MIRA: Multilingual Information Processing on Relational Architecture

A. Kumaran

Department of Computer Science and Automation,
Indian Institute of Science, Bangalore, India
kumaran@csa.iisc.ernet.in

Abstract. In today's global village, it is critical that the key information tools, such as web search engines, *e-Commerce* portals and *e-Governance*, work across multiple natural languages, seamlessly. We propose a new flexible architecture – Multilingual Information processing on Relational Architecture (MIRA) – that supports the multilingual processing functionality of the primary storage mechanism for such deployments – the relational database systems, effectively and efficiently. We propose new linguistic matching operators that enhances the standard lexicographic matching of database systems into *phonetic* and *semantic* domains. We further show that the performance of the systems may be made *language-neutral*. Our proposed architecture is based on standards and hence amenable for easy implementation in any type of query processing and information retrieval systems. In this paper, we present our approach to implement the above architecture and outline the host of research issues that are opened up due to the inherently fuzzy nature of the alternative matching semantics.

1 Introduction

In an increasingly multilingual digital world[1], the key information and commerce applications, such as *e-Commerce* portals, digital libraries, search engines etc., must work across multiple natural languages, seamlessly. A critical requirement to achieve this goal is that the principal underlying data source – relational database management systems – should manage multilingual data effectively and efficiently. Our proposal, Multilingual Information processing on Relational Architecture (MIRA), attempts to enhance the relational database systems with multilingual features and to make the query performance nearly *language neutral*. Further, our proposed architecture is amenable for easy implementation in any type of query processing and information retrieval systems.

Specifically, we propose multilingual operators that extend and complement the standard lexicographic matching operator, to match text strings across languages based on enhanced matching semantics. We propose a *phonetic* matching operator that matches proper names, after transforming them to equivalent *phonemic* strings, and a *semantic* matching operator that matches attributes based on their *meanings*, transformed using

[1] Currently, two-thirds of Internet users are non-native English speakers [1] and it is predicted that the majority of web-data will be multilingual by 2010 [2].

W. Lindner et al. (Eds.): EDBT 2004 Workshops, LNCS 3268, pp. 12–23, 2004.

ontological hierarchies. In both cases, the performance of the operators is shown to be at a level acceptable for online user interaction. Further, to make the performance of queries on multilingual data comparable to monolingual processing, we propose a new compressed storage format that results in a near *language-neutral* performance when implemented on commercial database systems.

The alternative semantics for the matching operators and the inherently fuzzy nature of such matching opened up several interesting issues that may be possible extensions of our current research. We are also expanding the scope of our application domains, to test the viability of our multilingual architecture.

2 A Sample Multilingual Application

Consider a hypothetical e-Commerce application – *Books.com* that sells books across the globe, with a sample product catalog in multiple languages as shown in Figure 1. The product catalog shown may be considered as a logical view assembled from data from several databases (each aligned with the local language needs), but searchable in a unified manner for multilingual users.

Author	Author_FN	Title	Price	Category	Language
Durant	Will/Ariel	History of Civilization	$ 149.00	History	English
நேரு	ஜவஹர்லால்	ஆசிய ஜோதி	INR 250	சரித்திரம்	Tamil
Adams	Laurie S.	Arte Di Rinascita Italiana	€ 75.00	Arti Fini	Italian
Lebrun	François	L'Histoire De La France	€ 19.95	Histoire	French
بهنسي ، د	عفيف	الحضارة عبر التاريخ	SAR 95	معماريّ	Arabic
Gilderhus	Mark T.	History and Historians	£ 35.00	Historiography	English
नेहरु	जवाहरलाल	भारत एक खोज	INR 175	इतिहास	Hindi
Σαρρη	Κατερινα	Παιχνίδια στο Πιάνο	€ 12.00	Μουσική	Greek
Nehru	Jawaharlal	Letters to My Daughter	£ 15.00	Autobiography	English

Fig. 1. Hypothetical *Books.com* Catalog

2.1 Multilingual Name Searches

In this environment, suppose a user wants to search for the works of an author in all (or a specified set of) languages. The SQL:1999 compliant query requiring specification of the authors name in several languages is undesirable, due to requirement of lexical resources in each of the languages and high error levels in data input even when working on mono-lingual data[2]. We propose a simple query syntax, as shown in Figure 2, that takes input name in one language, namely `English`, but returns all *phonemically* equivalent names in the user-specified set of languages, namely, `English`, `Hindi`, `Arabic` and `Tamil`.

