Satisfying Varying Retrieval Requirements in Case-Based Intelligent Directory Assistance

Vivek Balaraman
Sutanu Chakraborti

Artificial Intelligence Group
Tata Research Development and Design Centre, Pune, India - 411013
Email: vivekb,sutanuc@pune.tcs.co.in

Abstract

Intelligent directory assistance is the ability to retrieve relevant subscriber records in the presence of mismatches between the query and the subscriber record. The challenges are to provide inexact retrieval over very large data volumes within a reasonable timeframe as well as the ability to tailor retrieval to differing query requirements. In this paper we discuss a case-based approach to intelligent directory assistance which focusses on a layered approach to retrieval. The case-base is multiply indexed where each level indexes the case-base at a level of dissimilarity. We introduce the concepts of Levels of Lenience, Degree of Lenience and Search templates and demonstrate how these enable retrieval to be fast and yet tailored to query requirements. A production quality system based on these ideas has been implemented at a leading telecommunications service provider and is giving excellent performance.

Intelligent Directory Assistance

A directory assistance (henceforth DA) or subscriber assistance system provides information on it's subscribers. People query these systems based on partial information about the subscriber(s). Often, there is a mismatch between the information provided by the caller and the information in the subscriber repository. The mismatch may be due to the query having incorrect information about the target subscriber or being improperly understood by the call operator. Common errors are: Misspellings, synonyms, variations in ordinal forms, abbreviations and acronyms, different punctuation, different levels of specification, pronunciation and transcripting errors and numeric, enumeration and date errors. We estimate that there are in all twenty one different mismatches possible between a query and a subscriber record.

Due to these factors, the query and the record being searched may not correspond exactly. Unless the system can handle such inaccuracies, the call operator has to do a time consuming process of tweaking the query to locate the right record leading to long call closure times and poor customer service. Intelligent directory assistance systems are systems that enable call operators or users in the case of self-service systems to locate the most relevant records from a repository in the presence of mismatches.

Developing intelligent directory assistance systems throws up many engineering challenges: a) The system must be capable of retrieving the most relevant records from a very large volume within a limited time frame (typically 5 to 10 seconds) in the presence of mismatches. Since brute force inexact matching is computationally expensive, strategies are required to minimise the time taken to retrieve relevant records. b) The system should be capable of catering to different user query requirements. Directory assistance systems often have to cater to two types of queries, viz., queries where the user is looking for a specific entity in the subscriber base to yellow pages queries where the user is looking for all fits to a specification.

In this paper we discuss a case-based approach to intelligent directory assistance based on a layered approach to retrieval. The layering indexes the subscriber base at different levels of dissimilarity. The approach introduces concepts which we label Level of Lenience, Degree of Lenience and Search Templates. These enable the retrieval process to be fast and yet tailored to user needs.

Case Based Reasoning

Our approach is centered around case based reasoning (CBR). The techniques underlying CBR such as similarity based retrieval are applicable in more than experience retrieval. CBR is already widely used in recommender domains such as real estate estimation (Gonzalez and Laureano Orti 1992) (O'Roarty et al 1997) where cases are real estate properties. Similarly in this case, each subscriber record is modelled as a case where a case is a set of attribute value pairs. The case-base consists of the set of subscriber records. The input case is compared with the case-base to yield cognitively valid subscriber records. As the function of the system is only to provide effective retrieval, case-based processes such as case adaptation are not required. Case acquisition and maintenance is also less of an issue. Since directory assistance is a key function of telecommunications service providers, subscriber data is often available in standard data formats which can be ported to the form required by the case-based reasoner.

The challenges in this domain are across the axes of efficient yet effective retrieval where the retrieval needs to

be fast and tailored to different query requirements. In this paper we discuss our approach to tackling these problems and demonstrate it on name and address matching. We should stress that the techniques discussed in this paper are local similarity measures (attribute level match techniques).

