A Web-based Intelligent Case-based Reasoning Legal Aid Retrieval Information System

Kevin Curran, Lee Higgins
University of Ulster, Magee Campus, Londonderry, N. Ireland, BT48 7JL
Kj.curran@ulster.ac.uk

Abstract

The Internet has for some time been used by a wide class of lawyers as both a general business tool and more specifically as a research tool. Indications are that this trend is set to increase and infect all areas of legal practice. It should therefore be cultivated and exploited. However as the demand for legal services increases, two related challenges are presented to the Information Technology and Legal Communities. The first being the provision of easy to use services/applications which are cost-effective to develop and which improve the efficiency of the lawyer’s research task and the second challenge; and the second of finding effective means of making such services widely and publicly available across the Internet. This paper demonstrates in the first instance how such services might be created by building on lessons learnt from an investigation into current legal applications. Secondly it examines those technologies that offer an appealing means of realizing the second goal above, with the eventual goal of describing the framework of an application provides the basis for meeting both challenges.

1 Introduction

While the Internet remains a powerful and efficient means of exchanging and communicating information, the standard web-site approach of fetching documents through predetermined hyperlinks or via keyword search through search engines, is not likely to prove overly helpful in satisfying the sophisticated information needs of lawyers. In other words the static nature of the Web seriously limits its usefulness in this context. However the Java language promises to transform the Web into a truly interactive information forum and also provides simple yet powerful means of making services not particularly designed for the web available across the Internet. It could thus help to better meet the aforementioned information needs. At the same time it is crucial (and a central argument in this work) that we recognize that most modern legal information retrieval applications (whether web-based or not) are, for some basic reasons failing to service the requirements of non-specialist lawyers. In light of the above the current situation poses 2 distinct but related challenges for I.T. The first, providing easy to use services which improve the efficiency of the lawyers research tasks and thus effectively meeting the information needs of non-specialist lawyers and secondly, making these services widely and publicly available across the Internet.

Our proposed system aims at supporting the lawyer in his research tasks for a problem situation. Therefore some understanding of the how this task is generally carried out and how we might improve efficiency here is called for. The overriding goal for the lawyer in this context is to get to potentially relevant legal resources (here decided cases and doctrinal writings) that can help him better understand the legal issues he is dealing with and how he might go about tackling the legal problems at hand.

1.2 Improving the Efficiency of Research Task

Making information publicly available i.e. via the Internet, obviously overcomes the problem of obtaining sources which are identified as relevant. However our problem is also one of efficiently identifying which materials could be relevant. It is argued that the efficiency of the lawyer in this context could be greatly improved if an information retrieval system could boast, at the very least, the following functionality

- An interface designed the guidance of a legal expert which ‘walks’ the lawyer through the various possible issues in the case – this interface (through a series of yes/no questions) could help the lawyer build up a profile of his case at hand.

- Once a basic profile is built up the system should indicate (through a process of basic pattern matching) those important cases in the field which best match the profile of the current problem case. We do not aim here at reasoning through stare decisis. Instead the goal is not to retrieve cases which shall be used in an actual court hearing etc,
but cases which are most likely to discuss the kind of issues the lawyers problem case involves. This functionality would therefore serve as a springboard into more intensive and informed research.

- If the system does indicate which cases best ‘match’ the input case, it should explain how this match occurs and also indicate how the retrieved cases are distinguishable from the current case.

- Lawyers search by concept (i.e. legal issues) not (potentially) random keywords. Any index of our document repository should allow the lawyer to find, inter alia, the leading case on a given issue, the latest case on a given issue, important cases where a given issue is discussed and also doctrinal articles where a given issue is discussed.

- When looking at a given case the lawyer should be able quickly to identify other cases where this case was distinguished or cases similar to the case or doctrinal writings where the actual case is discussed.

### 1.3 Problems With Ai-Legal Applications

Despite the intensive and laborious research conducted into such machines they have largely failed to attain their goals and very few have made the transition from research ventures to applied systems. This failure, it is submitted is due to fundamental problems both at the philosophical/theoretical level and the practical level. Firstly all such systems involve the creation of a model of the legal domain – referred to as an ‘ontology’. The overriding goal here is one of representing the knowledge in a manner that is at once computer encodeable, and at the same time remains true to the meaning of the original source material. Making this knowledge computer encodeable almost always involves viewing the law as a (fixed) set of rules. It is almost universally accepted that the law is slightly more complex than this. The law is not self-contained and autonomous; instead it’s meaning must be interpreted in the light of many implicit and ever-changing assumptions in the political and social context. It is seriously doubted whether current technologies can handle such a complex model. It thus follows that this process of isomorphism has yet to be achieved and representing legal reasoning in a computer encodeable form involves a certain distortion of the subject material.

