Some uses of Fuzzy logic in multimedia databases querying

Didier Dubois, Henri Prade, Florence Sèdes

Institut de Recherche en Informatique de Toulouse (IRIT) – CNRS
Université Paul Sabatier
118 route de Narbonne
31062 Toulouse Cedex 4 - FRANCE
{dubois, prade, sedes}@irit.fr

Abstract. Fuzzy set methods have been already applied to the representation of flexible queries in databases systems as well as in information retrieval. This methodology seems to be even more promising in multimedia databases which have a complex structure and from which documents have to be retrieved and selected not only from their contents but also from their appearance, as specified by the user. This paper provides a preliminary investigation of two potential applications of fuzzy logic in multimedia databases querying. The first one concerns the detection of modifications in semi-structured documents and relies on a graph matching procedure where subparts of the graph describing the structures have different levels of importance. The second illustrates another aspect of the fuzzy logic methodology allowing for a querying by examples process. Examples and counter-examples are supposed to be provided by the user with their levels of representativity; attributes used in their descriptions can be also weighted according to their level of importance.

1. Introduction

The full use of multimedia information systems raises new issues, especially for the access and the manipulation of information in a more intuitive, less formalised and human-friendlier way. It poses new challenges from data representation to querying and retrieval. Due to the richness of multimedia data content, querying systems must have extended capabilities: providing high-level abstractions in order to model multimedia data and their presentation, querying them on their appearance as well as by their content, querying by example and allowing for flexible queries.

It is well-known that fuzzy logic provides a framework for modeling flexibility and handling vagueness in the interface between human conceptual categories and data. Such capabilities have been already developed in the database field, specially for handling flexible queries [BP92, BP95, BDPP97, BLP98]. There has been only a few preliminary and specialized papers on the use of fuzzy logic in multimedia databases systems [BCR98, CR97, C98]. In this paper, we take the example of two potential applications of different nature and which illustrate two important aspects of multimedia querying: i) retrieving documents having similar structures, and ii) looking for documents having their contents similar to examples (and may be dissimilar to counter-examples).

In the next section, we restate multimedia databases specificities and features, versus "classical" (relational) databases requirements. We discuss what functions, traditionally allocated to DBMSs, still make sense in a multimedia environment, what new ones would be useful, specially investigating the querying step. Then, in Section 3, we summarise the main contributions of the fuzzy approach to database systems querying. In Section 4, we focus on detecting modifications in semi-structured data or document. This is important in databases area, for active databases, data warehousing, view maintenance, and version and configuration management. Most previous works in change management have dealt with flat-file and relational data, but hierarchically structured data must be also handled. We examine how approximate matching techniques make it actually possible to improve such systems capabilities. In Section 5, an approach is proposed for evaluating to what extent a document can be considered as a part of the answer to a query expressed in terms of examples and counter-examples.

2. Multimedia databases

A multimedia DBMS must address the following requirements [AN97]: traditional DBMS capabilities, huge capacity storage management, media composition, integration and presentation, query support and interactivity. It must thus cope with data modelling, object storage, indexing, retrieval and browsing. We can view a multimedia database as a controlled collection of multimedia data items, such as text, images, graphic objects, sketches, video and audio, which can be related by temporal and/or spatial constraints. The multimedia DBMS should accommodate these special requirements by providing high-level abstractions in order to manage the different data types, along with a suitable interface for their presentation.

Audio data, for example, can be automatically indexed by signal analysis or speech recognition followed by keyword-based indexing. Images cannot: shape, colour, texture, identified through pattern recognition, are not sufficient, this is why textual descriptions usually refers to them. Thus, representing multimedia information such as pictures or image sequences raises some problems for their retrieval due to the limitations of textual descriptions usually associated to it, versus the massive information available according to the context, interpretation, user's profile, etc. The lack of standard structure and the variety of potential available information means that it can be difficult to express precise queries.

Thus, characteristic features of multimedia data are their lack of structured information, multiplicity of data types and heterogeneity of formats, spatial and temporal characteristics, and inadequacy of textual descriptions.