Lenience

Domains like directory assistance, job search and real estate search share certain characteristics: they require inexact search over very large data volumes N where the output is a small set of relevant records k, where k << N (for example in DA, k may be around 10 where N is of the order of 1000,000 or more) and secondly, have performance requirements where the time for retrieval has to be less than an upper bound.

Most case-based systems are singly indexed and retrieval time is independent of the distance between the input case and the set of the retrieved cases. While this may not be a problem when case volumes are low, it is a significant problem with large case volumes.

Our approach is based on the following decisions: a) Retrieval time should depend upon the distance between the input case and the set of relevant cases and b) Each case to be multiply indexed where an index tolerates an extent or level of dissimilarity between the input and the set of relevant cases. Lower the index level, lower the dissimilarity tolerated and higher the index level, higher the tolerance for mismatches. Correspondingly the search space at an index level is smaller than that at a higher level and thus in general lower the index level lower the retrieval time.

Thus where the set of relevant cases are close to the input case, retrieval is fast while where the relevant cases are far away, retrieval takes longer. Search begins at the lowest level of index and proceeds step-wise through higher levels gathering up cases until the required number of cases to be retrieved is acquired. Figure 1 depicts the difference between a uniform retrieval approach and the level based retrieval approach. In uniform retrieval (Figure 1 A), retrieval time is dependent on the total volume of cases but not on distance between the input case and the target cases. However, in level based retrieval as given in Figure 1 B, if only 3 cases are required, search will halt after the inner most circle. If 5 cases are required, search will proceed to the next level and if 6 cases to the outermost circle. We now introduce a few concepts that define our approach.

Lenience is the notion, *Expand the search space only to the extent to which you are willing to or need to tolerate mismatches between the query and the cases in the case-base*. In other words the tolerance for mismatches defines the search space around the query. A low tolerance leads to a small neighbourhood and a small search space while a high tolerance leads to an expanded neighbourhood.

Level of Lenience (LL): *A class or extent of mismatches tolerated between cases in a case-base C* for an attribute which defines a search space.* It is important to understand that the LL is **not** a thresholding function that given a set of retrieved cases, selects the cases using a similarity threshold. Instead each Level of Lenience entails a neighbourhood of cases around a query that will be searched. Lenience is not confined to expanding a neighbourhood by relaxing the *bounds* of the search but could change the *form* of the index.

We have enforced an important property of the LL approach which we call the *Subsumption Criterion*: The cases returned at a higher level of lenience will be a superset of the cases returned at a lower level. A case that matches at a lower level will match at all higher levels. Thus in Figure 1 C, Case 1 matches at Level 1 itself and thus matches at all higher levels upto n, while Case 3 matches only at level n. This property implies that we can stop the retrieval at a level once the required number of cases are identified as we know the cases will also match at higher levels. Formally,

