

Intelligent Jurisprudence Research: a new concept*

Rosina Weber

CCEPS/ULBRA

R.L.Linhares,657/204-A, Florianópolis

SC,BRAZIL,88.036-002

+55 48 333 1292

rosinaw@zaz.com.br

ABSTRACT

Intelligent Jurisprudence Research (IJR) is a concept that consists in performing jurisprudence research with a computational tool that employs Artificial Intelligence (AI) techniques. Jurisprudence research is the search employed by judicial professionals when seeking for past legal situations that may be useful to a legal activity. When humans perform jurisprudence research, they employ analogical reasoning in comparing a given actual situation with past decisions, noting the affinities between them. In the process of remembering a similar situation when faced to a new one, Case-Based Reasoning (CBR) systems simulate analogical reasoning. Therefore, CBR is an appropriate technology to deal with the chosen problem.

Keywords

Artificial intelligence and law, jurisprudence, case-based reasoning, case-based retrieval.

1 INTRODUCTION

Jurisprudence research is the search employed by judicial professionals when seeking for past legal situations that may be useful to a legal activity. Intelligent Jurisprudence Research (IJR) is a new concept that consists in performing jurisprudence research with a computational tool that uses Artificial Intelligence techniques.

Jurisprudence research is performed by legal experts when faced by a new legal situation that demands knowledge, experience, or grounds to define a legal action, a prediction, an argument, or a decision. Legal experts have always practiced jurisprudence research by searching and comparing similar legal situations in books that publish legal decisions. In some legal areas, obsolescence does not take place and the amount of jurisprudence keeps growing as time goes by. The search in books performed by

humans is constrained by a small working memory. On the other hand, the similarity assessment performed by humans is perfect since human experts are perfectly capable of assessing a given situation in order to identify its potential utility to a new legal episode.

The spreading of personal computers made evident the need for a computational tool to help legal experts in performing the task of jurisprudence research. However, the only tool that Computer Science is able to offer is text databases. The assessment remains performed by legal experts and the computational tool helps both in retrieving decisions that share some words and in reducing the amount of decisions to be evaluated. Hence, legal experts assess the similarity of the documents that are actually retrieved after a keyword search. The use of text databases is restricted by a low quality of retrieval although offering fast and large memory resources. This means is insufficient to yield results with a minimum reliability. Hence, this task demands a solution that combines the efficient assessment of the potential utility with the memory resources of a computational solution.

According to Blair and Maron [1] text database systems are limited to a recall that provides only 25% of useful documents. Evaluation in text databases can be performed by two parameters: recall and precision [7]. Recall is the proportion of useful documents to the total of useful documents in the base. Precision refers to the ratio of useful documents to the total retrieved. Moreover, the precision range varies, causing a low efficiency that could be extremely dangerous in legal domains.

The low accuracy of the text databases stems from the use of statistical methods of indexing. Statistical methods do not consider knowledge, *i.e.*; they blindly (without knowledge) select terms depending upon their frequency of occurrence. By contrast, the similarity-based retrieval employed in CBR systems can be exclusively based on knowledge. A knowledge-based indexing process guarantees more efficiency, because the indexes guiding similarity and retrieval are chosen with expertise knowledge, enhancing the chances of retrieving useful experiences. In addition, the knowledge-based indexing avoids low levels of precision since the chances of retrieving useless experiences decrease.

Case-Based Reasoning is an Artificial Intelligence technique that models aspects of human cognition to solve expert problems. CBR systems mimic the human act of reminding a previous episode to solve a given problem due to the recognition of their affinities. The main issue in case-based systems is the similarity assessment as it is responsible for providing the proper representation of the analogical reasoning simulated in the

*In Proceedings of the Seventh International Conference on Artificial Intelligence and Law (ICAIL-99), 164-172. Oslo, Norway: ACM.

system. The correct similarity assessment is supposed to yield the same result as if a human expert had performed the evaluation.

The most important contribution of the present work is to provide an efficient means to support jurisprudence research. Whenever a past legal decision is not retrieved, an injustice may prevail. Since human working memory is not capable of storing and evaluating large amounts of past situations, it is important that science provides the technology to compensate humans in such limitation. One major benefit from the development of large CBR systems in the legal domain is to make possible reuse of the knowledge embedded in jurisprudence that is used to reference new court decisions. Improving the access of past legal cases enlarges the horizon from where new decisions are grounded, consequently raising the quality of the results of the judicial system. This research is an attempt to orient available technology in the pursue of a just society.

The feasibility of this system depends upon the case engineering requirements of the development of a large case base that comprises the whole universe of judicial cases. Judicial cases are described with natural language text that represents a hard-to-use form, therefore demanding case engineering efforts. We have learned that hand coding this case base represents an unfeasible task. This is because even if we gather sufficient time and human resources, the coordination demanded to ensure a consistent result still seems to be impossible. Therefore, we have gathered our resources pursuing an automatic means of performing the case engineering what, in essence, concerns to an automatic indexing.

Instead of modeling Law as a real object we face Law as an abstract target that is pursued by legal professionals. The teaching of Law is about spreading a small body of principles of domain theory what is excelled by its substantial content: the legal view of the world. The real object we choose to represent is the way legal professionals interpret legal facts. We have found in legal experts the main source about this view of legal knowledge. Our knowledge acquisition processes attempt to elicit the way legal experts interpret and view the world.

The conversion is presented by the use of a methodology. First, we employ the development of the methodology; consisting of the work required to prepare automatic procedures to implement the methodology. The feasibility of the use of this methodology requires that the texts are regularly structured and expert knowledge of the content and context of the texts. The second step, implementing the methodology, refers specifically to the construction of the cases by assigning values that represent the useful knowledge present in every text.