Queries in relational systems are exact match queries: the system is able to return exactly those tuples a user is precisely asking for and nothing more. To specify the resulting relation, conditions concerning known attributes of known relations can be formulated.

From the Information Retrieval area, we know how difficult it is to characterise the contents of textual objects [FBY92]. Problems are encountered on the one hand in specifying the contents of the objects, and on the other hand, for describing the objects one is looking for. In multimedia databases, the question is thus how to characterise the 'contents' of image, audio or video data. Indeed, we must face the problem of giving an interpretation to a photo or a song, for which many interpretations exist in general, according to the context, the user's point of view [Ape97].

Multimedia data must be preferably interpreted before they can be queried, in order to generate content descriptions (otherwise it might be more costly to make it on line). Multimedia information can be retrieved using identifiers, attributes, keywords,... Note that indexing is context-dependent. Introducing abstractions allows the user to refer to the data in terms of high level features or metadata, which constitute his model of the application domain.

Extensions of conventional concepts of query languages -all the retrieved objects exactly match to the query- require approaches that can deal with the temporal and spatial semantics of multimedia data, or query languages that can incorporate flexibility in the expression of requests. Indeed, queries are usually imprecise, so, relevance feedback and meaning similarity, rather than exact matching, and mechanisms for displaying ranked results are important. This is particularly important in combination with content-based access, where the user's criteria are often approximately and imprecisely formulated. Moreover, multimedia querying should offer support for new requirements such as querying by examples, querying spatial or temporal data, flexible querying using fuzzy predicates.

In this context, the concept of document generally refers to any composite object combining elements characterised by different media. Making the most open form of querying possible is important to allow queries to refer to the document appearance as well as its content. So, the fuzzy set techniques referred to in the next section can be applied in multimedia data querying.

The next section deals with flexible queries to a "classical" database, that can also work on an abstract representation (metadata) of the objects, via indexes for instance.

3. Fuzzy data bases

Research on "Fuzzy databases" (see [BK95, Pet96]) has been developed for about twenty years. These works have been mainly concentrating on flexible querying in classical languages (e.g., [KZ86, BP95]), and in data representation and object-oriented languages [DeC97, C98]. An introduction to these different issues may be found in a recent survey by Bosc and Prade [BP97].

The flexibility of a query reflects the preferences of the end-user. Using a fuzzy set representation, for expressing a flexible selection criterion, the extent to which an object described in the database satisfies a request then becomes

a matter of degree. In this case, the end-user provides set of attribute values which are fully acceptable for him as well as set of values which are clearly not acceptable, the remaining values being rank-ordered according to their level of acceptability. Moreover, a query may also allow for some similarity-based tolerance: close values are often perceived as similar, interchangeable. Indeed, if for instance an attribute value *v* satisfies an elementary requirement, a value "close" to *v* should still somewhat satisfy the requirement. The introduction of a tolerance relation can make flexible a non-fuzzy query as in the example: “find people less than 40 year old” and where a 41 year old person can be retrieved with an intermediary degree of matching based on the (context-dependent) appraisal of the proximity between 40 and 41.

An advantage of fuzzy set-based modelling, is that it is mainly qualitative in nature. Indeed in many cases, it is enough to use an ordinal scale for the membership degrees (e.g., a finite linearly scale). This also facilitates the elicitation of (context-dependent) membership functions, for which it is enough in practice to identify the elements which totally belong and those which do not belong at all to the fuzzy set.

Fuzzy set membership functions [Zad65] are convenient tools for modelling user's preference profiles and the large panoply of fuzzy set connectives can capture the different user attitudes concerning the way the different criteria present in his/her query compensate or not; see [BP92] for a unified presentation in the fuzzy set framework of the existing proposals for handling flexible queries.

Thus, the interest of fuzzy queries for a user are twofold:

i) A better representation of his/her preferences. For instance, "he/she is looking for an apartment which is not too expensive and not too far from downtown". In such a case, there does not exist a definite threshold for which the price becomes suddenly too high, but rather we have to differentiate between prices which are perfectly acceptable for the user, and other prices, somewhat higher, which are still more or less acceptable (especially if the apartment is close to downtown). Obviously, the meaning of vague predicate expressions like "not too expensive" is context/user dependent, rather than universal.

