Information & Communications Technology Law

Volume 6, Issue 1, 1997

LASER: A system to retrieve UK employment law cases

DOI:
10.1080/13600834.1997.9965753

Mohammad Ali Montazeri[a], Trevor J. M. Bench-Capon[b] & Alison E. Adam[c]

pages 41-54

Publishing models and article dates explained

- Published online: 10 May 2010

Abstract

In this paper we describe LASER, a system designed to retrieve cases in the domain of UK Employment Law. Two issues are discussed in detail: the representation of cases, and the matching of cases with the facts of a new case under consideration.

Details

- **Published online:** 10 May 2010

Routledge
Taylor & Francis Group

Author affiliations

- [a] Department of Electrical & Computer Engineering , Isfahan University of Technology , Isfahan, 89156, Iran
- [b] Department of Computer Science , University of Liverpool , L69 7ZF, UK
- [c] Department of Computation , UMIST , Manchester, M60 1QD, UK

CK91A

LASER: A System to Retrieve UK Employment Law Cases

Mohammad Ali Montazeri
Dept. of Electrical & computer Engineering
Isfahan University of Technology, Iran
Trevor J.M. Bench-Capon
Dept. of Computer Science, University of Liverpool, UK
Alison E. Adam
Dept. of Computation UMIST, UK

July 9, 1996

Abstract

In this paper we describe LASER, a system designed to retrieve cases in the domain of UK Employment Law. Two issues are discussed in detail: the representation of cases, and the matching of cases with the facts of a new case under consideration.

1 Introduction

It is universally agreed that past cases are an essential input into legal reasoning about a new case. Past cases inform our interpretation of the legislation governing a case, by showing how the terms in that legislation have been interpreted in the past. At the extreme, past cases can determine the interpretation to be taken in a new case, although one must be wary of taking the doctrine of *stare decisis* too literally, as work on concept drift has shown [12]. The question remains as to how these cases are to used in computer systems designed to support lawyers in their day to day work.

One approach is to interpret the decision in a case as a rule [2]. This approach, however pragmatically useful it might be, lacks flexibility in that a single interpretation of the case is embodied in the system - an interpretation, moreover, which lacks authority. In practice decisions on cases themselves stand in need of interpretation, and no interpretation is definitive. Perhaps more importantly, the connection with the case and its circumstances is lost: the assumed *ratio* of the case is accepted and incorporated into the system without its context.

Other approaches have chosen to represent cases explicitly, rather than compiling the knowledge to be derived from a case as rules. Within this approach we may distinguish four positions:

- The cases are represented as free text, and retrieval is Boolean keyword in context style search. Commercial systems such as LEXIS and WESTLAW are well known examples;

- The cases are structured and retrieved according to their closeness of match to some current case, but all reasoning with cases is left to the user. This is often termed *conceptual retrieval*, and [14] can serve as an example;

- The cases are structured and *stare decisis* is taken very seriously so that the system can offer a decision on a current case based on the outcome of the most closely matching case(s). An example would be [9];

- The cases are structured in such a way that they can not only be retrieved, but also reasoned with - for example, arguments developed from them. This is real *case based reasoning* and is well exemplified by the work of Rissland and her colleagues [1], [15], [11].

The system we will describe in this paper falls squarely in the second of these categories. The first kind of system has its place, but as has been convincingly argued in e.g. [3], it does not address the true needs of a legal information system. Too many irrelevant cases are produced and too many relevant cases are missed. The third kind of system places a reliance in past decisions beyond that which we feel appropriate. The fourth kind of system, although without doubt the most potentially powerful, envisages a greater degree of analysis that we believe to be practical in the kind of environment in which we locate our system. We will return to this issue when we discuss case representation.

The system we will describe, LASER (Legal Assistant for Employment Regulations) is designed to interact with a large body of relatively lightly analysed cases, so as to retrieve the set of cases which are considered relevant to a case on hand. It operates in the domain of UK Employment Law.

LASER uses standardised cases, isomorphic to the original knowledge source (it legal precedents), in a flat memory organization. Standardized cases have a predefined case structure with a fixed number of features. The features are selected to reflect the important aspects of a legal case. The feature values are used as dynamic indices in order to perform an exhaustive search of the case-base.

Two matching mechanisms are used in the retrieval process, matching of corresponding features and cross-structural matching. Corresponding matching, matches the equivalent parts of the target and the source cases and calculates the degree of similarity according to the number of features matched, and their degree of importance (weights). Cross-structural matching is used in a complementary way to the corresponding match. The cross-structural matching is used to compare non-equivalent features, in the same category or related categories, even though there is a weak correspondence between the features.

2 Case Representation

If cases are to be used in a system, they must be represented. At one extreme no structure is imposed, and the case is simply stored as a free text representation of the decision. At

the other extreme, a great deal of structure can be imposed: for an example of a very highly structured representation see [5]. Obviously, the more structured the representation, the more effort that is involved in representing the cases. Typical is a representation which lies between these extremes, imposing some structure, but a structure concomitant with the resources available to analyse the cases, and with the needs of the particular application.

