1

Approximate Retrieval from Multimedia Databases Using Relevance Feedback

Ana Lelescu Ouri Wolfson Bo Xu
Department of Electrical Engineering and Computer Science
University of Illinois at Chicago
{alelescu,wolfson,bxu}@eecs.uic.edu

Abstract

In this paper we address the problem of retrieving stored multimedia presentations using relevance feedback. We model multimedia presentations using a crisp relational or object-oriented database, augmented with a text attribute. We also introduce a language for retrieval by content from such databases. The language is based on fuzzy logic. We also introduce a method for query refinement that uses relevance feedback provided by the user.

1 Introduction

In this paper we address the problem of retrieval by content from databases that contain multimedia presentations. We first introduce a model (i.e. an abstraction) of a multimedia presentation. Traditionally, retrieval by content from such databases concentrated on retrieval of images, and of video/audio sequences ([9, 12, 23]). In contrast, our model captures the textual, spatial, and temporal attributes of the presentation. In our model, for example, a slide in the presentation is represented by a relational tuple containing the text of the slide, the time at which the slide starts to be displayed on the screen and the end time of this display, and the coordinates of the location on the screen where the slide appears. The video/audio part of the presentation is represented by the audio transcription to text, which can be done using standard speech recognition techniques. This model is also appropriate for capturing collaborative work sessions. Thus, for example, the contents of a common whiteboard is represented by a tuple, and each participant is represented by another tuple in which the text attribute is his/her voice transcript. The database consists of *quants*, where each quant represents a multimedia presentation unit (that conveys some idea), or a collaborative work session; the reader can assume that a quant lasts between 5 and 20 minutes (although this assumption is not important for our results).

Next, we discuss the query language for retrieval of quants from the database. The same language can be used for filtering of quants in an environment in which collaborative work sessions occur continuously [6] and the user wants to join sessions that match his/her profile. The SQL language is unsatisfactory for retrieving and filtering since it performs exact matching. On the other hand, matching between the query and a quant may have to be satisfied in an approximate sense. For example, if the user is interested in quants that mention "light-weight batteries" and start at 2pm, he may find relevant a quant that starts at 2:01pm and mentions "alkaline batteries". The language consists of conjunctive queries in which an atom, e.g. "start_time $\sim=$ 2pm", is to be matched in an approximate sense using a fuzzy logic approach. The same approach to retrieval from multimedia databases was used by Fagin ([8, 9]), although the focus in his works has been on video and image retrieval. In the present paper, the atom "start_time $\sim=$ 2pm" is associated with a triangular fuzzy membership function (see [31, 29]) which normalizes the difference between the starting time of the quant and 2pm, by mapping this difference into a real number in the [0,1] interval. The number indicates the degree of similarity between the start-time of the quant and 2pm. Similarly, "text $\sim=$'alkaline batteries'" is mapped into a real number in the [0,1] interval using standard text retrieval techniques. Observe that in our model, in contrast to [33, 30, 21, 32], the data in the database is crisp, i.e. the database does not contain fuzzy terms such as "early", "small", etc.

Traditionally, in a multimedia retrieval model ([19, 23, 9, 14]) the query is represented as a point in multidimensional space, where each dimension is a feature (e.g. a keyword for documents, color and shape for images). Retrieval is performed using a similarity function that measures the distance in multidimensional space between the query point

*This research was supported in part by grants DARPA N66001-97-2-8901, NATO CRG-960648, NSF grants IRI-9712967, CCR-9816633, and CCR-9803974, and Army Research Labs grant DAAL01-96-2-0003

and the point that represents each object in the database. In contrast, we model the query as a set of similarity functions. Each similarity function is the fuzzy membership function associated with each atom. Further, the multimedia retrieval model considers a weighted sum [23] or the Euclidean distance [14] as the aggregation function to combine query components, whereas we use the fuzzy aggregation function *min*, as accepted in the fuzzy logic literature (see [27, 9]).

For approximate retrieval queries it is important to provide the user with iterative and interactive query refinement mechanisms. We do so by using relevance feedback, i.e. the user labels some of the retrieved quants as relevant or irrelevant. Based on this feedback the system automatically refines the query and resubmits it to retrieve a new set of quants. The process continues until the system cannot provide any new quants that satisfy the latest version of the query.

Traditionally, methods of query refinement based on user-feedback use two techniques ([19, 20]), namely query modification and query re-weighting. The first technique involves either query point movement [14] or query expansion [19]. The second method automatically adjusts the relative importance of each feature (i.e. weight) to the query. Our approach to query refinement is not using any of the above methods. The main idea is to perform query refinement by modifying the similarity function for each fuzzy atom in the query based on the relevance feedback. Specifically, we propose an algorithm to automatically adjust the set of fuzzy membership functions. In addition, we incorporate in the relevance feedback process existing information retrieval techniques for text (see [22]).

Although relevance feedback is natural for approximate queries using fuzzy logic, and it is a well accepted and studied technique in text retrieval (see [30, 25, 26]) and in multimedia retrieval by content ([14, 18]), as far as we know this is the first time a query refinement technique is applied in fuzzy logic systems.