Values are assigned automatically with template mining techniques. We have combined Information Extraction (IE) techniques that extract information from unrestricted texts to a database with Template Mining. In an IE task, a number of database entries (slots) to be filled is specified together with either a limited set of possibilities for descriptors for each slot or some specification for open-ended values. The values for the slots are strings from the source text [5][6]. Template Mining is a template technique that extracts data from texts when the text forms recognizable patterns from the target to be extracted or its surroundings. A template carries information on what to search in the text and it is triggered to extract the parts indicated [4].

Follows the main steps of the proposed approach. (a) Define attributes employing the reminding approach. (b) Read a

sufficient amount of texts to ensure you identify their structure, *i.e.*, what is the content of each paragraph. (c) Assign values to the attributes searching in the proper substructure of the text and confirm attributes with functional approach. (d) Associate the values to each substructure in the text. (e) Design methods to extract the selected values from each substructure. (f) Identify synonyms. (g) Extract the values mapping texts into cases. (h) Review extracting process until experts identify that attributes not assigned are exceptions.

The methodology is illustrated throughout the article with the current prototype of PRUDENTIA [9]; a system developed to demonstrate the approach. The current prototype embodies a collection of 3,447 experiences that have been autonomously converted into cases. The knowledge acquisition effort has been performed by groups of legal experts and knowledge engineers. The cases represent the experience of all criminal appeals that were submitted in the State Court of Santa Catarina in the period from 1990 to 1996.

2 THE SOLUTION

The solution we are presenting overcomes the knowledge engineering obstacle of converting textual decisions into cases by defining case attributes to represent legal decisions and employing template mining methods [4] to automatically extract values to the attributes. Cases are the formalism that represents domain experiences within the CBR environment. When modeled as cases, experiences can be manipulated in the intelligent system, *i.e.*, they can be assessed, retrieved, and reused.

The prerequisite of the present approach is that texts are similar in their structure. Since this condition is met, the first step in the development of the methodology is a preliminary evaluation of the experiences by the experts relating these experiences to the task of the system. This allows the definition of the attributes to comprise the formlike representation of the cases. Then, text analysis yields texts modeled into substructures. Next, the defined attributes are associated to the substructures where their values are supposed to be. Then, we create methods to elicit these values from the specified substructures and the knowledge required to extract values for the attributes within each substructure is identified. Finally, the methodology is amenable to implementation.

2.1 The Attributes

The solution we are proposing enables the development of a CBR system to retrieve legal texts based on the fact that case representation is a knowledge-based task, especially within the domain of law. The case representation step primarily consists of indexing cases. Approaches to perform the indexing represent a complementary foundation for the solution. Definition of indexing vocabulary is a domain expert task and thus we assert that it is deployed by human experts through a knowledge acquisition step.

The indexing process embodies the definition of the indexing vocabulary and index assignment. The knowledge acquisition required to implement the methodology of converting texts into cases can be divided in two stages. The first stage concerns the initial knowledge acquisition that aims at the definition of the content and context that must embody cases to fulfill the task of the system. The second refers to defining the attributes that

comprise the issues that better represent the experiences described in the legal decisions to achieve the respective task.

The most important characteristic of the retrieval in CBR systems is that it should be guided by the usefulness of the cases. Determining descriptors (attribute-value pairs) that will be useful in solving a similar problem is not a trivial task. In extracting attribute values, we use the functional approach and the reminding approach [3], although the descriptors we extract will not always be used as indexes. Some of them will be filled with expressions that we cannot expect to be automatically compared with other sentences from legal texts. This is because we are trying to get the most out of the texts regarding the number of lessons that will be presented to the user, however we cannot expect that all these lessons will be guiding the retrieval. Above all, lessons can be solutions, although we are using lessons to guide retrieval.

The initial evaluation is an attempt to identify how texts can achieve the system's task. In the IJR concept, the task is to provide the user with lessons of the most similar legal experiences to support the legal situation that originated the research. In PRUDENTIA, the attributes defined are:

Petition type. Single. The current example comprises only petitions of *criminal appeals*. Former versions contemplated also petitions for *habeas corpus*.

Number. Single. The ordering number given to order legal decisions.

Reporter. Single. The name of the reporter who issues legal decisions.

District. Single. Place where the act that triggered the lawsuit has taken place.

Page. Single. Localizes the decision in the original file.

Date. Single. Date of decision.

Foundation (1,n). Foundation is the basis on which an appeal stands, is founded, or is supported. List of 312 expressions.

Theme (1,n). After defining index foundation, experts claimed there was still something missing in the description of the content and context of legal decisions. The problem was that this aspect missing in some of the experiences differs from what was missing in others. Since CBR theory suggests that an index is to be defined when it can be valued in every case of the memory, knowledge engineers had to come up with a solution. The solution we have found is to define multi-purpose indexes [9]. These indexes comprise multi-purpose values that can be classified in different classes. For example, the index theme for the *habeas corpus* motivation would have the value *canceling* and the case representation would store the class motivation in another attribute, that is not an index. Therefore, by giving to this index a generic name theme, we assign as many values as necessary to represent one case, regardless of the nature, overcoming the necessity of recognizing all the values to every index in all cases. List of 393 expressions. The class is not explicit.

Secondary laws. (1,n). Secondary laws that may be brought up to support formal actions. An article number and a source.

Category. (1,n). The primary categorization of the original act such as a felony or a misdemeanor that corresponds to an article

of a law or the Constitution. List of sixty-seven (67) different categories within the criminal appeals.

Result. One of three. The court decision, either positive, negative or neutral.