2.1 General Considerations

We can think immediately of three classes of information that we need to hold on cases. First we need some identification of the case and contextualisation (date, level of court and the like). Second we need to represent the features of a case in some way; if we are to match a new case with similar previous cases, it will be largely on the basis of this information. Third we need to represent the outcome of the case, since this be vital in exploiting the retrieved cases.

It is the features that are problematic. One reason is that we can represent features at a number of different levels of abstraction. Consider a hypothetical employment law situation in which it is crucial to determine whether the plaintiff can opt to retire. It may be that retirement can be opted for either if the person concerned is over pensionable age, or if the person concerned has completed a certain number of years of reckonable service. In turn pensionable age may be dependent on age and sex, and the required number of years of reckonable service may vary according to occupation. We could record four facts: age, sex, years of service and years required. Alternatively we could record the derived information regarding pensionable age and whether the required years have been served. Finally we could abstract further and simply record that the person is able to opt for retirement.

The temptation is to record as much detail as possible. But this temptation should be resisted. For if we record the lower level facts divorced from their significance, we may get spurious matches. Were we to do this, two cases might be regarded as similar in that they involve occupations with the same service threshold, and yet the other facts may make this of no relevance. Moreover, other cases may make no use of these facts at all, entirely other issues being concerned in those cases. To include such facts would in these situations be misleading. Further, if the matter was not considered when the case was decided, the information might be unavailable from the records, giving rise to incompleteness of information in our case base.

It seems therefore that the matching needs to take place between the abstracted *issues* rather than the lower level facts (although in certain cases a low level fact may itself be regarded as an issue). If we include the detailed facts in our representation we must also include the means to relate them to the issues. This is the approach taken in HYPO [1]; where the low level facts are called *factual predicates* and the issues applicable on their basis *dimensions.*

Is it, however, necessary to store the factual predicates at all? In HYPO the answer is yes: recall that HYPO is a case based reasoning system, and its main focus is on the way in which cases can be manipulated once they have been retrieved. For this manipulation the factual predicates are vital. As an example consider the dimension *Competitive Advantage Gained.* This is calculated by comparing the development time and cost of the

plaintiff's product with the development time and cost of the defendant's product. It is possible to strengthen the plaintiff's case by increasing the development time or cost, or reducing the defendant's development time or cost. If the potential to make these kinds of adjustment is to be retained the lower level facts must be represented. Note, however, how meaningless it would be to match two cases on the basis of one of these factors in isolation: defendant's development time has significance only as an input to the calculation of competitive advantage.

Our objectives in implementing LASER, however, did not extend to this manipulation of cases, but stopped at retrieving cases. There would seem therefore no requirement in LASER to store such facts - but only the dimensions, or issues, that applied in the case, since it is on these that matching must be based.

Having decided that the case features we need are issues abstracted from lower level facts, we need to decide how they should be represented. One method would be to have a number of boolean slots, one for each issue, and for each case this will record if the issue arose in that case. This approach is that taken in [9]. We rejected this approach for several reasons. First it is inflexible: it is hard to deal with new issues, and hard to deal with cases that refine an issue, and so give rise to some sub-issues. Second the representation is somewhat wasteful: only a small number of issues relative to the overall number of issues is likely to apply to any given case. Third and probably most importantly, this approach does not allow us to make distinctions with regard to the issues: which issues were crucial in determining the outcome, and which had less impact; which issues are general and which rather specific. These distinctions and those like them can be used to improve the relevance of the cases retrieved.

We therefore chose to have slots representing the roles played by the issues in the case under consideration, and to fill them with a list of issues that played this role.

In the next sub-section we will describe the specific representation of cases used in LASER.

2.2 Case Representation in LASER

Our starting point for producing a suitable representation for LASER was the case representation format of the EC Law Book, shown in Figure 1.

The bulk of this information concerns the identification of the case, but it also includes a set of very general terms suggestive of its content. These are, however, on their own, too broad for our purposes, and need to be augmented by some more specific characterisation of the issues.

In LASER standardised cases are partitioned into four major parts; case denotation, case content features, case outcome (conclusion) and case text. Each part comprises several different features. The hierarchical relations between the case features are shown in Figure 3.

Case denotation has four features which serve to identify the case; *case number, case title, case date* and *case source*. The case *level of court* is also sometimes used to identify the case, but is not grouped with the other denotation features, since it has significance beyond mere identification. The denotation features are included for reference purposes, and their combination provides a unique label for the case.