This paper is organized as follows. In section 2 we present the data model for collaborative session. In section 3 we describe a vague query language. In section 4 we present our approach of query refinement by relevance feedback and in section 5 we evaluate the performance by simulations. Section 6 includes some implementation consideration and section 7 summarizes relevant work. We conclude in section 8.

2 DATA MODEL

A collaboration session or a multimedia presentation consists of a sequence of *quants*, each of which conveys an idea. A quant is represented by a set of media *objects* (e.g. video, audio, viewgraphs and other text files).

Each object is presented to the user in accordance with certain spatial and temporal attributes. The spatial attributes give the object's destination location on the screen (e.g. a rectangular window given by its Cartesian coordinates) and its source location in the network (e.g. IP address, host name or city). [1] The temporal attributes are start-time and end-time, which specify the time interval during which the object is on the screen. Each media object in the quant is associated with some form of textual information. For example, a voice transcript is the text attribute for an audio-visual segment, and the text contained in a PowerPoint slide is the text attribute for the slide. In summary, each object O is represented by a tuple with standard database attributes (e.g. title, author, date, type, etc), a text attribute *text*, temporal attributes *start_time* and *end_time*, and possibly two spatial attributes, *source_location* and *screen_location*.

Example1

Consider a quant created for a class lecture that contains four media objects: a journal article, the lecture audio track, and two PowerPoint slides, slide1 and slide2. In Figure 1 the quant's content is described using logical visual screens to specify its spatio-temporal structure. At time t_1=2:01pm the instructor starts the lecture, and the voice transcript is played continuously during the time interval (2:01pm, 2:20pm). Note that the voice transcript has no screen_location attribute. However its source_location spatial attribute is 193.100.2.1. During the instructor presentation (i.e. while the voice transcript is played), at time t_2= 2:02pm a journal article and slide1 appear on the screen. The time interval for slide1 is (2:02pm, 2:10pm) and its spatial attributes are given by its location on the screen and source address respectively. At time t_3=2:10pm slide1 will disappear from the screen and another slide, slide2, is viewed at the same screen location during (2:10pm, 2:20pm). Any modification in the set of active objects marks a screen change: the screen now contains the voice transcript and the journal as before, and slide2 instead of slide1. The relation that contains the quant presented in this example is given in Figure 2. The ID attribute denotes the quant's unique identification number. □

Note that in this paper we adopt the relational data model. However, our results are not restricted to this model, and in fact, they carry over even if the quant is represented as a complex object in the object oriented model.

3 THE QUERY LANGUAGE

In this section we introduce a fuzzy query language, called *fuzzy-SQL*, that allows the user to retrieve quants from a regular database (i.e. a database containing precise facts), based on their content. The same query language is

[1] Note that our *source_location* attribute is a spatial attribute in the sense that it can be mapped to a geographical location.

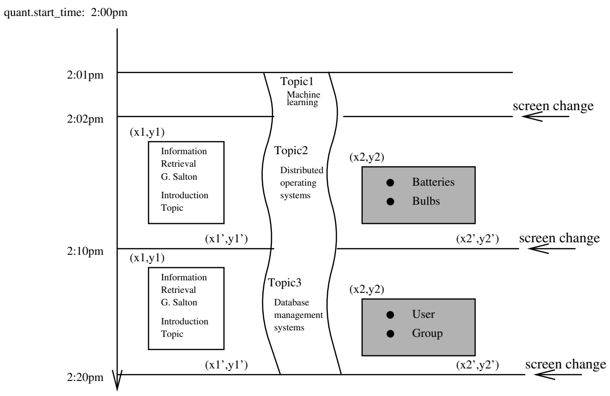

Figure 1. A lecture quant

ID	*Date*	*Author*	*Type*	*Text*	*start_time*	*end_time*	*screen_location*	*source_location*
1	4/3/98	John	journal	Information Retrieval G. Salton Introduction topic	2:02pm	2:10pm	(x1,y1)(x1',y1')	193.100.2.1
1	4/3/98	John	slide1	• Batteries • Bulbs	2:02pm	2:10pm	(x2,y2)(x2',y2')	193.100.2.1
1	4/3/98	Dan	slide2	• User • Group	2:10pm	2:20pm	(x2,y2)(x2',y2')	193.100.2.1
1	null	Laura	voice transcript	Topic1 Machine learning Topic2 Distributed operating systems Topic3 Database management systems	2:01pm	2:20pm	null	193.100.2.1

Figure 2. A relation representing the quant in Example 1

used for two different retrieval scenarios, namely the *off-line* scenario (i.e. traditional retrieval from a database), and the *on-line* scenario (i.e. filter/trigger of incoming quants). For the on-line scenario, the quants arrive continuously one at a time (rather than reside in a database) and the retrieval process is reduced to independent binary decisions to accept or reject each quant. Our goal is to provide a high level language such as SQL, that treats both scenarios in a uniform way and provides support for approximate matching.