Unanimity. One of two. The text informs whether the decision was made by majority of votes or by unanimity.

2.2 Text Analysis

The process we call text analysis is decisive with respect to the feasibility of the whole methodology. In the domain illustrated, the texts we map into case format are legal decisions expedited by a state court. Judicial reporters share very similar backgrounds, interact on a very regular basis, and follow some rules in the write-up of legal decisions. Consequently, these texts are highly stereotypical, making them amenable to computational treatment.

Branting and Lester [2] have proposed the self-explaining document framework claiming that knowledge of the illocutionary and rhetorical structures of complex documents can be used for indexing. The notion of illocutionary structures and expressions provided the required contribution to define functional substructures in the legal decisions.

The knowledge acquisition step allowed us to come up with a rhetorical structure of these texts. This way of envisioning texts is very helpful in determining the accomplishment of the proposed methodology. Texts are analyzed by domain experts to define their rhetorical structure and identify the parts containing illocutionary expressions or simply expressions that relate that portion of the text with a purpose. This is also a knowledge-based task. Experts who write these legal texts follow some rules. For example, texts always present an abstract after the heading, and the two last paragraphs of the abstract always start with the same words. This makes this task easier and allows easy computational treatment. The format of these legal texts motivated us to delineate substructures such as *heading* and *abstract*. Legal texts used in PRUDENTIA presented the structure as shown in Figure 1.

The identification of these substructures facilitates the computational treatment. Methods to select these substructures are mostly linguistically based. *Heading*, *abstract*, and *closing* offer no difficulties for selection. *Body: categorization* is a fuzzy one. However, the indication of the district where the incident has taken place and the occurrence of the article of the law helps finding this substructure.

The analysis of samples of legal texts by domain experts resulted in the observation of some repeated expressions that are usually present indicating relevant information about the case. Together with the identification of illocutionary expressions, another important ground of this process is the identification of *indicative expressions* [11]. There are four types of indicative expressions. (a) Nouns derived from adjectives: they express a condition; e.g., impossibility. (b) Sentences with verb to be in which the noun is an object of the domain: express an information about the state of the object; e.g., custody is, victim has been, evidence was, perpetrator is, defendants were, etc. (c) Verbs that are objects of the domain: indicate facts; e.g., certify, arrest, allege, prove, etc. (d) Adverbs meaning for this reason, indicate conclusive lessons; e.g., therefore, ergo, hence, thus, therefore, accordingly, consequently, so, etc.

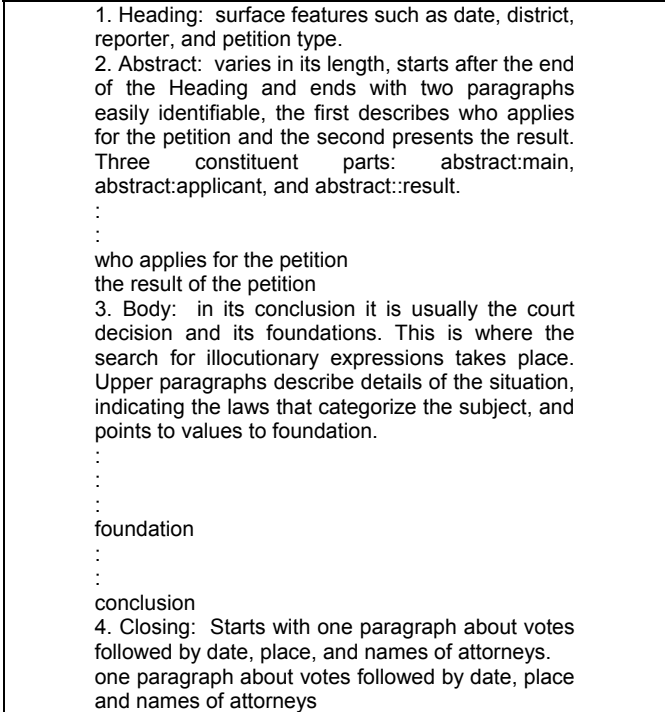


Figure 1 Rhetorical structure resulted from text analysis.

Body:conclusion can be identified with the occurrence of indicative expressions of the fourth type (adverbs meaning *for this reason*). When the function is not capable of finding the expression, the alternative employed is a heuristic suggested by legal experts, starting from the end of the *body* because the *conclusion* is about there.

2.2.1 Associating Attributes to Substructures

Text analysis aims at defining substructures to feed template mining methods since these methods require the directions about what and where to search in the source texts. Defining and associating substructures help pointing where to search. Lists of possible values help pointing what to search.

As long as substructures and the knowledge to model the rhetorical structure in a computer program are identified, experts indicate the substructure where values for each attribute can be extracted. Results are laid out in table 1.

attribute	substructure
petition type	heading
reporter	heading
date	heading
district	heading
number	heading
page	heading
category	abstract:main and body: categorization
result	abstract: result
unanimity	abstract: result
theme	abstract:main and body
foundation	abstract and body:conclusion
secondary laws	whole text

Table 1 Where to search for attribute values.

Substructure *abstract* consists of three subportions: *abstract:main*, *abstract:result* and *abstract:applicant* that indicate different sets of paragraphs that are consistently and clearly distinguished from the others. The clear distinction of the substructures is a starting point to employ template mining methods to extract attribute values, as mentioned in the preceding section.

2.3 Methods of Extraction

The information extraction is implemented via template mining methods that search for expressions indicated *a priori* in specific portions of texts. Since we have texts divided into substructures and every attribute is associated to a given substructure, the aspect left to contemplate is what to search, and it is also obtained with expert knowledge. Inherently, the result of this knowledge acquisition process is a list of possible values contained in texts constituting the list searched by template mining methods.

