An Integration of a Legal Knowledge Based System and a Content Management System for a Legal Education Support System

Seiichiro SAKURAI Hajime YOSHINO
Meiji Gakuin University Graduate Law School
1-2-37 Shirokanedai, Minatoku
108-8636 Tokyo, Japan
{sakurai,yoshino}@ls.meijigakuin.ac.jp

Abstract

This paper describes an integration of a knowledge based system and a content management system (CMS) for a legal education support system. The authors' education method utilizes a legal knowledge based system to help students to acquire the structure of law from statues in a bottom up manner. From the viewpoint of the acquisition of creative legal minds, erroneous knowledge bases are also needed to understand why the erroneous knowledge bases are erroneous. To provide only correct knowledge results in rote learning. For creative lawyers, thinking by themselves is the most important task. Therefore, in our approach, students must select or construct their correct knowledge bases from many alternatives through the comparison. In order to support our approach, a legal education support system should handle many separated knowledge bases which are stored in a database. This paper proposes a method for implementing a number of small knowledge bases on a CMS.

1. Introduction

To encourage the ability of creative thinking is one of the major roles of Japanese law schools. In order to avoid the rote learning of law, the authors have proposed an education method[5] which uses a knowledge based system. By comparing the inferred results of knowledge bases, students can find why a knowledge base is superior to others. However, since the traditional implementation of a knowledge based system is so heavy, simultaneous use of the knowledge based system may cause the degradation of the system performance. If all members of a class access the knowledge based system, the knowledge based system may stop. To solve the degradation of the system performance, this paper proposes a system design by using a light weight Prolog implementation.

A content management system (CMS) is very important technology for a Web based application server because of the

separation of contents and their representations. If a Web based application server does not have a CMS, the maintenance of contents of the server becomes a very hard task. Therefore most of e-learning systems have a function of a CMS or are constructed on a CMS. In order to promote creative thinking, embedding a rule based inference system into a CMS is also an important issue. In our education method, a number of small knowledge bases are utilized to educate legal knowledge through the interaction with law students.

Consequently, the efficiency and the scalability are the main issues in this paper. These issues can be solved by a light weight Prolog implementation and small objects of knowledge bases.

2. Content Management System

Zope[3] is a Web application server written in Python[1], which is a modern script language, and it functions as a Web server. In Figure 1, the Web application server is realized by Zope. Since all of contents of Zope are stored in a database, Zope dynamically generates HTML files according to users' requests.

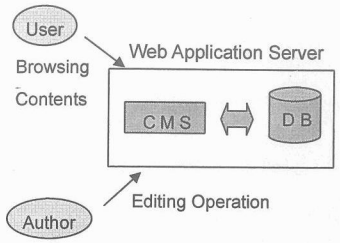

Figure 1: a Web Application Server with a CMS

While Zope outputs HTML files basically, Plone[2], which is a CMS product for Zope, can manage many kinds of contents such as documents, multimedia files, and so on. Furthermore, Plone has many functions for a Portal Web site. From the viewpoint of a system programmer, one of the most important features of Plone is its extensibility. Since Plone is easy to extend, many additional products such as BBS, assignment database or Weblog can be installed easily on Plone. Actually, the authors are implementing a Socratic methods support system on Plone.

3. Inference Engine Written in Python

The basic idea of authors' education method is to let students think the meaning of legal provisions by themselves. Since most of legal provisions are related to other legal provisions or legal concepts, in order to understand the meaning of legal provisions, the students must find hidden relations of legal

provisions. The process of finding hidden relations is basically by trial and error. Therefore, if the meaning of legal provisions is assumed to be represented by rules, many rules, including erroneous rules, are generated and abandoned. In order to ease the burden, providing many rule sets as knowledge bases can be considered as a solution. In that case, erroneous knowledge bases are useful to understand why some knowledge base is superior to the erroneous one.

One of the merits of the knowledge based system is that the knowledge based system can be used to verify the inferred results of a knowledge base. It is not so useful for students to view a knowledge base because of their ignorance of the relations of rules. In order to help students to understand hidden relations of rules, it may be useful to present the inference result and the inference trace of knowledge bases as shown in Figure 2. By comparing the results or inference traces, students can understand why some knowledge base is superior to others.

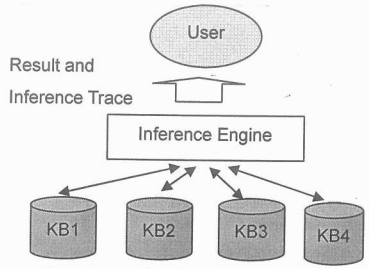

Figure 2: Multiple Knowledge Bases

An inference engine is the most important part of a knowledge based system. The basic inference engine of rule based system is well known as SLD-resolution[6] and its well-known implementation is Prolog[7].