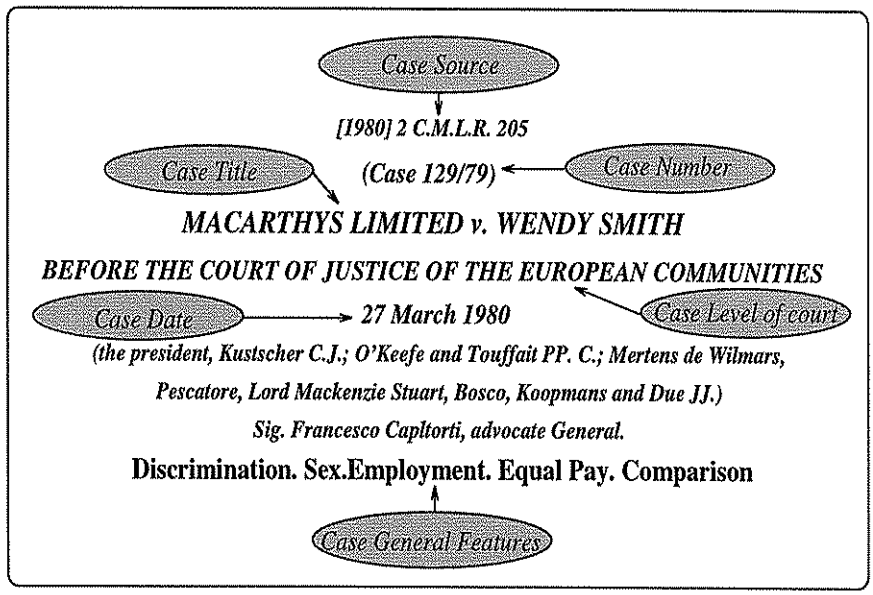

Figure 1: Case Representation Format in EC Law Book

The main body of a case is made up of three types of content feature; *general features* which are abstract legal terms common to many cases, *specific features* which present more specific issues found in a particular case, and *crucial features* which are those features (which may be either general or specific) which were of particular importance for the case in question. These three classes of content features are produced by interpretation of the case description from a legal perspective and are highly useful cues for retrieval. These three content features have two other important roles in relation to retrieval from case memory. Firstly, they are used as indices to the cases. Secondly, they are used to define the structural relations of cases in the context of the whole case memory. For example, general features (which are the most abstract features of cases) are used to define the category of cases.

The conclusion (or case outcome i.e. the information that may be most usefully transferred to a target case) is made up of two types of feature; the *result* and the *references*. The result is simply the judicial outcome of the precedent case which can be used as direct evidence in the argument of the target case. The references (other cases cited in the precedent case) do not directly provide useful evidence but act as a highly useful cue to other cases that are likely to be relevant for application to the target case. The case feature, *level of court*, as well as the identification role is significant because it determines the importance of the precedent case outcome. For two similar cases, one with the higher level of court carries more legal weight.

The final case features, *facts* and *text*, provide the basic description of a case. The *facts* provide a summary of the main features of the case whereas the *text* is simply a comprehensive textual description of it. The values of these features are stored prior to legal interpretation of the case, hence their power as cues for retrieval is low (though not necessarily zero).

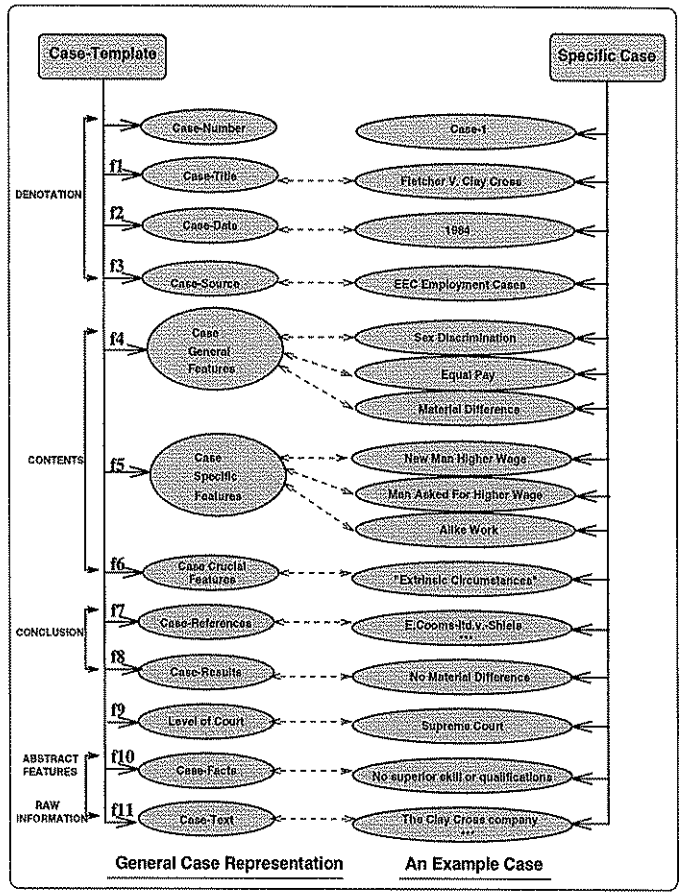

Figure 2: The Standard Case Format for LASER

3 Matching Cases

Once the cases have been represented it is necessary to decide how we are to match cases so that we can produce those which are most relevant. The matching process needs to evaluate the similarity of cases for two purposes:

1. To select from the case base those cases which are relevant to the case under consideration;

2. To order those cases so that the user can have an idea of the relative importance of the cases.

To fulfil objective 2, we must attach a numerical measure of similarity between two cases. If we achieve this, we can fulfil objective 1 by applying a threshold to this measure: if desired this threshold can be adjustable depending on whether more or fewer cases are wanted.