3.1 Syntax

The fuzzy-SQL language is conjunctive SQL (or conjunctive relational calculus (see [28])), extended to accommodate approximate matching. The SQL syntax is extended in two ways ([1, 29]). The first extension generalizes the definition of atoms, or atomic conditions (see [28]), to include the following approximate comparison operators: $\theta \in \{\sim=, \sim>, \sim<\}$, i.e. "approximate-equal", "approximate-bigger", "approximate-smaller". An atom is called an *approximate atom* if it contains an approximate operator. An approximate atom is called *textual* if it involves the TEXT attribute. For nontextual approximate atoms, we distinguish between *numeric atoms*, i.e. atoms that involve an attribute with a numeric domain, and *symbolic atoms* which do not do so. The only approximate operator allowed in a textual atom or in a symbolic atom is $\sim=$.

The second extension of the SQL syntax refers to the fact that each object in the database satisfies a query with approximate atoms with a certain degree of similarity. Therefore, we introduce a function called *score* to measure this value, and a parameter T that represents the acceptance threshold. The syntax of an extended SQL query is therefore:

SELECT < attributes>
FROM <relations>
WHERE score(A1 and A2 ... and Am) $>= T$

where each Ai is an atom (crisp or approximate), and T is a real number from the $[0, 1]$ interval.

Intuitively, the answer to this query is a set of tuples for which the degree of satisfaction is greater than or equal to T. The main motivation behind using the threshold value instead of the traditional top k (see [8, 4]) is to allow us to provide a unified language for both the on-line and the off-line scenarios (note that top k is meaningless in the on-line case).

Example 2

Consider the query in Figure 3.

```
SELECT O1.id
FROM collab_session O1
WHERE
    score(O1.author = "John D."
          AND O1.text ~= "multimedia retrieval and presentation"
          AND O1.start_time ~= 2pm)>=0.8
```

Figure 3. A fuzzy-SQL query

Intuitively, the query retrieves the id of the quants that contain a tuple O1 such that: (1) the author attribute value for O1 matches "John D." in an exact sense, and (2) the text value for O1 approximately matches the phrase "multimedia retrieval and presentation", and (3) the start_time value for O1 is approximately 2pm. For an object to be retrieved, the aggregate (i.e. minimum) score of the above atoms must be at least 0.8. □

3.2 Semantics

Let Q be a fuzzy SQL query and D be a database. Consider an *assignment* for Q, i.e. a tuple in the database D from each relation mentioned in the FROM clause of Q. The score of a crisp (i.e. nonapproximate) atom is either 0 or 1, depending on whether or not the atomic condition is satisfied by the assignment. In order to assign a score to an approximate atom A in a query Q, we associate with A a similarity function S: S assigns a score in the interval [0,1], which indicates how well the assignment satisfies the atom; a score of 1 indicates a perfect match. The overall score of the assignment is given by an aggregation function that combines the assignment's scores on each atom. In this paper, we adopt the standard fuzzy-logic aggregation function min ([7, 27]). The assignment satisfies the query if the minimum score of all atoms is higher than the threshold T.

Next we discuss the similarity function associated with each approximate atom A in the query Q. We distinguish between textual and nontextual approximate atoms. (For example, the second fuzzy atom in Figure 3 is textual.) For a textual approximate atom, S is a standard text similarity function such as the cosine (see [25]). It assigns a score to the assignment which indicates how similar are the texts that appear on both sides of the $\sim=$ operator.

For nontextual approximate atoms, we use a fuzzy logic approach ([31, 29]). A numeric atom with the approximate operator θ is associated with a fuzzy membership function denoted $\mu_\theta(x)$, where x is the difference between the values of the two operands in the atom. For example, given the fuzzy atom "O.start_time $\sim=$ 2pm" and a tuple O, x is the difference between the start_time attribute of O and 2pm. The membership functions are defined as follows and illustrated in Figure 4; a and b are called the *parameters* of each membership function.

$$\mu_{\sim=}(x) = \begin{cases} 0 & \text{if } x \le a \text{ or } x \ge b \\ 1 & \text{if } x = 0 \\ \frac{a-x}{a} & \text{if } a < x < 0 \\ \frac{b-x}{b} & \text{if } 0 < x < b \end{cases}$$

$$\mu_{\sim>}(x) = \begin{cases} 0 & \text{if } x < a \\ \frac{x}{a} & \text{if } a \le x < 0 \\ 1 & \text{if } x \ge 0 \end{cases}$$

$$\mu_{\sim<}(x) = \begin{cases} 1 & \text{if } x < 0 \\ \frac{x}{a} & \text{if } 0 \le x \le a \\ 0 & \text{if } x > a \end{cases}$$

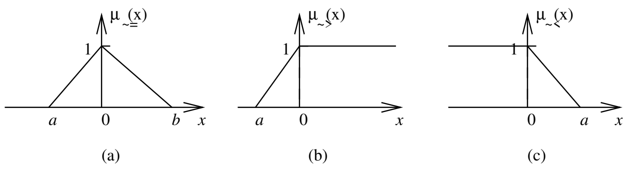

Figure 4. Fuzzy membership functions for the approximate comparison operators

The case of symbolic approximate atoms can be reduced to the numeric case by assuming that there exists a similarity (or distance) function that gets as input any two symbolic values from the domain of the atom, and outputs the similarity between them. For example, suppose that the query contains the atom "O.source_location $\sim=$ 'Sears Tower'". If for some tuple source_location is 'Chicago', then the score of the atom produced by the similarity function may be 0.8. Thus, with this assumption, the results of this paper apply to queries with symbolic approximate atoms as well.