The large panoply of fuzzy set connectives can capture the different user's attitude concerning the way the different criteria present in his/her query compensate or not. Moreover in a given query, some part of the request may be less important to fulfil; this leads to the need for weighted connectives.

ii) Fuzzy queries, by expressing user's preferences, provide the necessary information in order to rank-order the answers contained in the database according to the degree to which they satisfy the query. It contributes to avoid empty sets of answers when the queries are too restrictive, as well as large sets of answers without any ordering when queries are too permissive.

4. Looking for similar semi-structured documents

In the context of documentary applications, one of the main differences between a multimedia database and an ordinary one from a querying point of view, is that in the first case the request may refer to a document in terms of its appearance and not only in terms of its information contents. The lack of standardised structure of the document(s) to be retrieved calls for the use of flexible queries.

Indeed, an important class of multimedia documents are semi-structured documents, like HTML or XML documents [CS98]. The idea of allowing for flexibility in the exploitation of semi-structured information [Abi97] consists here to refer to the structure in an approximate matching procedure.

For example, a request may refer to the description of a document structure (e.g. title, author(s), probably an abstract, maybe key-words, a body structured in sections and sub-sections, certainly followed by references, among which we preferably may find some author’s name...). Thus, a first level of querying is to retrieve an already seen document where the description refers to its structure [DS96] in a flexible way.

In the following, the problem of detecting modifications in semi-structured data or document is addressed. This problem is challenging due to the irregularity, incompleteness and lack of schema that characterise semi-structured data.

Structural matching and discovery in document such as HTML are useful for data warehousing, version management, hypertext authoring, digital libraries and web databases, for instance to know modifications in an HTML document or to find common substructures. Indeed, in addition to offering access to large collection of heterogeneous and semi-structured data (e.g. HTML documents), the Web allows this information to be updated at any time. These rapid and often unpredictable changes create a problem of detecting these modifications. A user may visit documents repeatedly and is interested in knowing how each document has changed since the last visit. This may be achieved by saving a snapshot of the previous HTML pages at the site (something that most browsers are

able to do anyway). Assuming we have saved the old version of the document, we want to periodically detect the changes due to their evolutions, by comparing the old and new versions of the document and knowing how much similar they are.

Most previous work in change management has dealt only with text files [Mye86, Kif95], flat-files or relational data (for example, [LGM95] presents algorithms for efficiently comparing sets of records that have keys). A system, called *LaDiff*, has been implemented to detect, mark, and display changes in structured documents. It is based on their hierarchical structure analysis, taking two versions of a Latex document as input and producing as output a Latex document with the changes marked [PGMW95].

Much database research has been conducted about structured documents such as SGML [CACS94, CLS94, MZ98] : given a Document Type Definition (DTD), any document will conform to it. This DTD is, like a database schema, a generic structure for a class of documents. Few systems have been built to support semi-structured data. Indeed, these data are irregular, incomplete and does not necessarily conform to a fixed schema. Semi-structured data may have some structure but, if it exists, it is enclosed into the instance and a priori unknown. It must be elicited, in order to discover patterns. From a given mark-up language, a parser can analyse the document content in order to elicit the structure items. Documents are represented as ordered labelled trees, since hierarchically structured information can be represented as trees, in which the children of each node have a designated order (see *Fig. 1*).