If we have cases represented in a number of slots (dimensions), the obvious thing to do is to construct our measure of similarity by considering

- The number of dimensions which match;

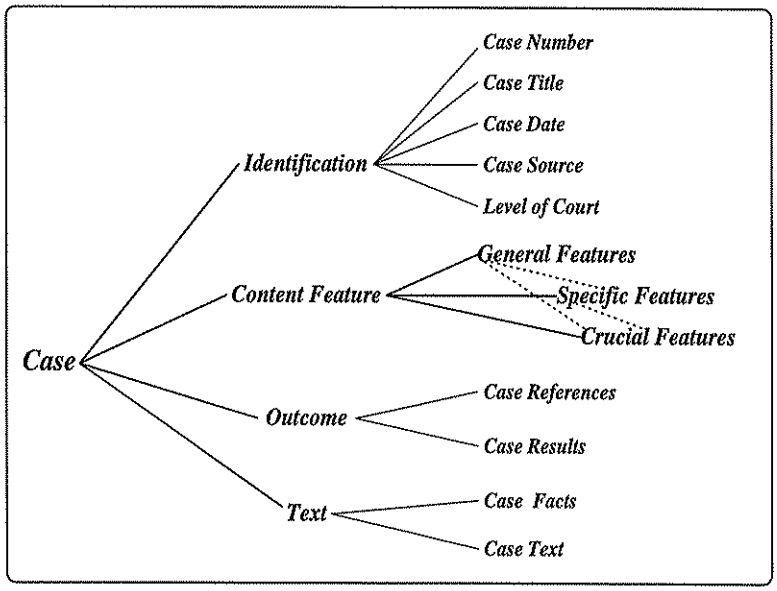

Figure 3: Features Relations in Case Structure

- The closeness of the match along these dimensions;
- The relative importance of the dimensions.

This can be produced by the use of a formula such as the following, similar to that used in MEDIATOR [7]:

$$MeanSimilarity = \frac{\sum_{i=1}^{n} W_i \times Sim(f_i^I, f_i^R)}{\sum_{i=1}^{n} W_i} \quad (1)$$

Here Wi is the weight representing the relative importance of the slot, fiI and fiR are the values of the slot i for the input and retrieved case respectively, and S is a function which gives a value for the similarity of the the values for the slot.

This method has some obvious advantages: it is straightforward to compute; it achieves what is required; it considers the features that we wish to take into account. There are, however, a number of drawbacks as well. First, the measure does not indicate why the match was considered good: such details are lost in the aggregation. Second, the weights will always be to some extent arbitrary, and may even vary according to other features of the case. This second point is important, but it can be addressed, at least in part by, for example, excluding cases which fail on some important feature or features [7], or by having different sets of weights which can be chosen on the basis of some features of the cases, as in RE-MIND[4]. On balance, it was felt that the advantages were sufficiently strong that the numerical approach should be adopted.

3.1 Similarity of Slots

A crucial decision relating to the matching concerns the nature of the similarity function. If the slots are boolean, the function can simply return 1 or 0 depending on whether the

contents of the slot are identical or not. This can also be used for slots which take a value from a small enumerated set of values. If the slot has a numerical value, similarity is represented by some function of the difference between the two values, normalised to account for the range of values that the slot can take.

The crucial slots in LASER are, however, somewhat different in nature. If we look at the fillers of the *contents* group of slots. we see that they describe issues, and as such are not selected from a small enumerated set. Moreover, issues may be closely related without being identical. Issues can be formed into an abstraction hierarchy and as such their similarity can be computed on the basis of the *most specific common abstraction* [8]. To illustrate this consider the taxonomy of animals in Figure 4.