A sample phonetic query and the corresponding answer set, when issued on *Books.com*, are given in Figure 2. The returned tuples have in Author column the multi-lexical strings that are *phonemically close* to the query string in English, namely, `Nehru`.

[2] The error rate for name attributes in English is estimated to be approximately 3% [9].

The specification of ALL for the list of languages would have brought all records containing author names that are phonetically equivalent to Nehru, irrespective of the languages. The Threshold parameter specified in the query determines the quality of matches, as described later in the paper.

```
SELECT Author,Title,Language FROM Books
WHERE Author LexEQUAL 'Nehru' Threshold 0.25
IN { English, Hindi, Arabic, Tamil }
```

Author	Title	Language
Nehru	Letters to My Daughter	English
நேரு	ஆசிய ஜோதி	Tamil
नेहरु	भारत एक खोज	Hindi

Fig. 2. A Sample LexEQUAL Query and Result Set

We refer *matching on multilexical text strings, based on their phonemic equivalence* as **Multilexical Phonemic Matching**. Though restricted to proper names, such matching represent *a significant part of the user query strings in text databases and search engines, as proper and generic names constitute a fifth of normal corpora* [9].

2.2 Multilingual Concept Searches

Consider the query to retrieve all History books in *Books.com*, in a set of languages of users choice. The current SQL:1999 compliant query, having the selection condition as Category = "History" would return only those books that have *Category* as History, in English. A multilingual user may be served better if all the History books in all the languages (or in a set of languages specified by her) are returned. A simple SQL query, as given in Figure 3, and the corresponding result set may be desirable.

```
SELECT Author,Title,Category FROM Books
WHERE Category SemEQUAL 'History'
IN { English, Hindi, French, Tamil }
```

Author	Title	Category
Durant	History of Civilization	History
Gilderhus	History and Historians	Historiography
Lebrun	L'Histoire De La France	Histoire
Nehru	Letters to My Daughter	Autobiography
நேரு	ஆசிய ஜோதி	சரித்திரம்
नेहरु	भारत एक खोज	इतिहास

Fig. 3. A Sample SemEQUAL Query and Result Set

The output contains all books that have their category values that are semantically equivalent to History in English. Note that in addition to all books with *Category*

having a value equivalent to `History`, the categories that are *subsumed* by `History`[3] are also retrieved. We refer *matching text strings based on their generalized meanings, irrespective of the languages*, as **Multilingual Semantic Matching**.

It should be specially noted here that though our solution methodology is designed for matching multilingual strings, it is equally applicable for extending the standard matching semantics of mono-lingual text strings. For example, the LexEQUAL operator may be used for matching the English name `Catherine` and all its variations, such as `Kathrin` and `Katerina`. Similarly, the SemEQUAL operator, may be used for matching `Disk Drive` with `Computer Storage Devices`.

3 MIRA Implementation Strategy

In this section, we outline our strategy for implementing the MIRA architecture. We explain the ontology of text data in relational systems and show how we define the semantics for the new operators for *phonemic* and *semantic* matching of multilingual text data.

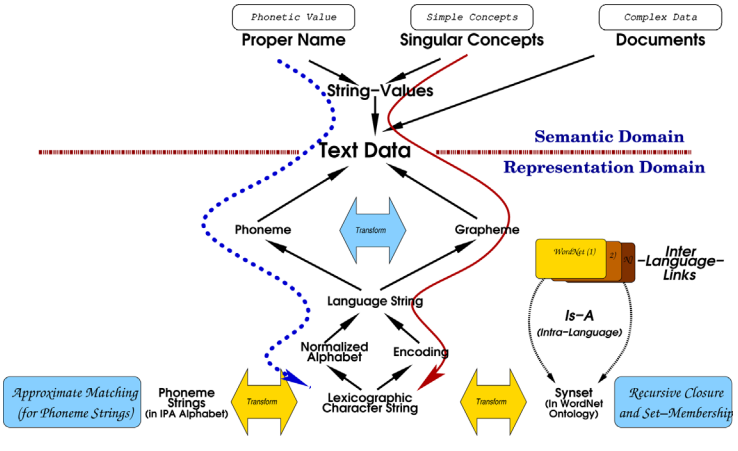

Fig. 4. Ontology for Text Data

Our view of storage and semantics of textual information in databases is shown in Figure 4. The semantics of *what* gets stored is sketched in the top part of the figure, and *how* the text data is stored is outlined by the lower half of the figure. Multilingual text strings are stored in a normalized alphabet (such as *English, Hindi, Arabic, Chinese*, etc.) and orthogonally in a specific encoding (such as *ASCII, Unicode*, etc.). Semantically, the text attributes may represent a wide variety of information, from simple strings to full documents; however, we consider only two specific type of attributes; first, those that store proper names, whose value is primarily in the vocalization of the name, tagged as

[3] `Historiography` (the study of of history writing and written histories) and `Autobiography` are considered as specialized branches of `History` itself.

