in `build.P`, and

3. The construction of fact triples from the annotation graph, by the `make_frame/1` predicate. defined in `docframes.P`. This last predicate is the only one that is actually called by the user. It calls the other predicates in order to appropriately perform the steps towards building a knowledge database. It

does not require an output argument; instead it asserts the facts, thus entering them directly into XSB's dynamically handled fact database. Queries may be posed immediately after calling `make_frame(FileName).`

4 Results

4.1 The Predicate Call

```
| ?- make_frame(files(['Galois.html.xml', 'Cauchy.html.xml',
      'Euler.html.xml'])).
[sgmlconfig loaded]
[sgml2pl loaded]

yes
```

This is how the application is to be invoked, i.e. through inputting the file-names of the IE application's output. The answer is "yes" indicating that XSB has successfully executed the predicate and asserted the fact-triples for three frames, one for each document. Note that in Prolog, a "no" answer given following an enumeration of successful instantiations does not mean failure (obviously as the system was successful in finding answers), but that it has failed to find any other answers.

4.2 Simple Queries

```
| ?- 'Galois.html.xml'('Protagonist', X).

X = Evariste;

X = Galois;

no
```

This query asks who is the protagonist of the biography `Galois.html.xml`. Naturally, the correct response is Evariste Galois; Galois is the protagonist of the biography on Galois. The KB correctly identifies the two possible strings that could refer to the character: both his names.

```
| ?- 'Galois.html.xml'('Mathematician', X).
```

```
X = Abel;

X = Cauchy;
```

abridged

```
X = Liouville;

X = Poisson;

X = Vernier;

no
```

The KB correctly returns all the mathematicians whose names appear in Galois' biography.[1]

4.3 More Intelligent Queries

The ability to present more complex and intelligent queries to our knowledge base with little difficulty is a great advantage provided by the use of Prolog.

```
| ?- 'Galois.html.xml'('Mathematician', X),
     'Cauchy.html.xml'('Protagonist', X).

X = Cauchy;

no
```

The query asks for the name of the individual, if he exists at all, who appears in Galois' biography while being a protagonist of Cauchy's biography (i.e. himself). Since as seen above, Cauchy appears in Galois' biography, the correct answer is obtained. One could define a subject-matter expert (SME) rule that defines when two documents may be considered to be linked together in this same fashion.

```
link(DocumentA, DocumentB) :-
        Query1 =.. [DocumentA, 'Mathematician', Mathematician],
        call(Query1),
        Query2 =.. [DocumentB, 'Protagonist', Mathematician].
        call(Query2).
```

[1]Cauchy is mentioned as someone who had badly abused Galois in criticizing his work and being delinquent in providing timely peer-review for his paper. Abel was also a victim of Cauchy's abuse.

The following is another complex query, this time asking for mathematicians that appear in the biographies of Galois, Cauchy, and Euler.

```
| ?- 'Galois.html.xml'('Mathematician', X), 'Cauchy.html.xml'(
      'Mathematician', X), 'Euler.html.xml'('Mathematician', X).

X = Cauchy;

X = Fourier;

X = Legendre;

no
```

4.4 A Discussion About Computational Power

Prolog is a high-level language which requires powerful computers, so that computational power can become a very severe restriction when one tries to execute applications in Prolog. The continuing improvements in the power of computing machinery slowly raises the ceiling on our ability to perform extensively computation-intensive first-order logic, and in fact, it may be that the recent increases in widely available computing power will finally bring about pragmatic feasibility of Prolog applications rather than just being an academic curiosity for theoretical computer science. However, this Prolog KB application is still beyond home use. For successful, crash-free runs of XSB, it required the allocation of two gigabytes of stack memory on a production-grade server in the lab.

5 Conclusions

The efficacy of the proposed two-stage process was demonstrated; the pair of applications successfully process natural language documents into a Prolog knowledge base that can be queried intelligently. The advantages of a document-based frame representation were immediately manifest in the ability to easily find out the relationships between documents, and the ability of the Prolog application to work with a GATE IE application designed to work with any subject matter.

Possible directions for further research in the area include the development of a document frame representation that is more expressive and the addition of SME rules

that are able to infer even more facts not directly specified in the KB. Use of nondeterministic and/or statistical methods for the analysis of language context is another promising direction of research. Computational power, however, will continue to be a limitation.

References

[1] Appelt, D. E., and Israel, D. J. Introduction to information extraction technology. In *Proceedings of the IJCAI-99* (1999).

[2] Bratko, I. *Prolog Programming for Artificial Intelligence.* Addison-Wesley, 2001.

[3] Cowie, J., and Lehnert, W. Information extraction. *Communications of the ACM 39* (1996), 80.

[4] Craven, M., DiPasquo, D., Freitag, D., McCallum, A. K., Mitchell, T. M., Nigam, K., and Slattery, S. Learning to extract symbolic knowledge from the World Wide Web. In *Proceedings of AAAI-98, 15th Conference of the American Association for Artificial Intelligence* (Madison, US, 1998), AAAI Press, Menlo Park, US, pp. 509–516.

[5] Cunningham, H. Information Extraction, Automatic. *Encyclopedia of Language and Linguistics, 2nd Edition* (2005).

[6] Cunningham, H., Maynard, D., Bontcheva, K., Tablan, V., Ursu, C., Dimitrov, M., Dowman, M., Aswani, N., and Roberts, I. *Developing Language Processing Components with GATE Version 3 (a User Guide).*

[7] Grishman, R. Information extraction: Techniques and challenges. In *Proceedings of the Summer Convention on Information Extraction* (1997), pp. 10–27.

[8] O'Connor, J. J., and Robertson, E. F. The mactutor history of mathematics archive, 2005.

[9] Sagonas, K., Swift, T., and Warren, D. S. XSB as an efficient deductive database engine. In *Proceedings of the ACM SIGMOD International Conference on Management of Data* (1994), pp. 442–453.

[10] Tablan, V., Bontcheva, K., Maynard, D., and Cunningham, H. *GATE—An Application Developer's Guide.*

[11] Zarri, G. P. Automatic representation of the semantic relationships corresponding to a french surface expression. In *Proceedings of the Conference on Applied Natural Langauge Processing* (Santa Monica, CA, 1983), pp. 143–147.