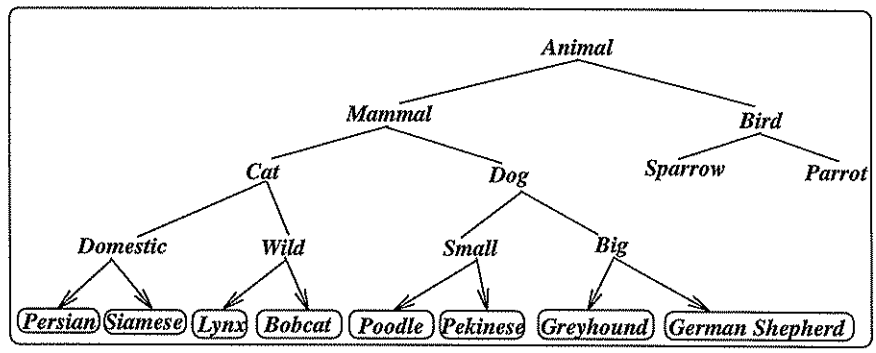

Figure 4: Simple Abstraction Hierarchy for Animal Taxonomy

A siamese is not very similar to a sparrow, but it is more similar to a sparrow than it is to a house, since both are animals. It is more similar to a poodle than to a sparrow, and more similar still to a lynx, although less similar to a lynx than is a bobcat. If we can form such an abstraction hierarchy, we can apply a metric, based on the number of abstraction levels through which it is necessary to pass in order to arrive at a common abstraction. Of course, this metric is not absolute: since different abstraction hierarchies could be produced, and there is not a definitive number of levels. Provided the abstraction steps are reasonably consistent along each branch, however, the gains from using this metric outweigh the elements of subjectivity. Part of an abstraction hierarchy of issues in the employment law domain is shown in Figure 5. The use of this hierarchy is in conjunction with a thesaurus which records synonyms for issues which may have been used in representing the cases. This thesaurus is small and specific to employment law, and also obviates the need to incorporate techniques such as stemming to avoid missing matches.

A second peculiarity of the LASER representation is that the slots can filled with a list of issues rather that a single value. Here we wish to capture both that cases are more similar the more issues they share, and that they are less similar if there are unshared issues. To reflect this we sum the individual similarities of the issues which do match for the slot and divide the result by the number of issues in the two slots taken together.

A third important feature of the LASER case representation is that the issues that appear in the contents slots may appear in *any* of these slots. For example an issue which was a *crucial feature* of one case may be a *specific feature* of another case. This allocation of issues to differing slots may occur because of the somewhat subjective process involved in representing cases, but equally it may reflect a genuine difference of emphasis in the

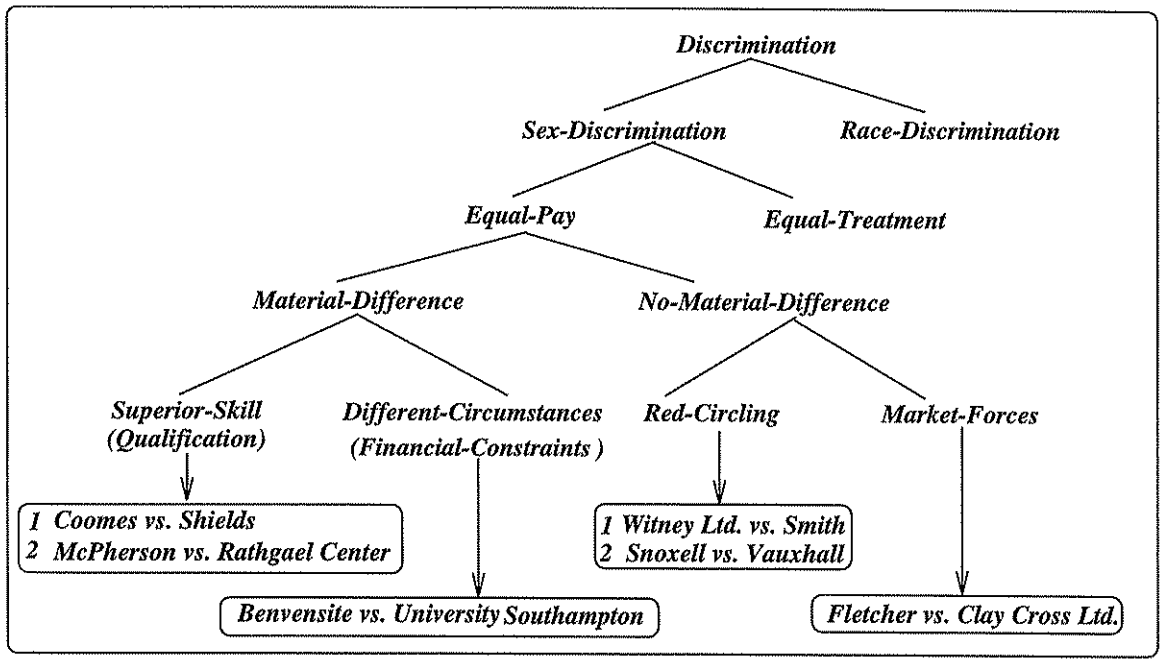

Figure 5: Abstraction Hierarchy of Employment Law Cases

two cases. In either event, it is important that the fact that the issue was allocated to different slots does not preclude the possibility of retrieving the case. For this reason it is necessary to allow matching not only between corresponding slots, but between slots of a different type also, which we term *cross-structural matching*. Not all pairs are sensible, however, and a choice must be made of which slots we wish to match a given a slot with. The cross-structural matches in LASER are shown in Figure 6.

As can be seen from this Figure, cross-structural matching takes place largely between features from the same group - for example, the *contents* group is fully connected; or between some specific feature and the more general feature from which it derives - for example, all the *contents* slots are cross-structurally matched with the case text. Again weights are attached to these matches, and the treatment is the same as the corresponding matches, except, of course, there are more potential matches to consider.