The knowledge acquisition step aims at enumerating the collection of possible values that might occur to represent every case. The knowledge acquisition is performed for every attribute of the case description through an iterative process starting from small samples.

Once template mining methods have been designed embodying all knowledge required, the execution of these methods result in the knowledge-based valorization of the attributes in the formlike representation of cases. Let us now comment on some specific aspects of the actual conversion of texts, examining some methods used to ascribe values to each attribute, accomplishing the modeling of cases. The knowledge required to identify the appropriate substructure and to extract the proper values is embedded in the methods.

The process of developing template methods is incremental in each sub-domain. First, we implement template mining methods based only on the experts' orientations and execute the method to verify its efficiency. The first iteration is based on the *reminding approach* [3] used to the indexing vocabulary. Experts think on the types of possible values they are reminded of and consider relevant to express each attribute. The result is a list of possible values that can be assigned to the attributes. The first test is held by the implementation of a mining function that searches in the associated substructure for the list of expressions provided by the experts. The function extracts all the instances found. This result is evaluated by the proportion of legal decisions in which values for indexes have been extracted. Whenever the function is not able to extract any value, it returns a flag value named *fail*. Knowledge engineers review the occurrences of this value to determine the proportion of failures. Texts in which no values are found are then reviewed by legal experts in order to elicit new knowledge to represent in the method. The number of texts represents the new population where experts have to look for expressions. A sample is evaluated and a second iteration starts. This process is repeated until function extracts valid values from every text or experts recognize texts that, for some reason, do not carry values for the given attribute.

The programming languages vary due to the different nature of problems and because of the individual knowledge and experience of the knowledge engineers that joined our team. We have used

from Amzi!Prolog¹ to WordBasic² for textual functions and Borland Delphi³ for object-oriented applications and translators.

It is important to point out that the definition of the attributes does not require the experts to examine a significant amount of texts. Their capability of pointing out these attributes relies on their expert knowledge of the domain.

Once the extraction is complete, *i.e.*, every text has values assigned for the given attribute, the values are reviewed in more detail. Some attributes require that the respective values are treated by a sort of translator. Experts identify polymorphic expressions that can avoid retrieval if kept as they appeared originally. This translator is a function that searches for some identified expressions and substitute them for one expression with the same meaning that has been chosen to cases. After translation, knowledge engineers review the resulting list searching for mistakes such as broken words (sometimes misprints may cause the extraction of only some letters). Finally, the list of values actually extracted is given to the experts in another attempt to identify errors in the extraction that will be converted as assignments in the cases.

The values extracted for all the attributes are gathered and all cases of the case-based reasoner are build at once. The case-based reasoner has an interface to the user for the input of new cases. This interface presents as options of possible values all the actual values that have been extracted from the texts to feed the cases (Figure 2). This device guarantees that experts and knowledge engineers are continuously checking the ascribed values to ensure their correctness.

Before running extraction methods, it is required that all texts are converted into ASCII format to facilitate the implementation of textual functions. As the essence of the jurisprudence paradigm, legal decisions refer to past decisions to substantiate their arguments. Hence, we have to exclude portions of other legal decisions that are included in texts for illustrative purposes. This may be a facultative decision depending specifically on the domain since past decisions are only referenced if they are useful.

After this brief view of the approach used to all extraction methods, let us now see individual descriptions of the methods adapted to each attribute. Some attribute values require that more than one kind of template mining technique is employed and in more than one substructure of the legal texts (Table 1). The way values appear in the texts demand different treatments that have been implemented differently as follows.

2.3.1 Category

Category refers to the law that has originated the lawsuit, *e.g.*, a felony or a misdemeanor. The format of the attribute is one of a list of words that represent the title of the article of the law. The possibility to value the attribute with the number of the article or law stands for future developments. Meanwhile, we use the

number of the law or article as a means of eliciting the textual value for the attribute.

In the current prototype, the values comprise a list of 67 different categories within the criminal appeals. Examples are: larceny, DUI, theft, homicide, counterfeit, extortion, kidnapping, drug dealing, bankruptcy, arson, charlatanism, prostitution, damage, child desertion, offense, gambling, assault, child desertion, illegal possession of drugs, abuse, etc.

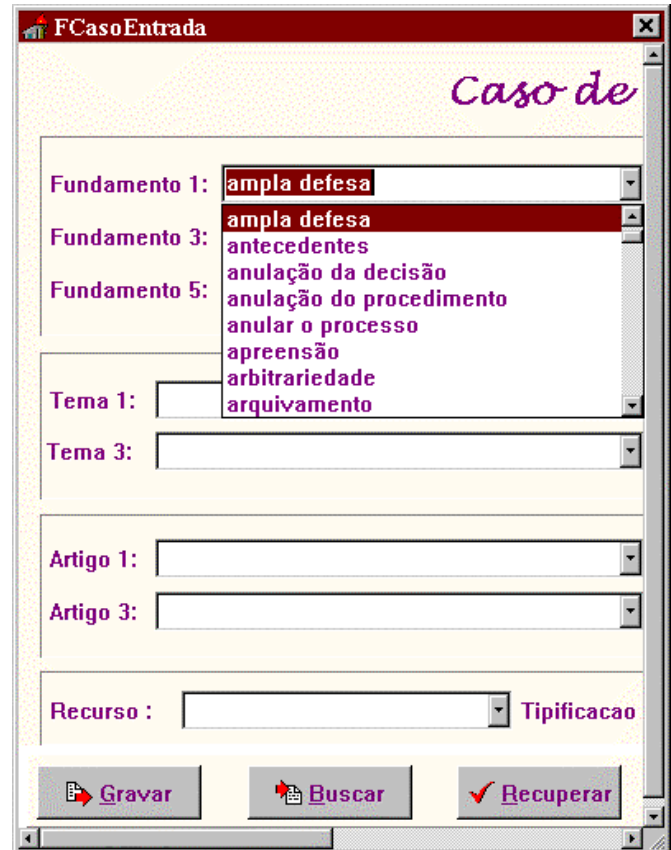


Figure 2. Possible values for input.