4 QUERY REFINEMENT BY RELEVANCE FEEDBACK

The main idea of our query refinement approach is to iteratively modify the membership functions associated with a query, using the information provided by the user about the relevance of objects retrieved up to the current iteration. The overall goal of the modification is to retrieve all the relevant objects if enough iterations are requested by the user, while minimizing the number of irrelevant objects that are presented to the user. Hence, query refinement by relevance feedback is an iterative process involving user interaction and membership function modification. Each iteration can be summarized as follows:

1. Retrieve objects from the database using the current query;

2. If there are "new" objects retrieved (i.e. objects that were not marked relevant or irrelevant by the user in previous iterations), present them to the user; Otherwise, go to step 4.

3. The user examines the objects presented. If the user does not request the next iteration, query refinement stops. Otherwise, the user marks some of the objects as relevant or irrelevant and step 4 is executed.

4. Execute the Membership Function Modification (i.e. MFM) algorithm to adjust the query (see Section 4.2 below).

The same query refinement procedure can be applied for both on-line and off-line retrieval scenarios.

In the next two subsections we present the MFM algorithm. In section 4.1 we introduce the preliminaries of the algorithm and in section 4.2 we describe the algorithm.

4.1 Preliminaries

In the following, for ease of exposition, we will assume that each atom A_i has the form $O.M_i \sim= 0$, where M_i is a numeric attribute. With each A_i is associated a triangular membership function $\mu_i(x)$ (see Figure 4(a)). Furthermore, since the algorithm is symmetric for either side of $\mu_i(x)$, w.l.g. we will consider only the right side, i.e. we will assume that all the values of M_i are positive. We denote the membership function in this case by $\mu_i(x : b_i)$, where b_i is the right bound.

Given a set of retrieved objects F, the *relevance assignment* of F is a pair of disjoint subsets of F, namely

(F_R, F_I), where F_R contains all the objects marked "relevant" and F_I contains all the objects marked "irrelevant". [2] A relevance assignment (F_R, F_I) is called *complete* if F_I is empty. At each iteration the system maintains a *marked objects collection* (MOC) which accumulates the marked objects.

Our algorithm involves two types of membership function modifications, namely *shrink* and *expand*. The shrink-modification changes the parameters of a membership function such that (1) the similarity of each object that has been marked relevant is higher than the threshold T; (2) the similarity of each object that is presented but not marked at this iteration is higher than T; and (3) The number of objects that have been marked irrelevant and that have similarities lower than T is maximized. The expand-modification increases the right bound of each membership function μ_i by a positive constant value c_i, i.e. a membership function $\mu_i(x : b_i)$ becomes $\mu_i(x : b_i + c_i)$. We call c_i the *expansion constant* of μ_i. The value of c_i depends on the density of the objects along the axis, and therefore it is a parameter to the MFM algorithm for each fuzzy membership function.

4.2 Algorithm Description

The MFM algorithm modifies the current membership functions as follows. If no new objects are retrieved at this iteration or the relevance assignment is complete (i.e. all the objects that the user marked at this iteration are relevant), MFM expands each $\mu_i(x : b_i)$ (expecting to retrieve more relevant objects). Otherwise, it shrinks $\mu_i(x : b_i)$. Below is the pseudo code of the MFM algorithm.

INPUT F, F_R, F_I, MOC, **each** $\mu_i(x : b_i)$
IF (there are no "new" objects retrieved (an object retrieved is "new" if it is in F **but not in MOC))**
  expand each $\mu_i(x : b_i)$
ELSE
  IF (the relevance assignment is complete)
    expand each $\mu_i(x : b_i)$
  ELSE
    $FMOC = F \cup MOC$
    FOR each $\mu_i(x : b_i)$
      find out in $FMOC$ **the non-irrelevant**
      (i.e. relevant or not-marked) object
      which is the greatest (i.e. the farthest from 0)
      IF (there is such an object O**)**
        $\mu_i(x : b_i)$ **becomes** $\mu_i(x : O.A_i/(1-T))$
      ELSE //all the objects in $FMOC$ **are irrelevant**
        $\mu_i(x : b_i)$ **becomes** $\mu_i(x : 0)$
      END
    END
  END
  $MOC = MOC \cup F_R \cup F_I$
END

[2]The union of F_R and F_I may be a proper subset of F, i.e. the user does not necessarily mark every object as relevant or irrelevant. Indeed, the user does not know a priori how many objects will be retrieved in an iteration, and the retrieved set may be too large to examine in its entirety.