4 Weighting Features

Now that we have identified the features that we wish to compare, and suggested how we shall measure their similarity, it is necessary to consider what weights we shall attach to these comparisons to express their relative significance. It is clear that significances will differ. Some features, such as *date* and *level of court* are important when it comes to applying the case, but are unlikely to be good predictors of relevance. Others, such as *crucial features* we would expect to be of great significance.

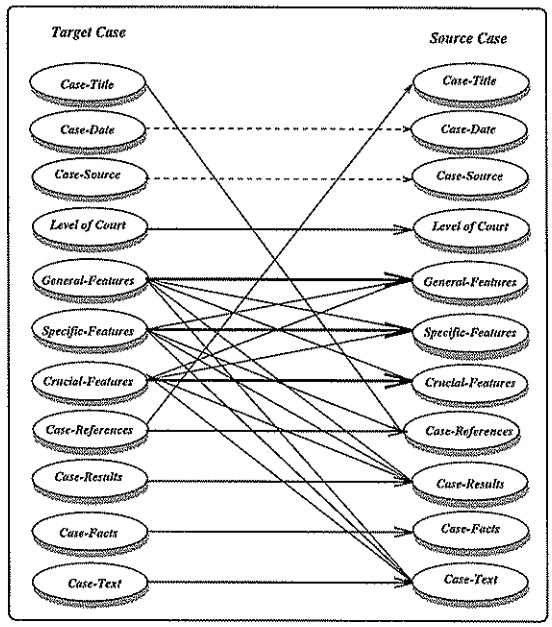

Figure 6: An Example of Cross-Structural Feature Match

4.1 General Considerations

One approach is simply to assign weights on the basis of judgement, of the system builder, an expert, or perhaps several experts. In some cases this may be enough: the relative importance of the comparisons may seem obvious. If we do this we shall probably use fairly coarse measures for the weights, as for example in MEDIATOR,[7], where only five weights we used, 1, 0.8, 0.4, 0.2 and 0, representing very high, high, moderate, low and no importance respectively. To attempt to be more precise than this would seem unnecessary, whereas we could expect experts to be able to supply information at this granularity. This method is not, however, entirely satisfactory; it requires validation against some set of cases with known results, and even if good results are achieved, we cannot be sure that this not in despite of some misjudged weights.

For this reason a better founded approach to weight assignment is required. One approach is to use a statistical analysis on some set of known cases, to see which features are the best predictors of relevance. The general idea is that good predictors will be present in relevant cases and absent from irrelevant ones. An excellent example of this sort of analysis can be found in [6] in the domain of so called "smart money" (damages paid for intangible factors such as suffering and diminution of the pleasures of life). This is of particular interest since there exists a formula for calculating smart money, and this should represent the most relevant features of the case. The analysis showed, however, that this was not so; in fact the formula is often deviated from, and so the importance of features is not what would be expected from the formula. This clearly shows the dangers of relying on expert judgement. Drawbacks of this method include the choice of cases to put in the test set, and the analytical effort involved for a large case base.

Another approach is to train the system, so that feedback from the user determines the relative importance of the various connections. A very good example of this is the SCALIR

system[13]. Essentially this system uses feedback on relevance from the user to adjust its weights. In this case no training set is used: the weights are adjusted in the process of the operation of the system. An alternative would, of course, be to train the system on a selected set of cases, and fix them, so that the users would not provide feedback as the system operates. This may be better for two reasons: the user is not burdened with the requirement to supply feedback, and there is no danger of the weights being distorted by eccentric users.

A final issue which needs to be considered is whether a single set of weights is adequate for all cases. In MEDIATOR [7], which attempts to determine the appropriate salary for baseball players, for example, the *batting average* is of crucial importance, unless the player concerned is a pitcher, in which case it is of little or no significance. It is possible to include, as in MEDIATOR several different sets of weights, the one to be used being determined by some key features of the case being matched. In LASER we decided that this would be over sophisticated: the domain does not exhibit the kind of disparity found in the above example.

4.2 Weighting in LASER

In LASER we adopted a supervised learning approach to the allocation of weights. Weights were originally all set at 0.5, and a training set was used to determine how the weights should be adjusted; the weight being increased for comparisons that found relevant cases and decreased for those that identified negative cases [?]. Of course, an opinion of an expert was still required to categorise the cases in the training set as positive and negative.

The training algorithm is a method analogous to the training cycle for neural nets. A set of 'training' target cases must be selected and, for each of these, two sets of source cases have to be determined: one set representing what are considered to be *relevant matches* to the target case, the other representing *irrelevant matches* as determined by a domain expert. The training proceeds, as shown in Figure 7, by the iterative selection of a target case, one corresponding relevant source case (i.e. positive case) and one corresponding irrelevant source case (i.e. negative case).