Proper Names in the figure above. Second, those that lend themselves to be described using ontological hierarchies, tagged as *Singular Concepts* above. We broadly classify all other attributes as Documents which may require more sophisticated Natural Language Processing algorithms that process the documents on complex semantics.

3.1 Phonemic Matching Strategy

In this section we briefly sketch our phonetic matching approach that extends earlier works in monolingual world [16] to matching of multilingual names. We further enhance the performance of such matching by defining a phonemic index based on the classic *Soundex* algorithm [10] and by adopting *q-gram* techniques [7] that has been successfully used in approximate matching of monolingual names to multilingual world. Interested readers are referred to [13–15] for details of the matching algorithm and its performance.

LexEQUAL (S_l, S_r, e)
Input: Strings S_l, S_r, *Error Threshold,* e
Languages with TTP transformations, $\mathcal{S}_\mathcal{L}$
1. $L_l \leftarrow$ Language of S_l;$L_r \leftarrow$ Language of S_r;
2. **if** $L_l \in \mathcal{S}_\mathcal{L}$ and $L_r \in \mathcal{S}_\mathcal{L}$ **then**
3. $T_l \leftarrow$transform(S_l,L_l); $T_r \leftarrow$transform(S_r,L_r);
4. $Smaller \leftarrow (\mid T_l \mid \leq \mid T_r \mid ? \mid T_l \mid : \mid T_r \mid)$;
5. **if** editdistance$(T_l, T_r) \leq (e * Smaller)$
6. **then return** TRUE **else return** FALSE;
7. **else return** NORESOURCE;

editdistance(S_L, S_R)
Input: String S_L, String S_R
Output: Edit-distance k
1. $L_l \leftarrow \mid S_L \mid$; $L_r \leftarrow \mid S_R \mid$;
2. Create *DistMatrix*$[L_l, L_r]$ and initialize to $Zero$;
3. **for** i **from** 0 **to** L_l **do** *DistMatrix*$[i, 0] \leftarrow i$;
4. **for** j **from** 0 **to** L_r **do** *DistMatrix*$[0, j] \leftarrow j$;
5. **for** i **from** 1 **to** L_l **do**
6. **for** j **from** 1 **to** L_r **do**
7. *DistMatrix*$[i, j] \leftarrow$ **Min** $\left\{ \begin{array}{c} \textit{DistMatrix}[i-1, j] + \textit{InsCost}(S_{L_i}) \\ \textit{DistMatrix}[i-1, j-1] + \textit{SubCost}(S_{R_j}, S_{L_i}) \\ \textit{DistMatrix}[i, j-1] + \textit{DelCost}(S_{R_j}) \end{array} \right\}$
8. **return** *DistMatrix*$[L_l, L_r]$;

Fig. 5. The LexEQUAL Algorithm

We propose a *phonemic* matching strategy (shown as dotted line in Figure 4) in LexEQUAL operator, as follows: First, the multilingual text strings are transformed to

their equivalent *phonemic* representations in *International Phonetic Alphabet* (*IPA*)[4] [3], obtained using standard *text-to-phoneme (TTP)* converters. The phoneme strings are stored in the Unicode [4] encoding format, as specified by Unicode Consortium, using basic *Latin* and *IPA supplement* code charts. The resulting phoneme strings represent a normalized form of proper names across languages, thus providing a means of comparison. Further, when the text data is stored in multiple scripts, this may be the *only* means of comparing them. Since the phoneme sets of two languages are seldom identical, we employ approximate matching techniques to match the phoneme strings. Thus, the multilexical comparisons are *inherently fuzzy*, making it only possible to produce a likely, but not perfect, set of answers with respect to the user's intentions.

In the algorithm shown in Figure 5, the LexEQUAL operator accepts two *to-be-compared* multilingual text strings and a *User Match Threshold* parameter that determines the quality of match, as inputs. The strings are transformed to their equivalent phonemic strings in IPA [3] alphabet by the Transform function, implemented using standard *TTP* converters. The editdistance function computes the traditional *Levenshtein* edit distance or a modified distance metric, as appropriate. The value for the user match threshold parameter may be fixed by application administrators, depending on the domain and the application requirements.