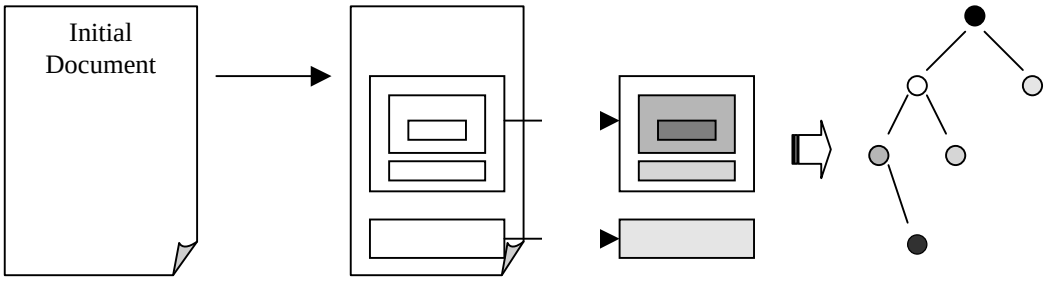

Fig. 1. Extracting the document structure

We must identify changes not just to the "nodes'" in the data, but also to their relationships. For example, if a node (and its children) is moved from one location to another, or if a leaf node has been split up into its children, we would like to detect it (e.g., given a level *l*, the structure has been reorganised, directly attaching the children at level *l+1*, to the father at level *l-1* (see *Fig. 2*)).

Hence, one of our problems is to find an appropriate matching for the trees we are comparing. In some application domains, the problem is easy, such as when data objects contain object identifiers or unique keys. In other domains, such as structured documents, the matching is based on labels and values only, so it is more difficult. Two documents can be compared using tree matching techniques [SZ90, ZS89, WZJS94]. Furthermore, not only do we want to match trees that are identical (with respect to the labels and values of the nodes and their children), but we also want to match trees that are "approximately equal", according to their similarity. Graph structure is not adapted to comparison other than exact identity. In the following, we suggest a process adapted to comparison algorithms [SZ90, ZS89], by successive adaptation, for example deleting a structure level if this one is considered as not much relevant. One of the main characteristics of this model is that it saves the ordering between elements. Without it, the problem of "subtree" relations between two trees cannot be handled [Kil92].

The possible cohabitation of multiple views of the same document handicaps any approach based on strict comparison. An additional semantic knowledge about elements (nodes) or relationships (arcs) is necessary to allow an approximate comparison. This knowledge can be extracted from the element name (mark-up content), from the attributes or their values, or from any external user's specification. Our approach to the problem of detecting modifications in pieces of hierarchically structured information has two main original features:

- *Semi-structured data.* Although some database systems, particularly active database systems, build change detection facilities into the system itself, we focus on the problem of detecting changes given old and new versions of the data. What can be known as a structure of a semi-structured document should be extracted from the encoding of the document itself.
- *Approximate matching.* By approximate matching of graph, we mean that the graphs at least match for their essential parts and if possible, for their less important subpart. Then, the matching becomes a matter of level.

A first version of the elicitation process of the structure (without any grading of the importance of each subpart of the document) has been implemented in the rewriting tool called eXrep [LQC95]. The ordered tree-like structure of the document is built by recognising structure items from the content itself, through automatic identification of marks and their rewriting by means of dictionaries. From this rewriting process, it is possible to identify in the tree eddges which can be considered as more important than others: from the document analysis, a syntactic and semantic marking can allow to infer that some edges have a lower weight than other ones. The ideas at the basis of the weights' computation are that the level of importance of the edge depends on the recognised mark, its content, the text associated with the pending node, and the depth of the tree representing the structure. Indeed, comparison cannot rely on structural similarities while content is ignored. The weighting process first deals with structural comparison and mark-up semantics, second with the content. In practice, one could only distinguish between a rather small number of levels in the importance scale.