Weights can be reinforced or weakened depending on whether or not they support the preference of retrieval for the relevant source case rather than the irrelevant source case. Tables 1 and 2 show the result of training LASER in this way. The results shown in Table 2, are obtain from three diffrent 65 trial training combinations.

Whilst these tables serve mainly to confirm what might be seen as intuitive, they are interesting both in the number of features shown to be relevant in these case, and the equality of contribution that these relevant features make. This helps the retrieval process, since it makes LASER robust against isolated counter-examples where a generally important feature retrieves a negative case. In such an instance the effect is more than counterbalanced by other features.

5 Interaction With the System

The final design choice that we wish to discuss here relates to how the system will be used. Standard boolean keyword in context systems tend to be rather interactive: once

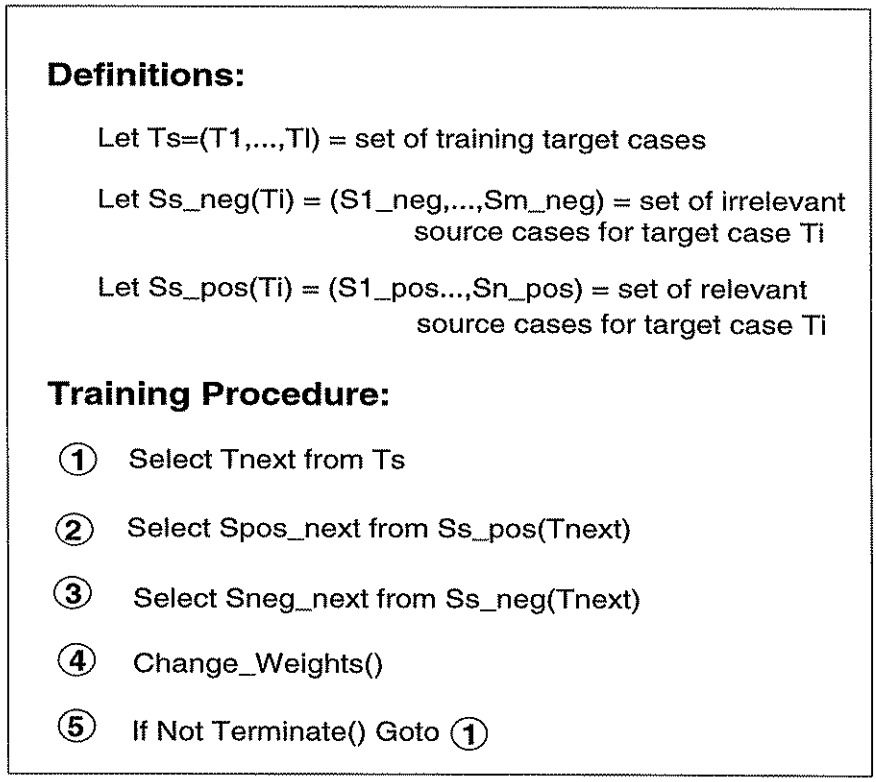

Figure 7: Training the Correspondence Network

an initial query has been issued, the user will be invited to refine the query, by adding a conjunct if too many items have been found, or a disjunct if too few items have been found. Alternatively some systems will proceed without user intervention, giving a ranked list of retrieved items on the basis of a query or a description of the source case.

The design of LASER is such that the second is a natural and feasible way to use the system. In addition, however, we give the user the power to stipulate certain values for certain features, so as to constrain the search and focus the results. As well as providing for a more flexible approach, with the user having the ability to concentrate on particular aspects of the case in hand, or, for example to exclude cases decided prior to a significant amendment in the law, it can help to overcome any difficulties that might arise from using a single set of weights.

6 Conclusion

In this paper we have discussed some of the crucial design issues associated with the retrieval of cases in a legal information system. Our discussion has been informed by experience in implementing a system in a particular domain, namely UK employment law. This implementation gives some pragmatic justification to our particular choices.

No.	Feature Pairs	Positive	Negative	No.	Feature Pairs	Positive	Negative
1	general-to-general	84	74	11	*T-specific-S-result*	2	0
2	*general-to-specific*	7	1	12	*T-crucial-S-result*	1	0
3	*general-to-crucial*	2	0	13	*T-general-S-text*	9	2
4	*specific-to-general*	3	0	14	*T-specific-S-text*	6	1
5	specific-to-specific	25	2	15	*T-crucial-S-text*	3	0
6	*specific-to-crucial*	2	0	16	*T-result-S-text*	2	0
7	*crucial-to-general*	1	0	17	*T-title-S-reference*	13	0
8	*crucial-to-specific*	2	0	18	*T-reference-S-title*	8	1
9	crucial-to-crucial	8	0	19	T-S-result	3	0
10	*T-general-S-result*	6	1	20	T-S-level-of-court	17	9

Table 1: Statistical Analysis of Feature Pairs Shared by Twenty Target Cases and Their Related Positive and Negative Source Cases