LexEQUAL Match Quality. While the performance of above algorithm can be optimized, we emphasize that the ideal parameters will depend on the data set and the domain of interest. The *User Match Threshold* and *Cost Matrix* (*the replacement cost between a given pair of characters*) parameters may be tuned at user or application level, based on the characteristics of the domain. We detail our experiments and a methodology for tuning the parameters for optimal matching in [15].

LexEQUAL Operator Performance. Since the approximate matching implemented as UDF affects the query run-times adversely, we outline two different techniques for improving the performance of the query processing – the *Q-Gram Filters* and an *Approximate Phonetic Index*. Both these techniques cheaply return a set of candidate strings, which are further processed using the expensive UDF calls to weed out the *false-positives*. Our experimental results show that these techniques vastly improve the performance of phonetic matching.

3.2 Semantic Matching Strategy

Our strategy for matching multilingual data based on *semantics* in SemEQUAL operator is as shown in Figure 6. First, we convert the query string to a set of *concepts* using a standard linguistic resources, such as WordNet [5]. The WordNet is a *lexico-semantic* database that provides, the context for all noun word-forms in standard taxonomical hierarchies covering all concepts expressible in a language. We propose to leverage the rich semantic hierarchies available in WordNet and match two different word forms, based on the concept that they map on to in WordNet's taxonomic hierarchy. Further, the matching

[4] IPA provides a complete set of phonemes of all the world's languages, thus providing a common representation for the vocalization of the proper name.

may be on specializations of the *meaning* of the query string, using semantic closure[5] computed using WordNet. In addition to English WordNet, there are several initiatives such as *Euro-WordNet* and *Indo-WordNet* around the world [6], to interlink concepts of WordNets in different languages. Once the multilingual word forms are mapped onto semantic primitives using WordNet of the appropriate language, the resulting semantic primitives may be compared for equivalence, generalization or specialization based on the common concept hierarchy between the languages.

SemEQUAL ($String_{Data}$, $String_{Query}$, $\mathcal{T}_{\mathcal{L}}$)
Input: Strings $String_{Data}$, $String_{Query}$
Set of Target Languages $\mathcal{T}_{\mathcal{L}}$
Output: TRUE or FALSE
[*Optional*] Gloss of Matched Synset
1. (L_D,L_Q) ← LangOf ($String_{Data}$, $String_{Query}$);
2. ($\mathcal{W}_{\mathcal{D}}$,$\mathcal{W}_{\mathcal{Q}}$) ← WordNetOf ($L_D$,$L_Q$);
3. $\mathcal{S}_{\mathcal{D}}$ ← Synset of $String_{Data}$ in $\mathcal{W}_{\mathcal{D}}$; $\mathcal{S}_{\mathcal{Q}}$ ← Synset of $String_{Query}$ in $\mathcal{W}_{\mathcal{Q}}$;
4. $\mathcal{TC}_{\mathcal{Q}}$ ← TransitiveClosure($\mathcal{S}_{\mathcal{Q}}$, $\mathcal{T}_{\mathcal{L}}$);
5. **if** $\mathcal{TC}_{\mathcal{Q}} \cap \mathcal{S}_{\mathcal{D}}$ **is not empty then**
return TRUE **else return** FALSE;
6. [Opt.] **return** Gloss of the Matched Synset;

TransitiveClosure (S, $\mathcal{T}_{\mathcal{L}}$)
Input: String S, Target Language Set $\mathcal{T}_{\mathcal{L}}$
Output: The specializations of $\mathcal{S}$
1. L_S ← Language of String S;
2. $\mathcal{W}_{\mathcal{L}}$ ← WordNet of Language L_S;
3. $\mathcal{S} \leftarrow \mathcal{S}_{\mathcal{C}} \leftarrow$ Synsets of S in $\mathcal{W}_{\mathcal{L}}$; $\mathcal{S}_{\mathcal{N}} \leftarrow \phi$;
4. **repeat until no change in** $\mathcal{S}$:
5. **for every element** s **in** $\mathcal{S}_{\mathcal{C}}$
6. $\mathcal{S}_{\mathcal{N}} \leftarrow \mathcal{S}_{\mathcal{N}} \cup$ hypernyms of s
$\cup$ Synsets linked to s through
InterLangIndex to $L \epsilon \mathcal{T}_{\mathcal{L}}$
not yet traversed to;
7. $\mathcal{S} \leftarrow \mathcal{S} \cup \mathcal{S}_{\mathcal{N}}$; $S_C \leftarrow \mathcal{S}_{\mathcal{N}}$; $\mathcal{S}_{\mathcal{N}} \leftarrow \phi$;
8. **return** $\mathcal{S}$