Given the work involved in building a satisfactory model, it comes as no surprise that such machines are notoriously costly to develop, and given the underlying complexity they are extremely difficult to maintain (ease of maintenance being one of the cornerstones of any applied system) and update [1]. Developing intelligent systems that can easily handle change is no trivial matter and this problem is all the worse if we accept that the law is a notably fickle and changeable creature [2]. Furthermore unlike other areas of AI the complexity involved in automating or providing support for legal reasoning means that no generic commercial shells are available and most systems (capable of covering only one or two legal problem areas) must be built from scratch [3]. In addition such applications (whether EBS, KBS or DSS) fail to recognize the realities of legal practice in the sense that they tend to place too much emphasis on the law as an entity embodied in written texts rather then the product of an oral tradition. Computer technologies should therefore assist with mechanical research/retrieval tasks and not delve into more creative (and inherently uncertain) task of legal reasoning. We might also ask ourselves whether such machines have a large enough target audience to justify the massive effort required in building them. To make sense of the complicated output they produce the user must have already a considerable knowledge of the target area of the law and sophisticated I.T. skills - qualities missing in our target (and most) users.

In addition we might note that the complex reasoning strategies and output they produce are likely only to be of use in cases decided in the highest courts in the land (about 1%) [4]. The CBR process of comparing cases based on the notion of factors is, it is argued, relatively easy to replicate. It is also quite useful to (and a common strategy adopted by) lawyers who use it not for any substantive purpose of legal reasoning but to identify cases that could help them better understand the issues involved in their case. Bearing this limited goal in mind our system shall attempt to implement some form of basic pattern-matching mechanism.

### 2 Legal-Aid Database Implementation

Most modern web applications (or indeed any category of application) to be truly interactive, informative and useful require access to structured data, which is not embedded in the application. Such data is most usually (and usefully) contained in databases. Database management system (DBMS) products, consist of a series of programmes which together offer highly effective means of managing the data. Through the use of powerful data definition languages and data manipulation languages (DDL, DML) such as SQL, these products offer an excellent basis for populating, querying and otherwise communicating with databases.
We choose to connect to the online database using the Java Database Connectivity (JDBC). The JDBC API allows for efficient development of multithreaded database applications allowing for almost seamless integration with powerful middleware solutions of the Java family. The API basically defines a number of Java interfaces, which enable developers to use Java as the host language for applications which access data independently of the actual database product. In essence JDBC shields an application from the specifics of individual database applications. Hence interoperability, combined with platform independence is JDBC’s major selling point in this context [5].

2.1 The Major Components of the System

The system/application proposed here basically aims at improving the lawyer’s research task by providing web-based access to a legal document repository, which resides on the server. To improve the efficiency of information retrieval performed on this document corpus the user can request documents by one of two methods – Legal Database Servlet or Case-Match Servlet. The components operate as follows (Figure 1).

- Legal database Servlet

User at web browser connects to the server and requests the database service inputting data into HTML forms (1), this data is passed to the servlet (2), which uses it to run a query against the database and receive results (3), the servlet formats this data into HTML tables and returns this to the client (4), using the returned results the user makes a request for documents from the server (5), the server retrieves these documents (6) and returns these to the client (7).

- Case-Match Servlet

User at a web browser connects to a server and submits a profile of his problem case (1), this data is passed to the Case-Match servlet (2) which runs a match against stored cases, selects the best matches, formats the results into HTML, and returns these to the user (4). Based on the information returned the user makes a request for the server for documents (5), the server retrieves these (6) and returns them to the user (7).