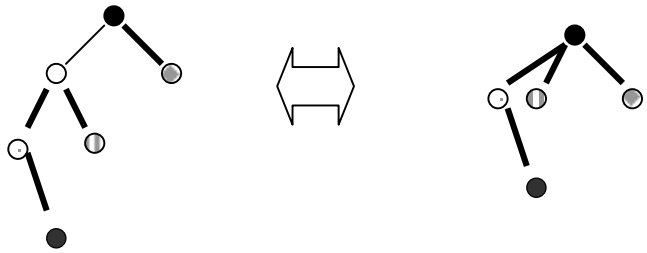

Fig. 2. Comparing two structures

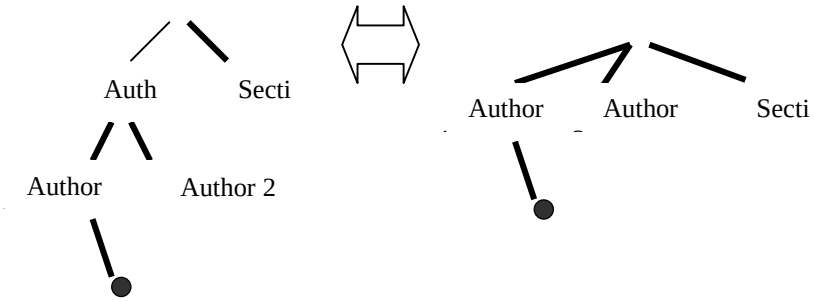

Fig. 3. Example of similar structures

Given two tree structures T^1 and T^2 whose nodes are weighted on a scale

$w_1 = 1 > w_2 > ... > w_n > w_{n+1} = 0$

their similarity can be evaluated in the following way. Let T_i be the tree built from T by only keeping edges whose level is greater or equal to w_i and fusing the nodes belonging to a suppressed edge.

Then

similarity(T^1, T^2) = w_{n-i+1}

where the function $i \rightarrow n-i+1$ is the order-reversing map of the scale {1,…,n}
if T^1_i and T^2_i are identical but T^1_{i+1} and T^2_{i+1} are different,
and similarity(T^1, T^2) = 0 if T^1_1 and T^2_1 are different.

So

similarity(T^1, T^2) = 1

if T^1 and T^2 are similar at each level, from 1 to n (at each level, similarity is binary, equals to 0 or 1).

What about the content ? When comparing T^1 with an approximation of T^2 where some edge has been deleted, we may have to detect whether the text associated with the suppressed node, if there is one, is pasted elsewhere in the approximation of T^2. More generally we may think in the computation of the similarity of adding requirements on identity or subsumption of the attached texts (in order to have a nonzero similarity).

Thus, given a document, it is possible to retrieve and rank-order approximately similar documents.

5. Querying by examples

The user is not always able to easily express his request even in a flexible way. First, it may be more convenient for him to express what he is looking for from examples. Second, since he may have absolutely no a priori knowledge of the amount of retrievable documents (for a given request), it may be useful if the system is able to guide him about what exists, by incrementally building queries from examples.

Querying based on examples, for eliciting user's preferences, can provide the necessary information for building a query. Thus, the user may say to what extent a few examples of documents are representative of what he is looking for by using some (finite) similarity scale. Then, relevance of current documents should be evaluated in terms of their similarity with respect to these examples and may be counter-examples. Issues are then close to fuzzy case-based reasoning [DEGGLP98].

Examples as well as counter-examples are described in terms of precisely known attribute values. These attributes are supposed to be relevant and sufficient for describing their main features from a user's point of view. Let a_{ij} with i=1,m, and j=1,n be the value of attribute i for example j. Let b_{ik} with k=1,p be the value of attribute i for counter-example k. Let λ_j and ρ_k be the extent to which example j and counter-example k respectively are highly representative (from the user's point of view). Let us now consider a document d described by the attribute values $d_1,..,d_m$, supposed to be precisely known (the attributes are supposed to be the same as the ones used for describing the examples and counter-examples). Clearly, the more *similar* d to *(at least)* a representative example and the more *dissimilar* to *all* counter-examples, the more *possible* d is eligible for the user.

This can be estimated by the following expression:

$$\mu(d) = \min(ex, c\text{-}ex) \qquad (\mathbf{1})$$

where $ex = \max_j \min(\lambda_j, r_j)$

is the extent to which there exists *at least one* highly representative example similar to d provided that r_j be the similarity between example j and d.

Note that if all the examples are fully representative (i.e. $\forall j$, $\lambda_j = 1$) then $ex = \max_j r_j$ as expected; if $\lambda_j = 0$ example j is not considered,

where $c\text{-}ex = \min_k \max(1-\rho_k, s_k)$

is the extent to which *all* highly representative counter-examples are dissimilar to d provided that s_k be the dissimilarity between counter-example k and d.