No.	Cross-Structural Feature Pairs	Initial Weights	1st	2nd	3rd	Average Weight	Training Effect
1	*general-to-specific*	0.5	0.8	0.9	0.76	0.82	*increases*
2	*general-to-crucial*	0.5	0.3	0.0	0.0	0.1	decreases
3	*specific-to-general*	0.5	0.65	0.9	0.85	0.8	*increases*
4	*specific-to-crucial*	0.5	0.0	0.3	0.3	0.2	decreases
5	*crucial-to-general*	0.5	0.1	0.2	0.3	0.2	decreases
6	*crucial-to-specific*	0.5	0.1	0.55	0.55	0.4	decreases
7	*T-general-S-result*	0.5	0.7	0.55	0.65	0.63	*increases*
8	*T-specific-S-result*	0.5	0.1	0.15	0.35	0.2	decreases
9	*T-crucial-S-result*	0.5	0.0	0.2	0.1	0.1	decreases
10	*T-general-S-text*	0.5	0.65	0.85	0.75	0.75	*increases*
11	*T-specific-S-text*	0.5	0.55	0.7	0.7	0.65	*increases*
12	*T-crucial-S-text*	0.5	0.4	0.4	0.75	0.52	*increases*
13	*T-result-S-text*	0.5	0.25	0.3	0.2	0.25	decreases
14	*T-title-S-reference*	0.5	0.97	0.65	0.75	0.79	*increases*
15	*T-reference-S-title*	0.5	0.5	0.6	0.55	0.55	*increases*

Table 2: Experimental Results of Weight Improvement in Cross-Structural Feature Pairs

7 Aknowlegment

This research was financially supported by the Ministry of Culture and Higher Education of Iran.

References

[1] K.D. Ashley. *Modeling Legal Argument: Reasoning with Cases and Hypotheticals.* Cambridge, MA: MIT Press, Bradford Books, 1990.

[2] Trevor Bench-Capon and Marek Sergot. Towards a Rule Based Representation of Open Texture in Law In Charles Walter (ed) *Computing Power and Legal Reasoning*, Greenwood Press, 1989, pp 39-60.

[3] John Bing. Designing Text Retrieval Systems for Conceptual Searching. In *Proceeding of the First International Conference on Artificial Intelligence and Law*, Boston, 1987. New York: ACM Press.

[4] Cognitive Systems, Inc. *Re-Mind Developer's Reference Manual*, 1992.

[5] J. P. Dick. A Conceptual, Case-Relation Representation of Text for Intelligent Retrieval. Technical Report CSRI-265, University of Toronto, July 1992.

[6] Cees Groendijk and Maaike Tragter Statistical and neural approaches to smart money determination in J.C. hage et al (eds) *Lecgal Knowledge Based Systems: Telecommunications and AI and Law*, Koninklijke Vermande, Lelystad, 1995.

[7] J.L. Kolodner and R.L. Simpson. The MEDIATOR: Analysis of an Early Case-Based Problem Solver. *Cognitive Science*, 13(4):507–549, 1989.

[8] J.L. Kolodner. *Case-Based Reasoning* . Morgan Kaufmann Publishers, Inc., 1993.

[9] Andrzej Kowalski. Case-Based Reasoning and the Deep Structure Approach to Knowledge Representation. In *Proceedings of the Third International Conference on AI and Law*, pages 21–30, St. Catherine's College Oxford, England, June 25-28 1993. ACM Press.

[10] Mohammad Ali Montazeri, Alison E. Adam, and Mike Brown. A Method for Training Standardized Case Base to Provide Robust Retrieval. In *Proceedings of Expert Systems 94, the Fourteenth Annual Technical Conference of the British Computer Society Specialist Group on Expert Systems, Bramer, M.A. and Macintosh, A.L. (eds.)*, pages 133–144, Cambridge University, December 1994.

[11] Rissland, Edwina L., Skalak, David B., and Friedman, M. Timur, BankXX: A Program to Generate Argument Through Case Based Search, in (*Proceedings of the Fourth International Conference on AI and Law*, Amsterdam, ACM Press, pp 117-124, 1993.

[12] Edwina L. Rissland and M. Timur Friedman Detecting Change in Legal Concepts in (*Proceedings of the Fifth International Conference on AI and Law*, University of Maryland, ACM Press, pp 127-136, 1995.

[13] Daniel E. Rose *A Symbolic and Connectionist Approach to Legal Information Retrieval* Hillsdale, New Jersey: Lawrence Erlbaum, 1994.

[14] J.C. Smith, D. Gelbart, K. MacCrimmon, B. Atherton, J. McClean, M. Shinehoft, and L. Quintana. Artificial Intelligence and Legal Discourse: The Flexlaw Legal Text Management System. *Artificial Intelligence and Law*, 3(1-2):55–95, 1995.

[15] D.B. Skalak and E.L. Rissland. Arguments and Cases: An Inevitable Intertwining Artificial Intelligence and Law. *Artificial Intelligence and Law*, 1(1):3–44, 1992.