12th BILETA Conference
The Future of Legal Education and Practice

Monday, March 24th & Tuesday, March 25th, 1997
Collingwood College, University of Durham

Using Hypertext and Parallel Processing to Integrate a European Law Database

Philip O'Shea and Eve Wilson, University of Kent

1. The CELEX Database

The CELEX database is the official legal database of the European Community. It is divided into eight sectors as detailed in table 1(Nunn-Price, 1992).

Table 1: The sectors of the CELEX database (sector 8 does not exist).

Each document in the database is divided into named fields, which include:

- The title of the document.
- The text of the document itself, which for the case reports of sector 6 is divided systematically into several named fields.
- Fields which summarise the document content by giving keywords to be used as indexing terms. Different fields, with different sets of keywords, may apply to the different sectors.
- Fields which contain a single item of information, for example giving the legal form of the document (whether it is a regulation, decision, etc).
- Cross-references to other documents which may be within the CELEX database, to indicate for example the legal basis of an act or which acts it affects; or external references to national measures implementing a directive.
- Dates relating to the document: there are many such fields, giving for example the date of publication, the date of debate on a resolution, the date on which an act becomes operative, and the dates on which national measures implementing a directive enter into force.

2. The DAPText Indexing Engine

DAPText is a database indexing engine from Cambridge Parallel Processing which runs on the DAP (Distributed Array Processor, a SIMD machine).

When applied to a free text database, its function is to accept a query which consists of boolean operators applied to word occurrence conditions and to return a list of documents which match the query. To do this efficiently, it maintains an inverted index, that is, it records for each word of the database vocabulary the documents and the positions within a document where that word occurs.

A database index such as DAPText typically divides a document into various fields, which are indexed separately from each other, so that it is possible to restrict a word-occurrence condition in a query to a specific field.

A disadvantage of this, where there are a large number of fields, is that the only way to apply a word-occurrence condition to all fields is to produce a query which specifies each field explicitly, combined by an 'OR' operator.

It is therefore advisable to merge together into a single DAPText field a group of CELEX fields which are much more likely to be searched together rather than separately: this was done for the fields which contain the main text of the document.

A problem encountered in using DAPText was that the number of available fields was not sufficient to cover the number of fields in a CELEX document.

This was solved by allocating more than one CELEX field to a single DAPText field: a tag (a prefix or suffix) is attached to each word to identify the CELEX field to which it belongs. A query then attaches a tag to a word-condition, and the result is in general equivalent to increasing the number of available fields.

An indexing engine typically provides a word-truncation facility, that is, the facility to specify a match of any word beginning with a given sequence of letters. When used in conjunction with word-tagging, the effect depends on whether the tag is a prefix or suffix. If a prefix, then word-truncation works as normal for each CELEX field. If a suffix, then word-truncation causes the tag to be ignored, so that all CELEX fields which are placed within a single DAPText field are matched simultaneously.

Where a field contains a numeric value, such as a date, and the field contents are indexed as a single term then the only way to use the field for retrieval is by specifying an equality condition.

An inequality condition is possible if the number is encoded in binary, that is, with one indexed term corresponding to each bit in the binary representation of the number, with the presence of the term indicating that the corresponding bit has the value '1'.

Any inequality condition can then be translated to a boolean expression.

Where a field contains a value drawn from a relatively small set of possible values, but expressed as a phrase of several words, it is natural to consider encoding the field using a single term. It is possible that doing this would save memory and, more importantly, model the semantics of the field. However, it was decided not to do this and simply to encode each word as an indexed term. This has advantages of simplicity and uniformity - it would otherwise be necessary to apply the encoding to every query of the field - and appears to be satisfactory for the CELEX data.

DAPText stores word position information only for some fields: this information is used to enable a query to specify that one word occurs adjacent to another in a document.

In general, for the above type of field which contains one of a restricted set of phrases, it is not necessary to store word position information, because it is not likely that two phrases with different meanings will contain the same two content words differing only in their relative position.

3. A Measure of the Relevance of a Word

In any document, some words (content words) are more strongly linked to the meaning of the whole document than others (function words). This section proposes a method for measuring the degree of absolute meaningfulness or relevance of a word. Later sections propose different uses for this information.

The procedure of 'relevance feedback' is a method for refining a system's initial response to a query by obtaining information from the 'term weights', which are a measure of the relevance of each term to the particular query.

The following formula for term weighting has been widely used and is due to Robertson and Sparck Jones (1976). Here it is given in terms of the number of 'postings' of a word, that is, the number of individual occurrences in the text of a document; there is an alternative form of the equation in which each word is only counted once for each document in which it occurs. The starting point is a set of documents, each of which is known to be either relevant to the present query or not relevant. Let *R* denote the set of relevant documents and s1 the set of irrelevant documents. Let s2 denote the number of postings of term *t* within *R*. Then the following estimate is used for the probability of term *t* within *R*:

See Equation 1

and s3 is given in the same way. The term weight s4 is now given by:

See Equation 2

When s4 is summed over all the words in a document, the result is a value which can be used to rank a set of documents according to their estimated relevance to the present query.

A measure for the absolute relevance of a word within a database can be found if there is an existing classification of each document into subjects: for the CELEX database, this is provided by the 'SUB' field which gives a list of standard phrases describing the subject matter of a document, drawn from a set of about 250. The absolute relevance s5 of a term is found by calculating the term weight for each standard subject *s* and taking the maximum:

See Equation 3

where is the term weight when subject is considered 'relevant'. Note that the standard subjects are not mutually exclusive.

One possible use of s5 would be as an initial term weight for a query, that is, before any relevance feedback information has been obtained.