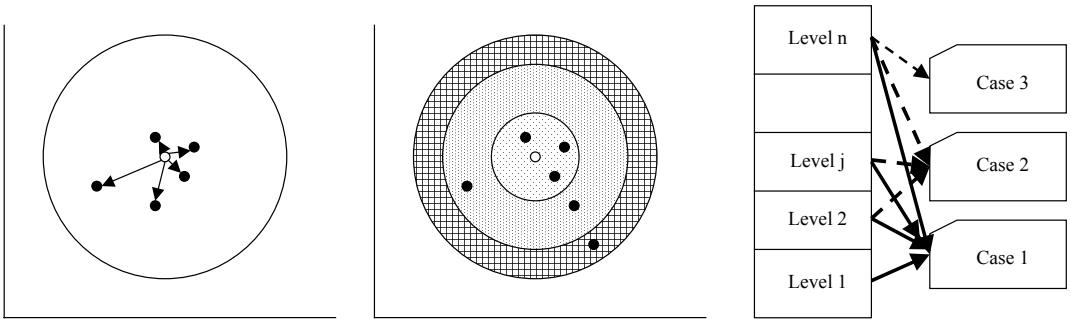

Figure 1: Uniform and level based retrieval

Let,

$n = Number\ of\ levels\ of\ lenience$

$S = \{s_1, s_2, ..., s_n\}$ *be the search spaces at* var *ious levels*

$T = \{t_1, t_2, ..., t_n\}$ *be the set of average retrieval times*

Then,

$\forall i, j, 1 \le i, j \le n, i < j, s_i, s_j \in S, t_i, t_j \in T$

$s_i \subseteq s_j$ *and* $t_i \le t_j$

Each level is assigned a weight that signifies the importance of matching at that level. The similarity measure between an input case q and a case r for an attribute is discussed later. The weighting at each level plays an important role since it has a direct influence on the field level similarity score of a case. Given field similarity thresholds, high weighting at lower levels and very low weighting at higher levels signals that the search is restrictive and wants only cases with a high likelihood of relevance to be retrieved. In a sense this may be called *Precision* centric. A less discriminative weighting signals that the intent is to get all possibly relevant cases and can thus be termed *Recall* centric. We define ***Degree of Lenience (DL)*** as

Let

$W_j = Weight\ at\ level\ j$

$MaxLevel = Maximum\ \ lenience\ \ level$

Then,

$$Degree\ \ Of\ Lenience\ \ \Delta = \frac{\sum_{j=1, MaxLevel-1} \frac{W_{j+1}}{W_j}}{MaxLevel - 1}$$

In general,

$\forall i, j, 1 \le i, j \le MaxLevel, i < j$

$W_i \ge W_j$

If all levels have equal weighting DL becomes 1 while a differentiated weighting will have low DL. The combination of LL and DL enables us to offer **Search Templates** where a Search Template is a combination of LL and DL suitable for a type of search. Users use the search template that best fits their requirement. While one template may offer two levels of lenience with a very low degree of lenience, suitable for high precision searches, another may offer four levels with a high degree of lenience, suitable for recall centric search. We demonstrate these ideas in the domain of name and address search in DA.

Matching over Names and Addresses

Names and addresses do not conform to standard linguistic forms. Thus traditional methods of handling inexact string matches such as stemming or trigrams are not blindly applicable. The volume of subscriber records (i.e. cases) in DA systems is often huge. Yet the system has to retrieve a small set of relevant cases within time bounds.

Meeting these challenges was the motivation behind our approach to name and address matching meant for DA systems where people call a number to get assistance. The essence of the approach is a pipelined strategy that goes step-wise from the least lenient search level to increasingly lenient levels.

We define four Levels of Lenience for names and addresses. (a) *The Token Level* : Each name or address is broken into a set of words after some transformations on the string. For example: The name John Jacob gives the two token set {John, Jacob}. (b) *Strict Phonetic Code* (SPC): The Strict Phonetic Code is a modification of the Metaphone algorithm (Binstock and Rex 1995) to encode English words phonetically. Matches at this level can recognise that "Rajeev" is similar to "Rajiv". (c) *Relaxed Phonetic Code (RPC):* The Relaxed Phonetic Code is a modification of the Soundex algorithm. It is similar to the Strict Phonetic Code technique, but is more forgiving of errors This can recognise that "Aswini" is similar to "Ashwinee" (d) *Gram level* : Here the Relaxed Phonetic Code is broken into a set of bigrams. It helps the algorithm to recover from insertion / deletion of a phonetically significant consonant or substitution of a phonetically significant consonant by another of a different phonetic category. This can recognise that "Ashirwad" is similar to "Asharswad".

Names and addresses also embed numeric information (such as house / street numbers and zip codes). We follow a lenience based strategy even for such numeric matching but do not discuss them in this paper for reasons of space. We also have inter-field matching strategies for handling high frequency tokens in names (common Indian surnames such as "Shah" or "Rao") where retrieval might stop and skip to another field to reduce the search space but which we do not expand on in this paper.