Fig. 6. The SemEQUAL Algorithm

The SemEQUAL function takes as input, strings $String_{Data}$ and $String_{Query}$. The transitive closures are computed by the TransitiveClosure function, using `Is-A` relationships within a language and using `Inter-Language-Index` across languages. In the implementation, only the WordNets corresponding to the target languages specified in the query are traversed. Once the transitive closure is computed, set-processing

[5] The semantic closure of a concept in a taxonomic (or ontological) hierarchy, is the set of nodes reachable from a given node, by tracing the `parent-child` relationships.

routines are used for computing set-memberships. The output is TRUE if the specified matching condition is met. Since the query string, $String_{Query}$, may match on any one of the several synsets (which are possible semantics of the same word form), SemEQUAL may be made optionally to return the *Gloss* of the synset on which the $String_{Query}$ is matched.

The transitive closure function is implemented using the recursive SQL feature defined in SQL:1999 [8], which was found to be adequate *functionally* to implement the SemEQUAL operator. However, as expected, the cost of computing semantic closures was high. Since the linguistic ontological hierarchies are typically large (containing as much as 100,000 concepts), and since the domain-specific ontologies are typically much smaller than the WordNet ontology, our experiments with WordNet ontological hierarchies may provide a *worst-case* performance scenario for SemEQUAL matching.

SemEQUAL Operator Performance. We first analyzed the performance of SemEQUAL, expressed using standard SQL:1999 features, in relational database systems. A direct implementation on three commercial database systems indicates that supporting multilingual semantic processing is unacceptably slow. However, by tuning the schema and access structures to match the characteristics of WordNet, we are able to bring the response times down to *a few milliseconds*, which we expect to be sufficient for most applications. The details of our implementations and optimization techniques would be published in a forthcoming technical report.

3.3 General Multilingual Query Performance

While most commercial database systems support management of multilingual data, we found that the relative performance in handling multilingual data, compared with standard *Latin* based scripts was upto 300% slower. A comprehensive study of the differential performance of popular database management systems with respect to multilingual data is given in [12]. Worse, we found that the query optimizer's prediction accuracy differs substantially between them. We analyzed the parameters contributing to the slowdown and narrowed down the differential performance to primarily the storage size of the Unicode format and its effect on in-memory processing, and secondarily due to the Unicode-specific function call overheads.

To alleviate the primary problem, We propose Cuniform, a compressed format that is trivially convertible to Unicode, yet occupying equivalent storage space when the data is expressed in native ASCII-based scripts. Our initial experimental results with Cuniform indicate that it largely eliminates the performance degradation for multilingual scripts with small repertoires, and makes the performance of queries nearly *language-neutral*, for languages with small repertoire. Further, by partitioning the multilingual data in language specific tables, we may be able to achieve a higher performance than monolingual data, under certain assumptions on the distribution of data among different languages.

4 MIRA Implementation Architecture

Our proposed architecture for multilingual query processing is shown in Figure 7. The shaded boxes emphasize the new processing modules, and the iconized boxes represent resources (lexical or semantic) that are to be installed.

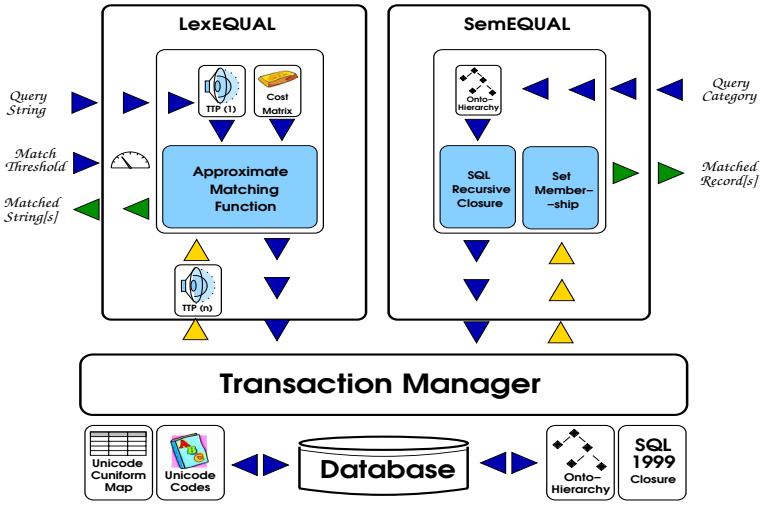

Fig. 7. MIRA Architecture

4.1 Design Goals for MIRA

We define the following design goals for the MIRA architecture, to implement the multilingual features and performance requirements in a usable, useful and scalable manner.