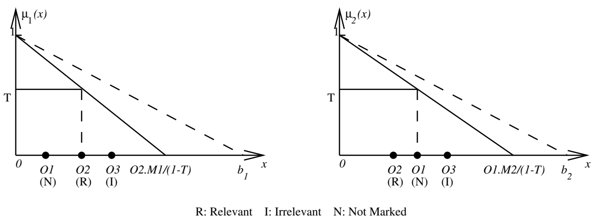

Figure 5. A shrink modification example

Note that expand-modification is fairly straightforward. Therefore, in the following we choose to give an example of the shrink-modification.

Example 3

Consider a query with two atoms A_1 and A_2. Suppose that at the beginning of an iteration, the two membership functions associated with these two atoms are $\mu_1(x : b_1)$ and $\mu_2(x : b_2)$ (see Figure 5). Assume that the marked object collection MOC is $\{O_2\}$. Suppose that there are three objects retrieved at this iteration, namely O_1,O_2 and O_3. Assume that the relevance assignment is $(\emptyset, \{O_3\})$, i.e. O_3 is marked irrelevant and O_1 is not marked (O_2 is not presented to the user, since it is not new). Since the relevance assignment is not complete, MFM executes the shrink modification for both $\mu_1(x : b_1)$ and $\mu_2(x : b_2)$. For μ_1, since O_2 is the farthest non-irrelevant object from 0, $\mu_1(x : b_1)$ is shrunk to $O_2.M1/(1-T)$ (see Figure 5(a)). Thus, the score of O_3 is lower than T, and the scores of O_1 and O_2 are higher than or equal to T. Similarly, the new bound of μ_2 is determined by O_1 (see Figure 5(b)). After the shrink modification, the overall score of O_3 is below T and the overall scores of O_1 and O_2 are above or equal to T.$\Box$

5 EXPERIMENTS

The main purpose of our experiments is to evaluate the performance of the above proposed MFM algorithm. The evaluation is conducted by comparing it with the MindReader system described in [14]. We also wanted to investigate the impact of the user iteration model on the performance of our algorithm.

5.1 Simulation environment

We implemented our MFM algorithm and conducted experiments using synthetic data. We randomly generated a database of 2000 objects in a n-dimensional space (i.e. objects with n attributes) where n was chosen to be either 2 or 5. For each dimension the data set is normalized in the interval $[0, 1]$. We use these attributes to build a query that has n atoms of the form $M_i \sim= 0$ for $i = 1, 2, ..., n$.

The relevance assignments were generated for each object using the following model. For each dimension we set up two intervals $[0, x_1]$, $(x_1, 1]$, $0 \leq x_1 \leq 1$. If an object falls into $[0, x_1]$ for each dimension, then it is relevant; otherwise it is irrelevant.

We performed two sets of experiments, that differed in the number of dimensions (2 or 5). The experiments in each set differed in the percentage of relevant objects. Each set consisted of three experiments, having 5%, 20%, and 50% of objects being relevant. Each experiment is conducted as follows. We start with arbitrary membership functions. We assume that the user will mark the retrieved objects from top to bottom until he meets the first irrelevant object. Each experiment stops when all the relevant objects have been marked by the user.

For MindReader the query is a point in the n-dimensional space and each dimension is associated with a weight (see [14]). Given a set of objects with relevance assignments, MindReader estimates the ideal query and weights. For the purpose of this experiment, at the first iteration, we train MindReader with the set of objects that were retrieved and marked at the first iteration in our algorithm. After that, at each iteration MindReader uses the previous query to retrieve top k objects and then adjusts the query and the weights based the relevance assignment for them. At each iteration we make k the same as the number of objects marked for MFM at that iteration.

5.2 Performance Results

5.2.1 Comparison with MindReader

The comparison between MFM and MindReader is based on the recall and precision (see [25] for the definition of these concepts), since they determine user's satisfaction. The comparison is conducted as follows. At each iteration, we compute the *cumulative recall+precision* which is the sum of recall and precision calculated based on the marked objects collection up to that iteration (i.e. MOC). In other words, the cummulative recall and precision at iteration i is the recall plus precision computed based on the MOC set as it exists at the i'th iteration.

Figure 6 illustrates the comparison results for 20% of objects in the database being relevant. The results for 5% and 50% of objects being relevant are similar. From Figure 6 we may see that the performance of MFM is consistently better than that of MindReader. Initially, the performance of MFM is very close to that of MindReader. However, the difference between them increases with the number of iterations. Regardless of the number of dimensions, the MFM algorithm is almost perfect, i.e. the precision and recall sum converges to 2.