Clustering (Kaufmann and Rousseeuw 1990) is implemented at all levels of lenience. However unlike traditional clustering where the global similarity between strings is used as the distance measure, we use cluster prototypes at all levels. Thus at the first level of clustering, each token acts as an index to the set of cases that contain this token. The clusters of cases formed at this level are the smallest. At the second level, the SPC of each token is used to index the set of cases that have at least one token that maps onto that SPC. The third level of clustering uses RPC to form the clusters and so forth. Once these clusters are established, the search process traverses these clusters at successive levels in a stepwise fashion during retrieval.

At the Token, SPC and RPC levels the match between a token in query and target is exact and the similarity score is either 0 or 1. At the gram level the similarity score is based on the cardinality of intersection of query token grams and target token grams divided by the cardinality of the union and is thus over the interval [0,1].

Both query and target consist of many tokens. The field similarity computation is a function of the similarity scores and weight of each token at it's lowest level of match. Formally,

$$sim(q,r) = \frac{\sum_{i=1,N_r} W_{mi} \times fsim(q,r,m,i)}{W_1 \times (N_r + N_q) - \sum_{i=1,N_r} W_m \times fsim(q,r,m,i)}$$

where,

N_r, N_q = *Number of tokens in t* arg *et and query*

$fsim(q,r,m,i)$ = *Similarity score of ith token at lowest level it matches* $\in [0,1]$

W_1 = *Weight for matching at lowest level*

W_{mi} = *Weight at lowest level ith token matches*

We define four Search Templates where a Search Template signifies how search is to be carried out over all tokens over the various levels of lenience defined.

A search template P is defined as the tuple,

$P = \{L, W, O, f, k\}$ *where*,

$L = \{l_1, l_2, ..., l_n\}$ *are the levels of lenience for P*

$W = \{w_1, w_2, ..., w_n\}$ *are the weights at* var *ious levels*

O = *A process for traver* sin *g L for all tokens*

f = *Field similarity threshold* $\in [0,1]$

k = *Number of cases to be retrieved* $\in \{[1, N], u, *\}$

i.e k can be a number, *a user specified value or no* lim *it*

O is a procedure for processing all tokens over the various levels defined in this template including the retrieval termination condition. The 4 Search Templates provided are:

Exact Search: {{Token}, {4}, {Breadth, until k}, {0.3}, {u}}. Exact uses only the lowest LL with weight for matching at that level as 4, processes all tokens through the one level of lenience defined, the field similarity threshold is 30% and the user specifies the number of cases to be retrieved.

Slam Search: {{Token, SPC, RPC}, {4,3,2}, {Breadth, one token match at Token level}, {0.3}, {1}}. Slam uses only 3 levels of search with weights of 4, 3 and 2 (thus DL is 0.7). It proceeds level wise for all tokens, has the filter condition that at least one of the tokens in the query must match a token in the target at the token level and will retrieve only the topmost case. It provides a high precision retrieval where the target is expected to lie very close to the query

Simple Search: {{Token, SPC, RPC}, {4,3,2}, {breadth, one token match at Token level}, {0.3}, {u (10)}}. Simple is identical to Slam except that in Simple, the number of records to be retrieved is user specified (default 10). Simple also provides a high precision retrieval but will however take longer to execute than Slam if u > 1.

Advanced Search: {{Token, SPC, RPC, Gram}, {w_1 (4), w_2 (3.5), w_3 (3), w_4 (2.5)}, {Depth}, {0.3}, {u (10)}}. Advanced search provides 4 levels of lenience, user specified DL (default-0.86), processes each token through all levels, no filter conditions and user specified k (default 10). Advanced provides a high recall, expansive search that searches over all levels. However, Advanced also takes more time on the average than Simple search.