The method for extracting values for *category* is oriented to search for values first in the substructure *body: categorization* and afterwards in the substructure *abstract: main*. This first substructure is a paragraph that usually brings the specific article, law and source. It is written in a sequence following the district where the felony has been committed and after an expression equivalent to "for infringing articles 26 & 97 of Penal Code". The second substructure is an alternative found by the legal experts where the category can be extracted as well.

The method extracts the article and its source and translates the information to the title of the category. A list relating article numbers and category titles has been built from text occurrences and corrected with the official codes. In the current application, out of 3,447 texts, this first step has ascribed values in about 2,600 cases (75%). The remaining texts either do not mention the category article explicitly or it mentions it without detailing the source or article or presents a new category that was not part of the list.

¹ Amzi!Prolog 3.3Mar96, Copyright ©94-95, Amzi! Inc.

² Microsoft language present in versions like Microsoft Word for Windows 95 Version 7.0 Copyright © 1983-1995 Microsoft Corporation.

³ Borland Delphi Version 3.0 (Build 5.53) Copyright 1983-1997 Borland International.

The first step of this method selects the header information of every legal text and finds the substructure *body: categorization*. Methods to select this substructure were implemented in WordBasic and Visual Basic⁴ for one first selection and a function in Borland Delphi completed the task. The search for the article number is conducted by the function programmed in Borland Delphi that confirms the article number with a list of categories. Whenever only the category expression or the article number and source is found, the remaining information is added based on formal knowledge. The resulting report informs every failure in assigning categories.

The second step in this method is held by a search on the substructure *abstract: main*; this step is held exclusively searching for the category expression. Article numbers are seldom described in this portion of the text. In the *abstract* the category is mentioned frequently. In the current prototype, this process resulted in about 800 assignments for the attribute *category*. The remaining 100 texts left were returned to the experts for evaluation. Here we face a difficult problem: recognizing when a given value is not present in the original text. We know when we successfully assign a value, but if the extraction does not happen, we do not know why. Sometimes the problem is that the petition is a very special case and there is not much information to be extracted. If an item has to be added to the list, experts confirm the addition. The incremental process remains until every case is properly valued.

We do not employ a specific evaluation phase to check the assignments at this point. We assume the association to the substructures is correct and that if, for instance, the expression *larceny* appears in the referred paragraph it means the value for category is actually *larceny*. However, when every assignment is done, the values are input in the case-based reasoner and every value assigned can be viewed in the list of possible values shown in an interface. This list is then reviewed by experts continuously and corrections can be done.

2.3.2 Foundation

One of the first indexes identified by legal experts was index *foundation*. It represents legal aspects or material facts that substantiate an appeal or its decision. Examples are commercial purpose, confiscated, good cause, first offender, insanity, *non compos mentis*, negligence, circumstantial evidence, alibi, confession, etc.

The foundations occur in different portions of the texts depending if they were mentioned as part of the applicant's argument or basing the judge's decision. To the assignment, we make no distinction.

A very interesting hint concluded from the knowledge acquisition is that some specific expressions simply cannot mean anything else but a foundation, especially if they are mentioned in certain portions of the text. Thus, experts come up with heuristics where we can assume that an expression such as *first offender* or *negligence* necessarily indicates a foundation if it appears in those

substructures. Hence, the choice made by legal experts support the correctness of the values.

However, some expressions have the semantic meaning guaranteed when extracted from the *abstract* that does not hold for the substructure body. Hence, this method has an intermediate stage that treats some words depending on the substructure from where they have been extracted. Examples would be expressions such as *blame* and different conjugations of the verb *to confess* that reveal their relevance in that decision if they are mentioned in the substructure *abstract*. These expressions within body may be simply part of an explanation, not necessarily indicating an important issue within the content of the decision.

The strategy deployed to extract values for foundation is the direct search for matching values with the list of expressions in the respective substructures. The most relevant values for foundation are usually given in the *abstract: main*. We mine first in *abstract: main* and afterwards in *body: conclusion*.

The attribute foundation is multiple-valued. This is because it is an attribute tailored to represent different aspects grounding the appeal. Moreover, we search for expressions in different substructures resulting in many values. Although it is possible that some texts may bring only one or two values for foundation.

After extracting a number of values from a collection of cases, experts have noticed the occurrence of polymorphisms. The list of expressions had to be reviewed by experts who indicated expressions that work as synonyms within the legal context to avoid misrepresentation. For instance, the words *jail*, *custody*, *prison* and *penitentiary* and sometimes the verb *to arrest*. This list of synonyms improves efficiency augmenting the retrieval of useful cases in a human fashion. The list of values has currently 312 expressions.

The extraction method is composed by functions that select the required substructure and reads the text searching for the selected 312 expressions. For instance, the search for expression *alibi* that is one of the values for foundation. The function attempts to find the occurrence of *alibi* in the portion of the text. One function responsible for searching all the expressions is composed of subroutines as the one laid out in Figure 3.

```

REM starts search
REM expression alibi routine #1
StartOfDocument
EditFind .Find = "[AaÁá]lib?", .Direction = 0, .MatchCase = 0,
.WholeWord = 0, .PatternMatch = 1, .SoundsLike = 0, .Format = 0,
.Wrap = 0, .FindAllWordForms = 0
If EditFindFound() = - 1 Then CharLeft 1 : WordRight 1, 1 : EditCut :
Activate "out.doc" : EditPaste : InsertPara : Activate "tempo.doc"
REM ends search

```

Figure 3. Example of routine Find.