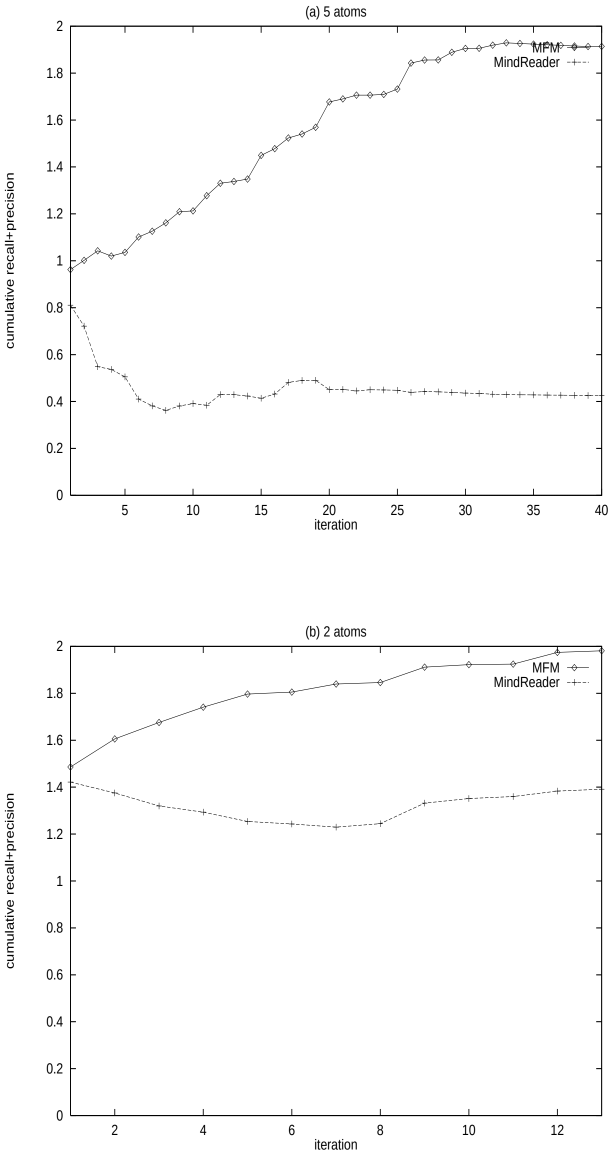

Figure 6. Comparison between MFM and MindReader

5.2.2 The Influence of User's Interaction

The following experiments illustrate two extreme examples of the user's marking behavior for the MFM algorithm. In the first scenario, the user always marks the ranked list of retrieved objects from top to bottom, and in the second scenario the user marks randomly from the rank list. Figure 7 shows the result of our simulation for both marking behaviors. We see that the top-bottom mode is much better than the random one and it converges much faster. Fortunately, top-bottom marking is also the most reasonable user behavior.

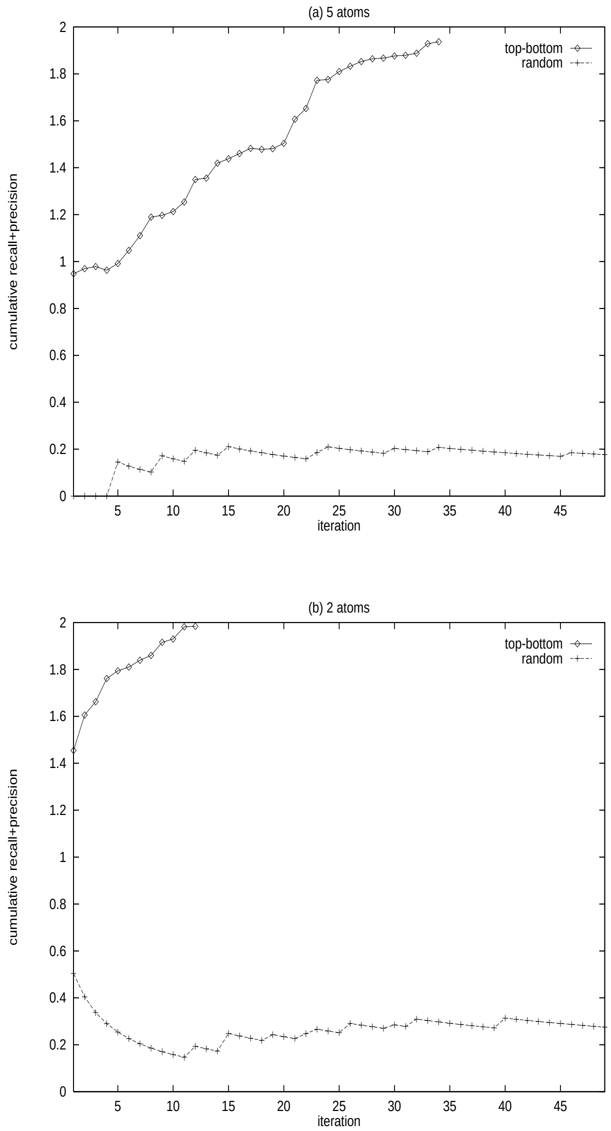

Figure 7. Comparison between top-bottom and random relevance feedback

6 Implementation Considerations

Since the query language is formulated in an extended-SQL dialect, the query processing is based on transforming it into one or more regular SQL queries. For example, consider the atom "A=(O.start_time $\sim=$ 2pm)" in a query with threshold T. Let $[a, b]$ be the interval for which $\mu(x) \geq T$, where $\mu(x)$ is the atom membership function. Then atom A is converted into a range condition $a <$ O.start_time $< b$.