Note that if all the counter-examples are fully representative (i.e. $\forall k$, $\rho_k = 1$) then

$c\text{-}ex = \min_k s_k$ reduces to a conjunctive aggregation; if $\rho_k = 0$, counter-example k is not considered,

Then, let S_i be a fuzzy similarity relation on the attribute domain of i with membership function μ_{Si} (S_i is supposed to be reflexive and symmetrical) and w_i be the level of importance of attribute i in a description,

the similarity between d and example j is computed as

$r_j = \min_i \max(1-w_i, \mu_{Si}(d_i,a_{ij}))$ (example j and d are similar as far as *all* the highly important attribute values which describe them are similar),

the dissimilarity between d and counter-example k is computed as

$s_k = \max_i \min(w_i, 1-\mu_{Si}(d_i,b_{ik}))$ (counter-example k and d are dissimilar as far as there exists an highly important attribute for which their respective values are much dissimilar).

When $w_i=1$ $\forall i$, r_j is just the conjunctive aggregation of the similarity degrees and s_k the disjunctive combination of dissimilarity degrees).

Finally, we get

$$\mu(d) = \min[\max_j \min(\lambda_j, \min_i \max(1-w_i, \mu_{Si}(d_i,a_{ij})), \min_k \max(1-\rho_k, \max_i \min(w_i, 1-\mu_{Si}(d_i,b_{ik}))] \quad (2)$$

This evaluation might be improved by requiring the similarity of d with *most* examples (see, e.g., [DPT88] for the introduction of a soft quantifier in a weighted min expression).

It is worth pointing out that the above expression can be used in another way that we now briefly outline and that is a topic for further research, apart from the ranking of documents w.r.t. a query. This expression also provides the starting point for generating a description of the documents which are looked for. This description may then be used for interface needs with the user. Indeed, letting d unspecified in the expression, it defines a compound fuzzy set expression which can be logically analysed. Suppose for simplicity that all the weights are equal to 1, we obtain:

$$\mu(.) = \min[\max_j \min_i \mu_{Si}(\,.\,,a_{ij}), \min_k \max_i 1-\mu_{Si}(\,.\,,b_{ik})] \quad (3)$$

Clearly, $\mu_{Si}(\,.\,,a_{ij})$ defines a fuzzy set of values close to a_{ij}, while $1-\mu_{Si}(\,.\,,b_{ik})$ defines a fuzzy set of values significantly different from b_{ik}, for each attribute i.

Note that the above expression may incorporate *interactivity* [Zad75] between attributes. For instance we have two examples of documents on a given topic, one which is 'short' without illustrations, and another which is 'long' but having many illustrations. Such a set of examples suggest that acceptable documents may be 'long' only if they have 'many illustrations' in them. In this case, there is an interaction between the length of the document and the number of illustrations. This interaction will be embodied in the above type of expressions and will make its reading less simple.

It should be pointed out that the set of examples and counter-examples provided by the user, enlarged by the similarity relations, does not usually cover all the cases which can be encountered. Namely, a document in the database may be just dissimilar to all the counter-examples without being similar to any examples. This suggests that if all the stored documents are in this situation, we may only use the c-ex part of the evaluation in order to avoid an empty answer to a request. However, the set of answers provided on the basis of c-ex only may be too large if it is not properly focused on the context of interest. For instance, looking for documents on a given topic having some length and number of illustrations characteristics, c-ex should only apply to these latter characteristics, in order not to retrieve all the documents on a different topic !

Another method for not "forgetting" the documents which may be in between the examples and the counter-examples, would be rather to allow for requests on the form: "all the documents satisfying some properties are relevant except the ones similar to counter-examples", or: "only the documents somewhat similar to at least one of the examples are relevant". Mixed requests specifying that the documents similar to example(s) are welcome, these similar to counter-examples should be excluded, while the remaining ones (in the context of interest) are provided to the user only if the search is enlarged (then it may help the user providing further examples and counter-examples).