The overall process of matching for a field for all tokens takes place as follows: After preprocessing of the tokens, control passes to O for the search template chosen. The process for simple search is: The cluster prototypes of Level 1 are checked and the short list of all cases where at least one token matches is created. This is the base set. The scores of the matched tokens are computed. The non-matching tokens are then passed to the next higher level where their SPC codes are checked against level 2 cluster prototypes. Only cases already in the base set are considered. Again the scores of the matched tokens are computed and added to the previous scores. The non-matching tokens are passed to the RPC level. Once all levels are traversed, the overall field similarity of the cases in the base set with respect to the query is computed and the field similarity threshold applied to get the final list of short listed cases.

While this paper focusses on local similarity measures, we should note that we use a variation of the k nearest neighbour algorithm for computing global similarity. Details on the global similarity algorithm can be found in (Balaraman, Chakraborti and Vattam 2000).

Implementation and Results

The test version of the system was first implemented as a Visual C++ DLL that used ODBC to access Oracle V8 which stored the case-base. The results discussed below in Figure 2 and Table 1 overleaf are from this test version operating on a set of 50,000 subscriber records. The subscriber record set was also 'salted' with records which were distortions of existing records in order to test the efficacy and power of the various search templates.

The graph on the left in Figure 2 compares the speeds of Simple with Advanced search (with k set to 10 for each). The speeds of Exact and Slam are not depicted because they are so fast they lie nearly on the x-axis in the graph (Exact averages sub 0.01 seconds while Slam averages 0.04 seconds for 50,000 cases). Simple search takes sub 0.2 seconds even for 50,000 cases while Advanced takes nearly a second. But as Table 1 shows, Advanced is able to handle more distortions than Simple. Of the 4 distorted cases displayed, with a field similarity threshold of 30%, Exact would retrieve only the second record, Slam the first, Simple the first two and Advanced all four. Where the target is close to the query, Slam and Simple perform effectively while Advanced is required for phonetically significant structural distortions. The graph on the right shows the differences between a low DL and a high DL approach in Advanced Search.