The representation of the documents referenced (legal cases and doctrinal writings) aims at providing efficiently for information needs. We represent the documents in the database in accordance with our target area of the law being broken down into a series of ‘factors’ symbolising legal issues. These issues/factors (we identified 8 for demonstration purposes) may or may not be present in our case or doctrinal writing. Here these 8 factors are denoted $I_1$ to $I_8$. Thus a document can be described in terms of the issues it deals with. Importantly a document can also be described in terms of other cases. The database contains tables made up of rows (and columns), which correspond to our legal documents. Each document is represented as a tuple, having a unique identifier (Cnum/Anum) with the other attributes being used to describe various facets of the document referenced (See Figure 2 and Figure 3). The attribute values are used for query and retrieval purposes.

<table>
<thead>
<tr>
<th>Cnum</th>
<th>Chname</th>
<th>Citation</th>
<th>Date</th>
<th>Link</th>
<th>Verdict</th>
<th>Lead</th>
<th>Mem</th>
<th>Dist</th>
<th>Similar</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>C2</td>
<td>24/01/1973</td>
<td>P.1</td>
<td>G1</td>
<td>yes</td>
<td>G2</td>
<td>C3</td>
<td>C12</td>
<td>C3</td>
</tr>
<tr>
<td>C2</td>
<td>C3</td>
<td>24/01/1973</td>
<td>G2</td>
<td>P.2</td>
<td>no</td>
<td>G3</td>
<td>C4</td>
<td>C12</td>
<td>C4</td>
</tr>
<tr>
<td>C3</td>
<td>C4</td>
<td>24/01/1973</td>
<td>G3</td>
<td>C4</td>
<td>yes</td>
<td>G5</td>
<td>C6</td>
<td>C12</td>
<td>C6</td>
</tr>
</tbody>
</table>

Figure 2 : The legal case table

The column headers include (for cases), the unique identifier, name, law reports citation, date of judgement, the full address of the document on the server and the verdict of the case (either pro-plaintiff, or pro-defendant). These fields are pretty self-explanatory. However certain other fields will represent the opinion of some expert in the legal area –
• **Lead** – Is this case the leading case on an issue? i.e. the most important case. If so we insert the identifier of the appropriate issue.

• **Main** – What are the main issues discussed in this case? Insert the identifier(s) for the most important issue(s) discussed here.

• **Distinguished** – Has this case been distinguished in any other case? If so insert the unique identifier(s) for the appropriate case(s).

• **Similar** – Are there any cases which closely resemble this one? If so enter the unique identifier(s) for the appropriate case(s).

Please note that instead of using multi-valued attributes here we use continuous strings. Thus, for example, case C1 above is the leading case on Issue 1, the main issues discussed in the case are issues 2, 3 & 6, it hasn’t been distinguished in any case but is similar to case C3.

<table>
<thead>
<tr>
<th>AName</th>
<th>Author</th>
<th>Citation</th>
<th>Link</th>
<th>Date</th>
<th>Issue</th>
<th>Case</th>
</tr>
</thead>
<tbody>
<tr>
<td>IPC1</td>
<td>J. Smith</td>
<td>[1990] 12LR 264 1st half = 100756  1147</td>
<td>C1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IPC2</td>
<td>J. Jones</td>
<td>[1991] 12LR 211 2nd half = 100562  1123</td>
<td>C023</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IPC3</td>
<td>B. Brown</td>
<td>[1992] 13LR 95 1st half = 100548  8588</td>
<td>C013</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IPC4</td>
<td>P. Blue</td>
<td>[1993] 13LR 54 2nd half = 100750  12718</td>
<td>C4</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Figure 3 : The doctrinal writings table**

The ‘Article’ table is structured along the same lines as the ‘Case’ table, also having similarly column headers for name, author etc. The ‘Issue’ field informs us as to what legal issues are dealt with in the doctrinal writing. If any cases are discussed this is stated in the ‘Case’ field. For example article A1 above deals with issues 4 and 7 and discusses case C1. Overall this structure allows us to move away from the ‘false drop’ inducing keyword search. The number of possible queries on the database is potentially massive. For this prototype application we have created only a select number of queries that correspond to the most likely information needs. The types of query we specifically cater for here include -

• Find the latest case on a given issue
• Find the leading case on a given issue
• Find those cases where a given issue is discussed in detail & doctrinal writings which deal with a given issue
• Given a case find cases similar to that case & given a case find cases where that case distinguished
• Given a case find articles where that case discussed
• Find the latest case/article on the general area

**2.2 The Web-Based Interface**

The interface (Figure 4) is presented in the lawyers own terms i.e. the user is invited to find leading cases, main cases etc. according to a certain legal issue, or the user can request cases which are distinguished from the current case he is reading. This enables the lawyer to more easily make specific and meaningful queries. The lawyer basically selects a legal issue within the appropriate form and submits by clicking ‘Find’. The ‘Find Latest Case/article’ forms at the top of the page do not allow the user to specify input parameters – instead here we use ‘hidden’ fields.