6. Conclusion

The intended purpose of this paper was to provide a preliminary investigation of potential applications of fuzzy logic techniques in multimedia databases. Emphasis has been put on two querying issues: detecting modifications in semi-structured documents, and querying by examples. The different uses of fuzzy logic techniques in relation with these two querying problems have been discussed. The application of these tools to semi-structured documents could be developed to hypermedia. Our work could be adapted to the detection of changes in documents that can be represented as graphs but not necessarily as trees. Concerning the querying by examples, instead of counter examples, we may think of hybrid queries made of examples and of classical (fuzzy) restrictions expressing what attribute values are undesirable.

7. References

Abi97, Abiteboul S., Semi-structured information. In Proc. of Intl Conf. On Database Theory ICDT'97, International Conference on Database Theory, Invited talk, 1997.

ACM95, Abiteboul S., Cluet S., and Milo T., A database interface for file updates. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 1995.

AN97, Adjeroh D. A., Nwosu K. C., Multimedia Database Management Requirements and Issues, IEEE Multimedia, Vol. 4, n 3, pp. 24-33, 1997.

Ape97, Apers P. et al, Multimedia Database in Perspective, Springer-Verlag, 1997.

BCR98, Bosc P., Connan F., Rocacher D., Flexible querying in multimedia databases with an object query language. Proc. 7th IEEE Int. Conf. on Fuzzy Systems, Anchorage, May 5-10, 1308-1313, 1998.

BDPP97, Bosc P., Dubois D., Pivert O., Prade H. (1997) Flexible queries in relational databases —The example of the division operator— Theoretical Computer Science, 171, 281-302.

BK95, Bosc P., Kacprzyk J. (Eds.) Fuzziness in Database Management Systems. Physica-Verlag, Heidelberg, 1995.

BLP98, Bosc P., Liétard L., Prade H. (1998) An Ordinal Approach to the Processing of Fuzzy Queries with Flexible Quantifiers. in: Applications of Uncertainty Formalisms (A. Hunter, S. Parsons Eds.). Springer Verlag LNCS 1455, pp. 58-75, 1998.

BP92, Bosc P., Pivert O., Some approaches for relational databases flexible querying. J. of Intelligent Information Systems, 1, 323-354, 1992.

BP95, Bosc P., Pivert O., SQLf: A relational database language for fuzzy querying. IEEE Trans. on Fuzzy Systems, 3(1), 1-17, 1995.

BP97, Bosc P., Prade H., An introduction to the fuzzy set and possibility theory-based treatment of soft queries and uncertain or imprecise databases. In: Uncertainty Management in Information Systems: From Needs to Solutions (A. Motro, Ph. Smets, eds.), Kluwer Academic Publ., Chapter 10, 285-324, 1997.

CACS94, Christophides V., Abiteboul S., Cluet S., Scholl M., From structured documents to novel query facilities. 1994 ACM SIGMOD Intl Conf. on Management of Data, pp. 313-324, Minneapolis, May 1994.

CGM97, Chawathe S., Garcia-Molina H., Meaningful Change Detection in Structured Data, *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 1997.

CLS94, C. Chrisment, P.Y. Lambolez, F. Sèdes, Hyperdocument Management and Exchange in an OO System, Indo-French Workshop on OO Sytems, pp. 313-333, Goa, India, 1994.

C98, Connan F., Interrogation flexible dans un environnement objet. Rencontres Francophones sur la Logique Floue et ses Applications, Lannion, pp. 253-259, 1998.

CR97, Connan F., Rocacher D., Gradual and flexible Allen relations for querying video data. Proc. 5th Eur. Congr. on Intelligent Techn. and Soft Computing (EUFIT'97), Aachen, Germany, Sept. 8-11, 1132-1136, 1997.

CS98, Chrisment C., Sèdes F., Bases d'objets documentaires. Tutoriel, INFORSID'98, 1998.