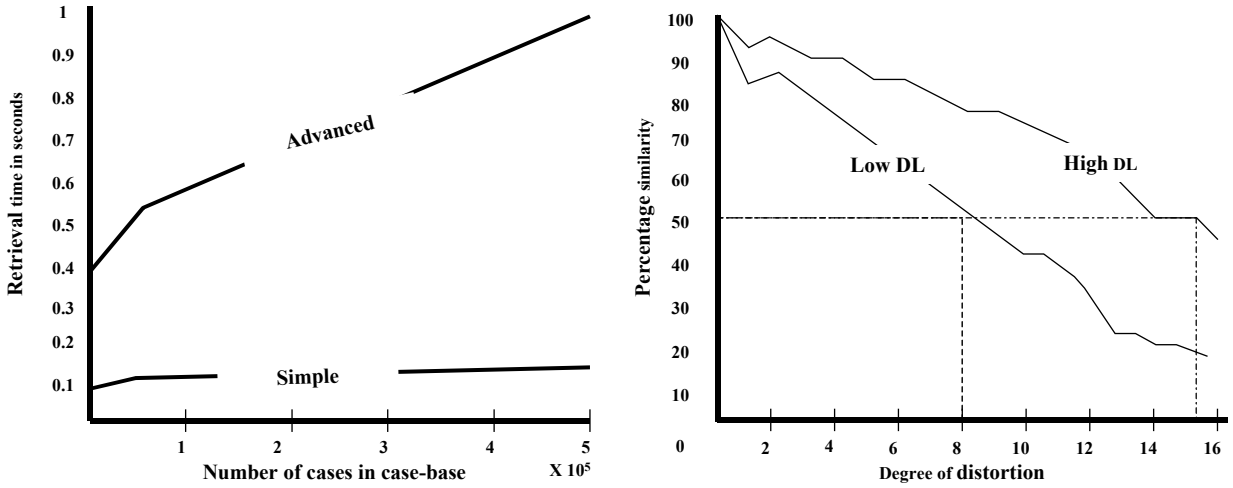

Figure 2: Simple Vs Advanced Search and Advanced Search Tuning

Query String	Similarity of all templates				Advanced Search Retrieval Results Normal font token – Match at exact *Italicised* token – Match at SPC / RPC **Bold** token – Match at gram
	Exact	Slam	Simple	Advanced	
Datta Niwas 1019/2 Deep Bglw Chow	10	50	50	78	*Dutta* Niwas *1019-2 Dip Buglw* **Chaw**
	33	33	33	70	**Dartta** Niwas 1019/2 Deep **Bnglw Chowk**
	10	17	17	60	**Dartta** Niwas *1019-2* **Deelp Bnglw Chowk**
	0	0	0	52	**Dartta** *Neewas 1019-2* **Deelp Bnglw Chowk**

Table 1: Retrievals with Varying Distortion

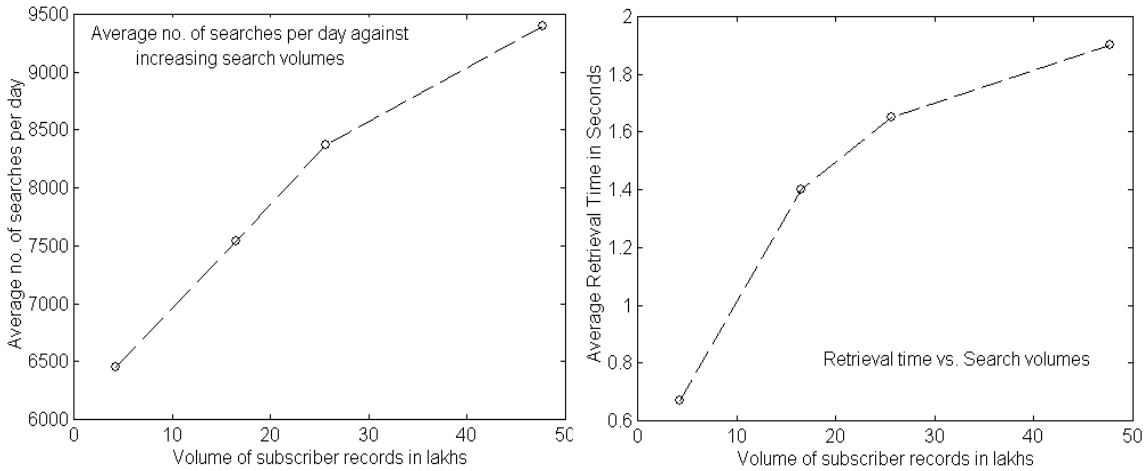

Figure 3: Production version results

The former reduces to 50% similarity with just 8% distortion while the latter tolerates 16% distortion for the same similarity score.

The production version of the system has been implemented and is live at a leading telecommunications service provider in India. The system provides directory assistance on a volume of nearly 5 million subscribers with over 100 concurrent users. The client system uses three Levels of Lenience of Exact, SPC and RPC and three Search Templates of Exact, Slam and Simple searches.

Figure 3 provides details of the system size, growth and performance. Search times averaged over all search templates is sub 2 seconds even for 5 million records. While the figure does not provide this detail, the average retrieval times for different Search Templates were, Exact 0.9 seconds, Slam 2.1 seconds and Simple 5.5 seconds. On average, 80% of the searches used Exact, 19% used Slam and 1% used Simple. Thus around 20% of searches required inexact matching.

Relevant Work

Several case-based systems use multiple indexes on the case-base such as in (Gordon and Domeshek 1995). However in these systems the multiple indexes are used to provide different views on the case-base as required by different users. In contrast, in our approach, the indexes are provided primarily to provide a time performance advantage when operating over very large case-bases with strict time performance constraints. Additionally, the indexes in our approach satisfy the criterion that cases retrieved at an index level will be supersets of the cases at a lower index level. This enables search to proceed in a step-wise fashion on a need to basis from low levels of lenience to increasing levels.