The search for the Portuguese word for *alibi* demands further explanation. The Portuguese word has a stress shift, which is eligible to misspellings and misprints as any other word. This is why we write more than one routine for one expression at times, in order to avoid missing the extraction due to misspellings. The wild cards allow us to give options of acceptable digits and even a

⁴ The programming languages embedded in word processors such as WordBasic and Visual Basic are an easy way of prototyping procedures that require textual functions such as "Find" and the use of wild cards at the same time.

wild card such as “?” let us accept any digit. Care is necessary in using these cards or we cannot guarantee the values.

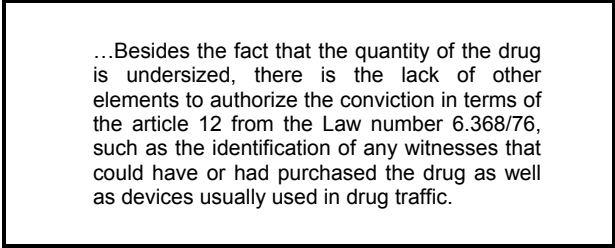
Legal experts also help in indicating frequent misspellings that actually vary in each language. This is another kind of experiential knowledge that contributes with the correctness of the approach.

The extraction method returns a report containing header values of the text and the list of values for foundation found in both substructures searched. The texts that come out empty (with the flag *fail*) are evaluated by the experts. If the number of texts without assignments is over fifteen (15), representing a time-consuming task to review every text, we proceed the evaluation in pilot samples. Then, when we had sixty-five (65) texts missing the value for foundation, experts examined five (5) texts and added the new values. Another iteration runs and the experts examine only samples of texts.

2.3.3 Secondary laws

The index *secondary laws* refers to articles of laws that are mentioned throughout the texts. This may happen when a different categorization is pursued or, for instance, when no substantial matter is to be considered due to an annulment caused by some formal reason. Legal problems that can even cause a mistrial are usually formalized in laws.

The law articles may indicate arguments used by one of the parts in validating an assumption. The different sources of law demand for a two-dimensional valued attribute represented at the level of the number of the article or law and the source, such as article 12 of Federal Constitution. An example is laid out in Figure 4, an excerpt from a legal text translated in [9].



...Besides the fact that the quantity of the drug is undersized, there is the lack of other elements to authorize the conviction in terms of the article 12 from the Law number 6.368/76, such as the identification of any witnesses that could have or had purchased the drug as well as devices usually used in drug traffic.

Figure 4. Excerpt from a legal text.

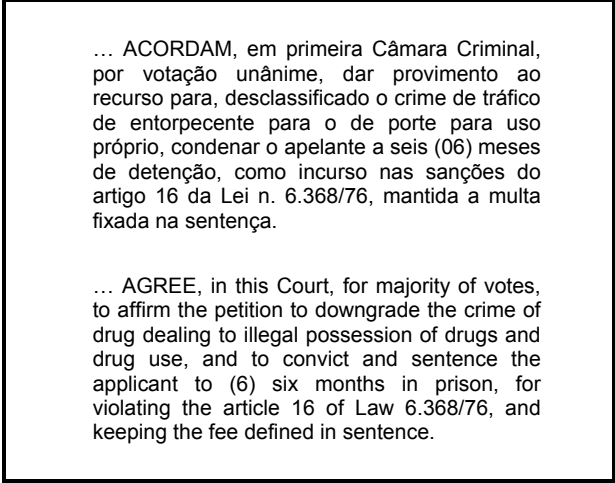
In this example, the value of the article extracted is 12 whereas the source is *Law number 6.368/76*. The correctness of this assignment is verified with the existence of such an article and such a law in the formal theory.

Values for secondary laws are not searched in any specific substructure but in the whole text. The method is implemented by first extracting the articles that refer to the categorization, since they are the value of the attribute *category*. Then, we select only portions of the text that contain numbers. The mining is used to find the valid sources after expressions such as article and its variations through wild cards.

2.3.4 Unanimity

In the substructure *abstract*, the last paragraph starts with a sentence where the value for unanimity can be extracted: the substructure *abstract:result*. In the occurrence of a dissenting opinion, text reads “*under majority of votes*”. In contrast, text

reads *unanimously* or a similar expression, when all opinions are unanimous. This substructure is extremely easy to be selected since it starts always by the same way; reporters even use capitals to highlight this paragraph due to the importance of the *result*. An example is shown in Figure 5 with an excerpt of an actual decision followed by a non-technical translation.



... ACORDAM, em primeira Câmara Criminal, por votação unânime, dar provimento ao recurso para, desclassificado o crime de tráfico de entorpecente para o de porte para uso próprio, condenar o apelante a seis (06) meses de detenção, como incurso nas sanções do artigo 16 da Lei n. 6.368/76, mantida a multa fixada na sentença.

... AGREE, in this Court, for majority of votes, to affirm the petition to downgrade the crime of drug dealing to illegal possession of drugs and drug use, and to convict and sentence the applicant to (6) six months in prison, for violating the article 16 of Law 6.368/76, and keeping the fee defined in sentence.

Figure 5. The result of a legal decision.