We implemented the fuzzy query as a wrapper around a conventional DBMS using INFORMIX. The main advantage of using an object relational system is the fact that it offers support for users to define any new data types and operations (i.e user defined functions) on them. The approximate comparison operators were implemented in the C language. For text attributes, our implementation uses a commercially available library of functions designed to process text using IR techniques. This package is provided by the Excalibur Text DataBlade for INFORMIX.

7 RELEVANT WORK

Vague Queries in DBMS

The need for vague or imprecise queries has been pointed out for many years and has been addressed in the context of fuzzy databases ([30, 21, 32]). Vague queries are formulated by introducing definitions of fuzzy attribute values (e.g 'young') and fuzzy comparators similar to the approximate operators presented in this paper. In contrast, we assume that the database stores only crisp information and our query language is intended to be used in combination with existing conventional database systems and IR technology. None of the current work in fuzzy databases provides any means for query refinement.

For crisp databases, there are two approaches in current literature which support vague queries. None of them is based on fuzzy logic. First, the VAGUE system introduced in [16] is based on a variant of the vector space model for database relations. The distance between two tuples is computed by aggregating the individual distances between the corresponding attributes by means of a weighted Euclidean distance measure. In contrast, our aggregation function is is given by min. Moreover, VAGUE doesn't use relevance feedback. Second, Fuhr introduced in ([11, 10]) a method of integrating text queries and vague queries in databases based on a probabilistic model using relevance feedback. The main disadvantage there is that the query language is more restrictive due to independence assumptions.

Multimedia Retrieval

Our concept of approximate queries is similar to that of content-based queries in multimedia applications [12]. Most recent work in this area focused on specific data types such as text, images, video audio, etc. There is a solid IR technology to retrieve documents based on their content ([25, 26]), and research in image processing led to systems such as QBIC [17], MARS [23]. In contrast, we introduced a novel data model that combines text, temporal and spatial data types for fuzzy retrieval of collaboration sessions. The idea of using a multimedia retrieval model based on fuzzy sets appears in ([8, 18]). However, none of these works uses relevance feedback.

Relevance Feedback

A different approach to modify the membership function for a fuzzy attribute value (such as "young") was introduced in [30]. The modification is motivated by the fact that each fuzzy term may have a different meaning in the query. However, this modification does not involve relevance feedback.

Relevance feedback methods have been extensively studied in IR area ([2, 3, 22]). For the text part of our query refinement approach, we assume that we can plug in any IR system that supports relevance feedback, such as SMART [25].

Most of the research using relevance feedback in a multimedia retrieval system is related to the MARS system (see [19, 20, 23]) and MindReader [14]. Different models for query reweighting and query modification were explored. There are three different query refinement strategies, namely query point movement [14], query expansion model [19], and query reweighting [23]. Query point movement moves the query toward relevant objects and away from the irrelevant ones. Query expansion adds more components to the query. Our approach is completely different than the above methods. First, for query reweighting, if the weight on one dimension is increased, it is decreased for another one, but in our case we expand or shrink all atoms simultaneously. Second, we do not move query and we do not add components to the query.

The work in [24] considers an image retrieval system that supports multiple similarity measures for a given image feature such as shape. Relevance feedback mechanism is employed here to identify the similarity measure that actually fits the user. In contrast, we use a single similarity function for each attribute and we modify this accordingly.

8 Conclusion

In this paper, we focused on the problem of retrieval by content from databases that contain collaborative work sessions. We first introduced a data model that captures the textual, spatial, and temporal attributes of multimedia sessions (quants). Then we described a fuzzy query language that uses approximate comparison operators to retrieve the quants for two different scenarios, i.e. on-line and off-line. For the on-line scenario the quants arrive continuously, whereas for the off-line scenario they reside in a database. The main contribution of this paper is a new approach to perform query refinement based on user's relevance feedback by modifying the fuzzy membership function associated with each atom. We proposed an algorithm (i.e. MFM) to automatically adjust the set of fuzzy membership functions in the query. The simulation results show that the MFM algorithm outperforms the MindReader system.

Acknowledgements

We wish to thank Prof. Clement Yu and Prof. Simon Kasif for helpful discussions.

References

[1] Bosc P., Galibourg M., Hamon G., "Fuzzy querying with SQL: Extensions and implementation aspects", *Fuzzy Sets and Systems*, 28, 1988.

[2] Buckley C., Salton G., "Optimization of Relevance Feedback Weights", in *Proceedings of the 18th Annual Intl ACM SIGIR Conference*, Seatle, 1995.

[3] Buckley C., Salton G.,Allan J., "The effect of adding relevance feedback information in a relevance feedback environment" in *Proceedings of the 17th Annual Intl ACM SIGIR Conference*, 1996.