PyLog[4] is a Prolog interpreter written in Python[1]. In PyLog, since the depth first search of Prolog[7] is elegantly implemented by using the "generator" feature of Python, Prolog codes can be run efficiently[1]. The "generator" feature of Python is a control structure like a thread, which is well known as a device of a light weight process, and the generator runs much faster than the usual thread. Thus Prolog programs can be run only with little overhead of term unifications[2].

[1] While one of the most important features of Prolog is the backtracking mechanism, if the completeness of the solution search is guaranteed, the backtracking mechanism is not always needed for an inference engine.

[2] The unification algorithm without occur checking is known to be very

As the Prolog codes are automatically translated into Python classes, any knowledge written in rules can be embedded in a Python program. The most important feature of PyLog, which is at most 2000 lines, is its compactness. Thus the invoking overhead is very small for Plone/Zope. This fact means that our implementation is also effective to other CMS with the ability to invoke external programs. For example, since Python is also implemented on Java, our method is effective to Java based CMS.

4. Examples of Knowledge Bases

Each rule is stored in the database as an object, which has two elements, requirement and effect. Both of the requirement and the effect are character strings which denote predicate names, logical variables and logical connectives. For example, a rule, "if p(x) and q(x) then r(x)", are stored in as follows:

Effect: "r[X]".
Requirement: "p [X] AND q [X]".

The rule object such as above is the primitive of a knowledge base. A knowledge base can be defined as an object which has pointers to actual rule objects. By defining the knowledge base as a small object, the memory space of the knowledge bases is in proportion to the number of knowledge bases.

When a user requests the result or the inference trace of a knowledge base, the knowledge base is automatically compiled into Python classes by using PyLog. For example, the above rule is compiled into the codes in Figure 3.

```
class r(PyLogPredicate):
  def __init__(self, a0):
    self._form_args = Term(a0)
    X = Var('X')
    self._exprs =
     [(Term(X), lambda:And(q(X),r(X)))]
```

Figure 3: Python Class of Prolog Code

Since Prolog predicate "r", which is shown in Figure 3, is defined as a subclass of PyLogPredicate, the provability of "r(X)" can be confirmed by invoking "r(X)" in any Python scripts.

In this way, the workable knowledge base can be stored in the working memory as Python executable codes. This fact can also guard the knowledge base against the leaks.

5. Related Works

Several attempts of implementing Prolog by using client side programming have been done. Since one of the most popular client side programming languages is JavaScript or ECMAScript, some implementations of Prolog are available on the Internet. While the Prolog written in ECMAScript[10] is very portable in the sense that most of Web browser can execute ECMAScript codes, it may leak the program codes of knowledge bases because ECMAScript codes are parts of a HTML file. In contrast to the Prolog written in ECMAScript, each knowledge base can be guarded by the security policy of Plone in our method. The mode of a knowledge base, which indicates whether the knowledge base is executable or viewable, can be easily controllable by the authors of the knowledge base. This is one of the merits of the combination with a CMS.

Another client side implementation can be considered as Java applets[8,9]. Since the Java applets are restrictive with respect to the communication with the server because of the security, a naïve implementation can be that a Java applet must have only a knowledge base. If the number of knowledge bases is small, the naïve approach seems to be a candidate implementation. However, if the number of knowledge bases is big, the cost of downloading the Java applets may be expensive. Therefore, especially in case of ours, the naïve Java applet approach is inappropriate because of the number of knowledge bases.

6. Conclusion

This paper clarifies the implementation issues of our education method. In our method, since each knowledge base must be confirmed by students that it is superior to any other knowledge bases, the support system must store many separated knowledge bases and infer the result of a selected knowledge base. In order to solve the issues, a knowledge based system is embedded in a CMS. Currently, the legal education support system is being constructed.

References

[1]Python programming language, http://python.org/

[2]Plone: a user-friendly and powerful open source Content Management System - plone.org, http://plone.org/

[3] Zope.org, http://www.zope.org/

[4] Delord, C., PyLog – A first order logic library in Python, http://christophe.delord.free.fr/en/pylog/

[5]YOSHINO,H. and SAKURAI,S., A Knowledge-Based Systems Approach to Educating Creative Legal Minds, in ICAIL2005 Workshop.

[6]Lloyd, J.W., *Foundation of Logic Programming*, Springer Verlag

[7] Sterling, L. and Shapiro, E., *The Art of Prolog*, The MIT Press Cambridge Massachusetts

[8] Banbara,M. and Tamura, N., Prolog Cafe: A Prolog to Java Translator System, http://kaminari.scitec.kobe-u.ac.jp/Prolog Cafe/

[9] Minerva, http://www.ifcomputer.com/MINERVA/

[10] ioctl.org: Prolog in JavaScript, http://ioctl.org/logic/prolog-latest

efficient.