There are at most ten inflections to express unanimity, making it very simple. The value for unanimity is Boolean, because it is either unanimous or not. Hence, we need to have two functions: one finds and extracts the given substructure, the other searches for one of the ten occurrences of the list in this portion, and returns the value extracted to an outcome report.

If the function is not capable of finding any of the ten expressions, it returns the flag *fail* and experts review the new way of expressing the unanimity to incorporate to the function. This type of iterative process guarantees success even if reporters tend to change style of writing.

2.3.5 Result

There are different fashions to express if a petition has been affirmed or not, and these forms vary in correspondence with the type of petition. This is one of the initial experiential knowledge obtained in the knowledge acquisition processes. The substructure indicating the result is very stereotypical.

After extracting values for the attribute unanimity, we already have the report with text headers and the substructure *abstract:result*. Because the result depends upon the petition type, the extraction method employs a kind of a demo rule verifying the petition type and orienting to a specific knowledge base where the respective list of expressions is.

For instance, in petitions for *habeas corpus*, the verb used to express its acceptance is *conceder*⁵ (concede, affirm, accept), whereas the verb *denegar* (refute, reject) is used to reject the petition. In different types of petitions, other verbs are employed to express acceptance, such as the verb *prover*, that is a synonym

⁵ Verbs in Portuguese.

of *accept* although it is not used in certain types of petitions. This information is obtained by the knowledge acquisition step. It narrows the problem in a such a way that we can draw rules as, "If petition type is habeas corpus then search in the substructure *abstract:result* for the verbs *conceder* and *denegar*".

In this method, as well as in others, a flag indicates when no result is extracted. It may happen and it usually means a neutral result when nothing is actually decided.

In the example presented in Figure 5, one can easily comprehend that the extraction method returns the verb *affirm*. At this point the report presents the header and the verb. Another translator converts the values extracted into the proper result positive, negative, or neutral.

2.3.6 Theme

The index theme refers to some secondary aspects or circumstances that model the cases. The complexity of these indexes stems from the fact that they were defined to complete the universe of the attributes in describing the content and context of the experiences on legal decisions.

As explained above (2.1) the index *theme* actually has a class but it has not yet been valued in the current prototype. The list of expressions totals 393 values. Examples are: quantity of drug, cocaine, civil imprisonment, mental health evaluation, drug dependency required, annulment, canceling, abatement, mistrial, traffic accident, strikes, penalty reduction, cruelty, break jail, etc.

The knowledge acquisition process to define the possible values for theme was incremental. First, we took about twenty-five (25) texts to start the process of identifying values. Then we ran the extraction method and incrementally the legal experts examined 5% of the texts that had no assignments. At every iteration, new classes were defined. The method to assign values for this index completes the task of automatic index assignment as the definition of this attribute has completed the task of case representation.

In the current prototype, index *theme* has been valued via template mining employed to the substructures *abstract:main* and *body*. The extraction method is very similar to the one applied to extract values for foundation.

3 INTELLIGENT JURISPRUDENCE RESEARCH

The ultimate result of the IJR concept is the construction of a case-based reasoner. The memory in this case-based reasoner consists of the cases built with the attribute values extracted from the legal texts. A case is the formalism used to represent and manipulate experiences in an intelligent system. Legal experiences are represented with legal cases. The illustrative example of an implementation of the Intelligent Jurisprudence Research concept is the system named PRUDENTIA.

PRUDENTIA searches for legal situations that can be useful in teaching lessons to support reasoning about a new situation that the user inputs. The system returns similar situations that are found through analogical reasoning simulated by the CBR inference. The case-based reasoner performs analogical reasoning comparing a new legal situation to the legal descriptions in the case base and returns a set of similar situations. The basic architecture in which PRUDENTIA is built upon is laid out in Figure 6.

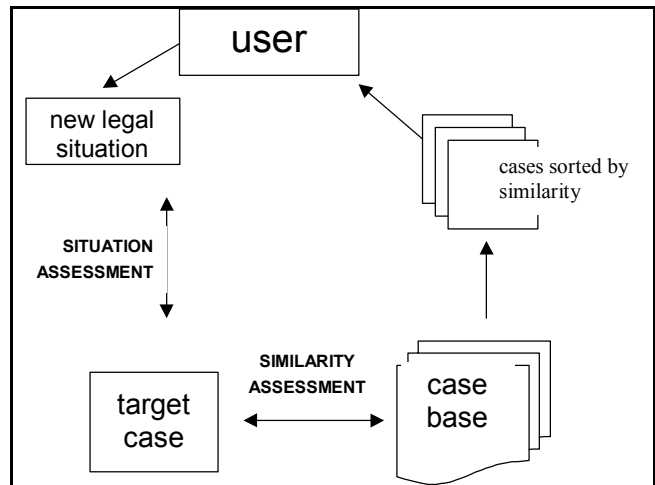


Figure 6. System architecture.

The inference in PRUDENTIA starts with the identification of a new legal situation. A judicial professional faces a new legal situation that requires jurisprudence research while performing usual legal activities. This legal professional accesses the system with an interpretation of this new legal situation in mind. The system attempts to elicit the new legal situation from the user's mind through the process of situation assessment. Situation assessment methods infer values to assign the attributes in the formlike representation, modeling the new legal situation in the same fashion as cases in the case base. The system then compares the new legal situation to every candidate case in the case base. A similarity metric measures the value of each similarity that is used to sort candidate cases to be offered as the outcome of an iteration. Further explanations and examples in [9].