**Figure 4 : Legal Aid Web Page**

When reading a legal case or article the user may also access the database. In this instance however the user does not specify any input data and the data sent is decided according to the actual page we are on (i.e. the value attached to the name is the case identifier). By clicking the form buttons the user submits a query to the Servlet. A successful request returns a HTML table to the user which contains details of and a link to, documents satisfying the request. The user can then follow the link to obtain more details on the document.

**2.3 The Case-Match Servlet**

Here we describe how we might use a strategy based on that in the previous to perform a case-match function. To recap, the main goal of this component of the proposed application is not to produce highly complex and detailed reasoning strategies which devices such as HYPO and CATO generate but instead to provide the lawyer with a useful guide as to what important cases could prove useful in their reasoning task. As such a very simple pattern-matching algorithm/method, based on the model is employed. This basically operates on the simple premise that those cases which bear the most
positives similarities to the lawyer’s current problem case are most likely to be of use in building an argument which supports the lawyer’s case. The corollary here is that those cases with more negative factors (i.e. TCS and PDS factors) will generally not be so beneficial.

The value the lawyer would take from such an application would reside in the output of a core set of cases that could help form the basis of the legal reasoning process. Here, no attempt is made to simulate or usurp the reasoning task of the lawyer, instead an effort is made to speed up the initial input (the legal research), which forms the basis of this task. Importantly the core cases returned to the user would be described in terms of comparison to the current problem scenario (i.e. in terms of PPS etc. so that the user could easily determine how the retrieved case might be used in their favour or to their detriment\(^1\)). The case-match Servlet operates with a web browser which the user uses to create a profile of their problem case by answering a set of yes/no questions. The more info column allows the user to access materials that help him to better answer this question. This leads to a series of HTML documents which help clarify what constitutes a Yes or No with regards to each question (as this can be a grey area at times). Data is submitted in the same way as with the previous Servlet. The parameters input by the user are then compared against the pre-defined database data. Comparison would be based on the PPS/OCS/TCS/PDS approach. At the same time, a counter is set up to help determine the best matching cases. The count of each pre-defined case is represented as a cell in an array. What forms a good match (and scores highest) is based on the simple approach described above. For example PPS could score 2pts, OCS 1pt, TCS 0pt and PDS –1pt. Once all comparisons have been run, we search for and identify the highest (possibly 3) scoring cases. Once the best matching cases are found a second round of comparisons is run (between the best matching cases and the user-defined case). The goal here is to classify each retrieved case in terms of PPS/OCS/TCS/PDS factors. We then use the output from this to generate ‘HTML-on-the-fly’, and send a response back to the user which describes the ‘best’ matching cases in terms of PPS etc and also attributes such as name, citation, verdict and location.

### 3 Conclusions

The Internet (or more specifically the World Wide Web) has become the forum for information gathering and will surely be an essential tool of all modern lawyers. The Java language can enable us to transform the Web into a truly interactive law library. Although more information is available on the web, the efficient and effective retrieval and management of these web documents are still very challenging research issues. Intelligent information retrieval involves much more than retrieving free text, it involves systems that enable users to create, process, summarise, present, interact with (e.g., query, browse, navigate), and organise information within and across heterogeneous media. When navigating the web, with such a vast collection of linked documents, users can easily get lost in its depths. Information retrieval also poses users problems in finding appropriate resources and extracting information from within documents. Text and relational databases can be searched on content and indexing terms. Most modern legal information is contained (or can be structured as we have described) within databases. The use of JBDC can enable us (remote client) to efficiently access this information from a central location where the ever-changing knowledge base can be controlled and updated as required.

### 4 References

3. Hunter & Zeleznikov, supra
5. Ablan, Developing Intranet Applications with Java (1996), Sams.net

---

\(^1\) e.g. if a case exhibits PPS factors then these factors could be used to advance the current case and should be stressed. However if a case exhibits TCS factors then we must explain why these factors are not of crucial importance in this case if we wish to use the case in our favour.