In (Schumacher and Bergmann 2000), a similarity system is built directly on top of a database with query relaxation taking place if the requisite cases k are not retrieved. The concept of relaxation step is analogous to our concept of level of lenience and query relaxation satisfies our subsumption criterion. Our approach too is targeted at recommender domains with the cases stored in a conventional DBMS. However our approach differs from their approach in at least four ways: a) our query relaxation steps are precomputed in advance and the case-base is indexed at the level of each relaxation step. b) relaxation may be not just by increasing the *bounds* of the search but by using different *forms* of indexes c) the concept of degree of lenience, where different weightages are given to matches at different levels and d) the concept of a Search Template.

The directory assistance system developed by (Kawabe, Fukumura et al 1997) carries out a sequence of operations on each query of morphological analysis, semantic analysis, intention understanding and building name analysis to derive a code for an address which is used as a single index. While details are sparse, it does not appear that search conditions are relaxed or tightened based on query proximity to existing records as carried out in our approach. Additionally our approach differs from this system and from commercial tools like Scansoft (Scansoft), in that we do not use pre-compiled geography-specific name or address dictionaries for our retrieval.

While the concepts in this paper were demonstrated in the domain of name and address search, the same approach is applicable to other applications with similar performance requirements such as Job or Realtor search. Prototypes have been developed in these domains and are showing promising results. These domains require use of data types like ontologies, ordered lists and numbers for which lenience based algorithms have been developed.

We are currently working on extending the concept of lenience to global similarity matching. Global similarity will have global search templates which combine search templates of different attributes. We have realised the need for query planning in inexact retrieval especially when operating over high volume, high dimensionality repositories.

Acknowledgements

The implementation and ideas presented in this paper are the joint effort of the authors and the Consult development team, Pune, India.

References

Gonzalez, A.J.; Laureano-Ortiz, R. 1992. A Case-Based Reasoning approach to Real Estate property appraisal. *Expert systems with Applications* 4(2):229 – 246.

O'Roarty, B.; Patterson, D.; McGreal, S.; and Adair, A. 1997. A Case-Based Reasoning Approach to the selection of comparable evidence for retail rent determination.. *Expert Systems with Applications*, 12(4):417 – 428.

Binstock, A; and Rex, J. 1995. Metaphone: A Modern Soundex. In *Practical Algorithms For Programmers*. 160-169, Addison-Wesley.

Kaufmann, L.; Rousseeuw, P. 1990. *Finding groups in data – An introduction to Cluster analysis.* Wiley-Interscience Publication.

Gordon, A.; Domeshek, E. 1995 Retrieval Interfaces for Video Databases, In AI Applications in Knowledge Navigation and Retrieval, Working notes of the AAAI-95 Fall Symposium, Massachusetts Institute of Technology, Cambridge, MA.

Schumacher, J.; Bergmann, R. 2000. An Efficient Approach to Similarity-Based Retrieval on Top of Relational Databases. In *Advances in Case-Based Reasoning, Proceedings of the 5th European Workshop*, EWCBR-2000, 273-284, LNAI 1898, Springer.

Balaraman, V.; Chakraborti, S.; Vattam, S. 2000. Using CBR Inexact Search for Intelligent Retrieval of Data. In Proceedings of the International Conference KBCS 2000. India. 119-130. National Centre for Software Technology, Mumbai, India.

Kawabe, H.; Fukumura, Y.; Mutoh, N.; Karasawa, H.; Iwase, S. 1997. An Intelligent Directory-Assistance System Using Natural Language Processing and Mapping, In Proceedings of the International Conference on Tools in Artificial Intelligence, 486-487, California, IEEE Computer Society

Scansoft product website.
http://www.scansoft.com/directoryassistance/features.asp