[4] Chaudhuri S., Gravano L., "Optimizing queries over multimedia repositories", *Proceedings of ACM SIGMOD'96*, Montreal, Canada, 1996.

[5] Croft B, Callan J., et all, "Integrating IR and RDBMS Using Cooperative Indexing", in *Proceedings of the 18th Annual Intl ACM SIGIR Conference*, Seatle, 1995.

[6] Dao S., Shek E., Vellaikal A., Muntz R., Zhang L., Potkonjak M., Wolfson O., "Semantic Multicast: Intelligently Sharing Collaborative Sessions", *ACM Computing Surveys*, 1998.

[7] Dubois D., Prade H., "Criteria aggregation and ranking of alternatives in the framework of fuzzy set theory", *Fuzzy Sets and Decision Analysis* (H.J. Zimmermann, Zadeh L.A, Gaines B., Eds).

[8] Fagin R., "Combining Fuzzy Information from Multiple Systems", in *Proc. 15th ACM Symp. on Principles of Database Systems*, 1996.

[9] Fagin R., "Fuzzy queries in multimedia database systems", in *Proceedings of ACM SIGMOD Conf*, 1998.

[10] Fuhr N., "Integration of probabilistic fact and text retrieval", in *Proceedings of 15th Annual Intl ACM SIGIR Conference*, Denmark, 1992

[11] Fuhr N., Rolleke Th., "A Probabilistic Relational Algebra for Integration of Information Retrieval and Database Systems", in *ACM Transactions on Information Systems*, 15(1), 1997

[12] Grosky W.I., "Managing multimedia information in database systems", *Communications of the ACM*, 40(12), 1997.

[13] Harman D., "Relevance feedback revisited", in *Proceedings of 15th Annual Intl ACM SIGIR Conference*, Denmark, 1992

[14] Ishikawa Y., Subramanya R., Faloutsos C., "MindReader: Querying databases through multiple examples", *Proceedings of the 24th VLDB Conference*, New York, 1998

[15] Korth H.F., Silbershatz A., *Database System Concepts*, McGraw-Hill, Second Edition, 1991.

[16] Motro A. "VAGUE: A user interface to relational databases that permits vague queries" *ACM Transactions on Office Information Systems*, 6(3), 1988.

[17] Niblack W., et al. "The QBIC Project: Querying Images by Content Using Color, Texture and Shape", in *Proceedings SPIE*, San Jose, CA, 1993.

[18] Ortega M., et al. "Supporting similarity queries in MARS", in *Proceedings of ACM Conference on Multimedia*, 1997.

[19] Porkaew K., Ortega M., Mehrotra S., "Query Reformulation for Content Based Multimedia Retrieval in MARS", in *Proceedings of ACM Conference on Multimedia*, 1999.

[20] Porkaew K., Mehrotra S., Ortega M., Chakrabarti K., "Similarity Search Using Multiple Examples in MARS", in *Intl Conference on Visual Information Retrieval*, 1999.

[21] Prade H. Testamale C., "Generalizing relational algebra for treatment of incomplete or uncertain information and vague queries", *Information Sciences*, 34(2), 1984.

[22] Rochio J.J., "Relevance Feedback in Information retrieval", in Salton Ed.. *The Smart Retrieval System Experiments in automatic document processing*, Prentice Hall, Englewoods Cliffs N.J, 1971.

[23] Rui Y., Huang T.S., Mehrotra S., "Content-based image retrieval with relevance feedback in MARS", *Proceedings of IEEE Intl Conference on Image Processing*, 1997.

[24] Rui Y., Mehrotra S., Ortega M., "Automatic matching tool selection using relevance feedback in MARS", in *Intl Conference on Visual Information Retrieval*, 1997.

[25] Salton G., *The SMART Retrieval System-experiments in automatic data processing*, Prentice Hall, Englewood Clifs NJ, 1971.

[26] Salton G., Buckley C., "Term weighting approaches in information retrieval", *Information Processing and Management*, 24(5), 1988.

[27] Thole U., Zimmermann J., Zysno P., "On the suitability of minimum and product operators for the intersection of fuzzy sets", *Fuzzy Sets and Systems*, (2), 1979.

[28] J. D. Ullman, *Principles of Database Systems*, Computer Science Press, 1982.

[29] Yen J., Langari R., *Fuzzy Logic: Intelligence, Control and Information*, Prentice Hall, 1999.

[30] Yu C., Meng W., *Principles of Database Query Processing for Advanced Applications*, Morgan Kaufmann Publishing, 1998.

[31] Zadeh L.A., *Fuzzy Sets*, Information and Control, 8, 1965.

[32] Zemankova M., Kandel A., "Implementing imprecision in information systems", *Information Sciences*, 37, 1985.

[33] Zhang W., Yu C., Reagan B., Nakajma H., "Context dependent interpretation of linguistic terms in fuzzy relational databases", *IEEE Data Enginning Conference*, 1995.