4 CONCLUSIONS

The conversion of texts into cases solves the problem of implementing the IJR concept. The solution is based on the use of expert knowledge and template techniques. Expert knowledge together with proper theory tell us how to build the indexing vocabulary and we mine the values with template techniques. Once indexing vocabulary and template mining methods are in compliance with the respective theories, and functions are designed and implemented in accordance with Computer Science principles, we assume the indexing is appropriate. Besides the revisions performed both by legal experts and by knowledge engineers, the use of the resulting case-based reasoner continuously by this team guarantees correctness of the indexing.

Textual CBR systems outperform text database systems in efficiency in retrieving knowledge and information from texts. The advantage of CBR systems stems mainly from the knowledge-based approach to indexing that is the essence of similarity-based retrieval. Since statistical methods select only indexes with a medium ratio of occurrence, terms that appear very often or very seldom may not be selected. In addition, terms with similar meanings might be selected misleading retrieval results. This seems to be the main reason of the low efficiency of such systems that supports the use of knowledge-based systems to the legal domain [10].

Comparing with the alternative of retrieving legal decisions with text databases we can guarantee that the concept of IJR offers better levels of recall and precision. A better recall is obtained since more indexes are always used and the intelligent jurisprudence research paradigm does not eliminate useful texts due to one single non-matching index. A better precision is based on the overall evaluation of similarity that sorts all texts showing the documents in order of similarity.

One important conclusion from our research is that keeping facts of life together with domain theory simplify the process of representing legal knowledge. The knowledge representation approach adopted in the present research is the modeling of the interpretation of the way legal experts envision the world. Instead of modeling Law as a real object we have faced Law as an abstract target that is pursued by legal professionals.

In researching the alternatives of treating texts, we have been discouraged from making use of traditional natural language processing techniques, such as sentence and semantic analysis, because they have been designed for tasks like machine translation and are therefore not suitable to aid document retrieval [8]. Biased by this idea, we have conducted our research avoiding those techniques, what turned out to work effectively. The use of template mining to extract expressions to index and represent cases was enough to our purposes. We have combined template mining with the use of lexicons, word stems, and wild cards.

The texts used in the development of the methodology are descriptions of legal decisions issued by an intermediate appellate court written by reporters who produce highly stereotypical texts. The consistent and homogeneous structure of these texts was decisive in the success of our methodology. The requirements to implement the methodology are structured texts and previous knowledge of the domain.

5 ACKNOWLEDGMENTS

I would like to acknowledge the anonymous reviewers for their comments and suggestions that have strongly contributed to the inclusion of relevant information appropriate to the AI&Law community. Also and always I want to acknowledge and thank the legal experts and knowledge engineers that helped me throughout this research.

6 REFERENCES

[1] Blair, D.C. and Maron, M.E. (1985). An Evaluation of Retrieval Effectiveness for a Full-Text Document-Retrieval System. *Communications of the ACM*, 28, 3, 280-299.

[2] Branting, L. K. and Lester, J. C. 1996. Justification Structures for Document Reuse. In *Advances in Case-Based Reasoning: third European Workshop; proceedings EWCBR-*

96, 14-16, Lausanne, Switzerland, Ian Smith; Boi Faltings (ed.), Berlin: Springer.

[3] Kolodner, J. (1993). *Case-Based Reasoning*. Morgan Kaufmann, Los Altos, CA.

[4] Lawson, M.; Kemp, N.; Lynch, M. and Chowdhury, G. (1996). Automatic Extraction of Citations From the Text of English-Language Patents – An Example of Template Mining. *Journal of Information Science*, 22, 6, 423-436.

[5] Lehnert, W. (1993) *Natural Language Processing Overview* (from the 1993 Research Brochure for the Department of Computer Science at the University of Massachusetts, Amherst) in *Natural Language Processing Laboratory, University of Massachusetts*. Available Online <http://www-nlp.cs.umass.edu/~nlpgroup/nlproj.html> [August 20,1996].

[6] Lehnert, W. (1996) *Information Extraction. Short introduction to information extraction systems*. In *Natural Language Processing Laboratory, University of Massachusetts*. Available Online <http://www-nlp.cs.umass.edu/~nlpgroup/nlpie.html#what is IE?>, [August 20,1996].

[7] Salton, G. (1975). *Dynamic Information and Library Processing*. Prentice-Hall, Inc. Englewood Cliffs, New Jersey.

[8] Smeaton, A.; Kellely, F.; O'Donnell, R.; Quigley, I.; Richardson R. and Townsend, E. (1995). *Information Retrieval Applications: Using Linguistic Resources or Using Language Processing*. *Proceedings of IA'95 Language Engineering Conference*, pp.65-74, June1995, Montpellier, France. In Alan Smeaton's Online Publications. [Online] Available <http://lorca.compapp.dcu.ie/~asmeaton/pubs-list.html>, [September 12,1996].

[9] Weber, R. (1998) *Intelligent Jurisprudence Research*. Doctoral Dissertation, Graduate Program of Production Engineering at the Federal University of Santa Catarina, Brazil. May, 1998. In *Activities*. Available online <http://www.eps.ufsc.br/~rosina/html/activities.html>.

[10] Weber, R.; Martins, A.; Barcia, R. (1998). On Texts and Legal Cases. *AAAI Fifteenth National Conference on Artificial Intelligence*. July 26-27, Madison, Wisconsin. *Workshop on Textual Case-Based Reasoning*.

[11] Weber-Lee, R.; Barcia, R.; Costa, M.; Rodrigues Filho, I.; Hoeschl, H.; Bueno, T.; Martins, ^a & Pacheco, R. (1997). *A Large Case-Based Reasoner for Legal Cases*. *Lecture Notes in Artificial Intelligence: 2nd Int. Conference on CBR, ICCBR97*. David Leake, Enric Plaza (eds.)Berlin: Springer.