In applying this measure to the CELEX database, a 'term' consists of a single word with a tag suffix to identify the database field to which it belongs - a different tag was used for each field considered distinct by the retrieval engine. This allows for the fact that a word in a particular field may have a specialised meaning which should be kept distinct from other occurrences of that word. For example, it is clear that the terms with the highest relevance are among those which belong to the 'SUB' field. The idea of tagging is not simply a matter for the implementation of the system: it allows the description of the relevance measure above to be phrased without taking account of the existence of database fields.

4. Document Clustering

The conventional user interface to an indexing engine requires the user to specify a query based on word-occurrence conditions. An alternative approach to retrieval is to start from a particular document and to find other documents which are similar in subject matter, sometimes called 'query by example'.

A computationally efficient method for achieving this effect is to precompute a clustering of the documents in a database, so that a query consisting of any document is answered by listing the members of the relevant document cluster.

For each document there is a vector **d** which has one element for each term in the vocabulary of terms. The value of the element corresponding to a term *t* is s6 where s5 is the absolute relevance of term *t* and s7 is the number of times that term *t* occurs in the document. A cluster of documents is represented by a vector **c** which is the sum of **d** for each document in the cluster. The degree of similarity between a document **d** and a cluster **c** is measured by the cosine correlation (see for example Stanfill et al. 1989; Salton and Lesk 1971):

See Equation 4

There are many standard clustering algorithms which are not practicable for a database the size of CELEX on a single-processor machine. Willett and Rasmussen (1990) report experiments with different clustering algorithms using a DAP - they use a simpler distance measure, the unweighted squared Euclidean distance.

The clustering algorithm used here is based on the simplest single-pass algorithm, in which each document in turn is added to the nearest existing cluster, or used to initiate a new cluster if its distance from all clusters exceeds a certain threshold.

This algorithm is applied a number of times, first to divide the data into relatively large clusters, and then recursively to break these down into smaller clusters.

5. Retrieval and Ranking of Similar Documents

This section describes a method for using a boolean query engine to retrieve a set of documents similar to a given one, and at the same time to provide a partial ranking of the retrieved documents in order of their estimated similarity.

This involves generating a set of queries from the submitted document, each of which is executed by the query engine to provide a part of the result. These queries produce mutually exclusive sets of documents. The queries are ranked and executed in order, to give the ranking of the result list.

Let the terms which occur in the submitted document, sorted in order of absolute relevance using the measure described in section ??, be s8 where s9 is the most relevant term, and no duplicates occur in this list.

A 'sub-query' of length is the boolean '*and*' of *n* terms, where the *i*'th term is either s9 or not(s9). So an example of a sub-query of length 3 would be s9 *and not* (s10) *and* s11.

The ranking of sub-queries is defined by stating that where *S* is a sub-query of length *n*, then the sub-query *S and s12* has a higher rank than *S and not*(s12).

The central operation is a recursive procedure which takes an argument which is a sub-query S (See Figure 1).

The initial argument is a sub-query of length zero, which has the boolean value 'true' (so that it matches all documents in the database). This procedure has the properties that all executed queries whose results are output are executed in their ranked order; the procedure terminates naturally when every document in the database has been listed exactly once; and that the document against which similarity is being measured is listed as a result of the first executed query.

See Figure 1: The algorithm for retrieving and ranking similar documents

A sub-query with more terms takes a longer time to execute, and so the execution time becomes large when the procedure goes through many levels of recursion.

To limit the execution time, the procedure was terminated after either a certain amount of CPU time has been used or a certain number of document identifiers have been output.

To prevent the possibility of no results being produced, when the time limit is reached the current query is executed before the procedure is terminated.

It is not sufficient to limit the depth of recursion instead of limiting the execution time, because a query of many terms is sometimes required in order to establish a ranking of a group of similar documents - if this is a large group at the top of the result list then it is worth the effort to provide a ranking, at the price of reducing the total number of documents listed.

The ranking of sub-queries used here is not the best possible: a better way would be to sum the relevance values for all terms which occur unnegated in the sub-query, and to order the sub-queries by the resulting value, as in (Radecki, 1982). However, the present algorithm could not then be applied.

6. Results

It is not clear how the results can be objectively evaluated, however, in comparing the output from the clustering process and the ranking algorithm as implementations of query-by-example it is apparent that the ranking algorithm gives results which are far superior.

The clustering process frequently combines in a single cluster two documents which have little apparent connection (it is more difficult to assess the extent to which connected documents are placed in separate clusters).

The fact that a document has in general more than one area of significance means that there is probably no single division into clusters which would be entirely satisfactory, so that using a better clustering algorithm would give only a limited improvement.

7. References

N. Nunn-Price (1992) The Celex Database: A guide to European Community Law. Context Limited, London.
T. Radecki (1982) Incorporation of Relevance Feedback into Boolean Retrieval Systems.in *Research and Development in Information Retrieval: the Proceedings of the Fifth SIGIR/ACM conference,* Berlin, 1982, Springer-Verlag Lecture Notes in Computer Science **146**, ed. G. Salton and H.-J. Schneider, pp. 133--150
S. E. Robertson and K. Sparck Jones (1976) Relevance weighting of search terms. *Journal of the American Society for Information Science* 1976 **27** pp. 129--146
G. Salton and M. E. Lesk (1971) Computer Evaluation of Indexing and Text Processing. Chapter 7 of *The SMART Retrieval System: Experiments in Automatic Document Processing*, ed. G. Salton. Prentice-Hall, New Jersey.
C. Stanfill, R. Thau and D. Waltz (1989) A Parallel Indexed Algorithm for Information Retrieval. *Proceedings of the Twelfth ACM/SIGIR Conference,* Cambridge, Massachusetts, pp. 88--97
P. Willett and E. M. Rasmussen (1990) Parallel Database Processing: Text Retrieval and Cluster Analysis using the DAP. Pitman, London.