CRGMW95, Chawathe S., Rajaraman A., Garcia-Molina H. and Widom J., Change detection in hierarchically structured information. *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 1996.

DeC97, De Caluwe R. (ed.) Fuzzy and Uncertain Object-Oriented Databases. Concepts and Models. World Scientific, Singapore, 1997.

DS96, Djennane S., Sedes F., Audio facilities for hypermedia consultation. 2nd Intl Workshop on Natural Language and Databases, NLDB'96, Amsterdam, IOS Press, 91-101, 1996.

DEGGLP98, Dubois D., Esteva F., Garcia P., Godo L., Lopez de Mantaras R. and Prade H., Fuzzy set modelling in case-based reasoning. Int. J. of Intelligent Systems, 13, 301-374, 1998.

DPT88, Dubois D., Prade H., Testemale C., Weighted Fuzzy Pattern Matching, Fuzzy Sets and Systems, Vol. 28 n° 3, 313-331, 1988.

FBY92, Frakes W. B., Baeza-Yates R., Information Retrieval: Data Structures and Algorithms, Prentice-Hall, 1992.

Gro97, Grosky W.I., Managing Multimedia Information in Database Systems, Communications of the ACM, Vol. 40, n 12, pp. 73-80, 1997.

Kif95, Kifer M., EDIFF- a comprehensive interface to diff for Emacs 19. Available through anonymous ftp at `ftp.cs.sunysb.edu`, 1995.

Kil92, Kilpelaïnen P., Tree matching problems with application to structured text databases. Tech. Report, Dept. of Computer Science, Univ. of Helsinky, Finland, 1992.

KZ86, Kacprzyk J., Ziolkowski A., Data base queries with fuzzy linguistic quantifiers. IEEE Trans. on Systems, Man and Cybernetics, 16(3), 474-478, 1986.

LGM95, Labio W. and Garcia-Molina H., Efficient algorithms to compare snapshots. Manuscript, available by anonymous ftp from `db.stanford.edu` in `pub/labio/1995/`, 1995.

LQC95, P.Y. Lambolez, J.P. Queille, C. Chrisment, EXREP : a generic rewriting tool for textual information extraction. Revue ISI, "Ingénierie des Systèmes d'Information", vol. 3, n° 4, pp. 471-485, 1995, Hermès.

MZ98, Milo T., Zohar S., Using Schema Matching to Simplify Heterogeneous Data Translation, VLDB'98, pp. 122-133, New York, August 1998.

Mye86, Myers E., A difference algorithm and its variations. *Algorithmica*, 1(2):251--266, 1986.

Pet96, Petry F.E., Fuzzy Databases: Principles and Applications. Kluwer Acad. Pub., Dord, 1996.

PGMW95, Papakonstantinou Y., Garcia-Molina H., and Widom J., Object exchange across heterogeneous information sources. In *Proceedings of the 11th International Conference on Data Engineering*, pp. 251-260, Taipei, Taiwan, March 1995.

SZ90, Shasha D. and Zhang K., Fast algorithms for the unit cost editing distance between trees. *Journal of Algorithms*, 11(4):581-621, 1990.

WSCRZP97, Wang J., Shasha D., Chang G., Relih V., Zhang K. and Patel G., Structural Matching and Discovery in Document Databases, *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pp. 560-563, 1997.

WZJS94, Wang J. T. L., Zhang K., Jeong K. and Sasha D., A system for approximate tree matching. IEEE Transactions on Knowledge and Data Engineering, 6(4) :559-571, August 1994.

Zad65, Zadeh L.A., Fuzzy sets. Information and Control, 8, 338-353, 1965.

Zad75, Zadeh L.A., The concept of a linguistic variable and its application to approximate reasoning, Information Sciences, Part 1: 8: 199-249, Part 2: 8: 301-357, Part 3: 9: 43-80. Reprinted in [SP], 219-366. 1975.

ZS89, Zhang K. and Shasha D., Simple fast algorithms for the editing distance between trees and related problems. *SIAM Journal of Computing*, 18(6):1245-1262, 1989.