Relational Systems Oriented. Our focus is on relational database systems due to their popularity as data repositories for most operational data.
Attribute Data Oriented. Our architecture will focus on processing attribute-level data, for supporting multilingual keyword searches.
Standards Based. We rely on standard linguistic resources to promote uniformity and consistency across different information processing systems.
Light-Weight Processing Components. Our architecture will focus on OLTP environments, and hence light-weight components for handling text data.
Customizable Matching. The matching quality must be customizable by users, depending on the domain and application requirements.
Modular and Dynamic Architecture. The linguistic resources must be easily added, to make MIRA *language aware*, dynamically.

5 Conclusion and Future Research Issues

In our thesis, we propose an architecture for processing multilingual data transparently across languages, on the traditional information processing platforms, such as relational database management systems. A survey on the functionality and performance of the current systems indicate that the state-of-the-art falls short of these requirements on several counts, motivating our research on multi-lingual database systems.

From the efficiency perspective, we profiled in [12] the performance of standard relational operators (e.g. Select, Join) applied on multilingual data in commercial database systems. Our results showed that severe performance penalties may be incurred, upto about 300%, when compared against equivalent query processing on ASCII based data. We proposed efficient compressed storage format, Cuniform, to reduce these penalties and demonstrated that the query processing can be made nearly *language-neutral*.

From the functionality perspective, we introduced a new SQL multilingual operator called LexEQUAL [13–15], for *syntactic* matching of attribute data across languages. We confirmed the feasibility of our strategy by measuring the quality metrics, namely *Recall* and *Precision*, in matching a real, tagged multilingual data set. Further, we showed that the poor performance associated with the UDF implementation of approximate matching may be improved by orders of magnitude, by employing optimization techniques. We also proposed a new SQL operator – SemEQUAL, intended for matching multilingual text attribute data based on their meanings, leveraging the rich taxonomic hierarchies in cross-linked WordNets in different natural languages. Our experiments with WordNet on three commercial database systems, confirmed the utility of the SemEQUAL operator, but underscored the inefficiencies in computing transitive closure, an essential component for semantic matching. By tuning the storage and access structures to match the characteristics of resources in the linguistic domain, we speeded up the closure computation by 2 to 3 orders of magnitude – to *a few milliseconds* – making the operator viable for supporting user online query processing.

In summary, in the performance area, we have profiled and optimized the multilingual performance of popular database systems. In the functional area, we have defined two operators – namely, LexEQUAL for *phonetic* matching and SemEQUAL for *semantic* matching of multilingual attributes, and shown that they may be efficiently implemented on existing relational database management systems. We expect that such operators may effectively and efficiently complement the standard lexicographic matching, thereby representing a first step towards the ultimate objective of achieving complete multilingual functionality in database systems.

5.1 Research Issues

The following open issues are being addressed as a part of our current research.

Real-Life Application and Multilingual Performance Suites. We need to identify a *real-life* application that can benefit from the multilingual processing and establish user work-flows using the multilingual operators. Such an application may provide a test-bed for performance suites to calibrate and compare different database management systems on multilingual performance [11], along the lines of TPC benchmarks for OLTP applications.

Automatic Fine-tuning of Phonetic Match Quality. In LexEQUAL operator for phonetic matching, clearly the parameters for the best match quality depends on the phoneme set of the languages being considered and the requirements of the application domain. For example, a *Homeland Security* application may require tighter matches, where as a *Telephone Subscriber Search* application may be willing to tolerate much looser matches. We are currently automating the determination of optimal match parameters, based on user-defined training sets.

Approximate Indexes for Efficient Searches. Several approximate indexing methodologies offer search capability on pre-generated phonemic strings corresponding to names. We define the *Search Efficiency* of an index tree as the fraction of data elements in the database that were *examined*. The search efficiency indicates the effectiveness of the index structure in narrowing down the search. However, we find that all the approximate indexes are inefficient in searches, as shown in Figure 8. For example, about 75% of the strings in the database are retrieved as candidate matches for a user match threshold of `0.5`, while less than 1% of the database are real matches.

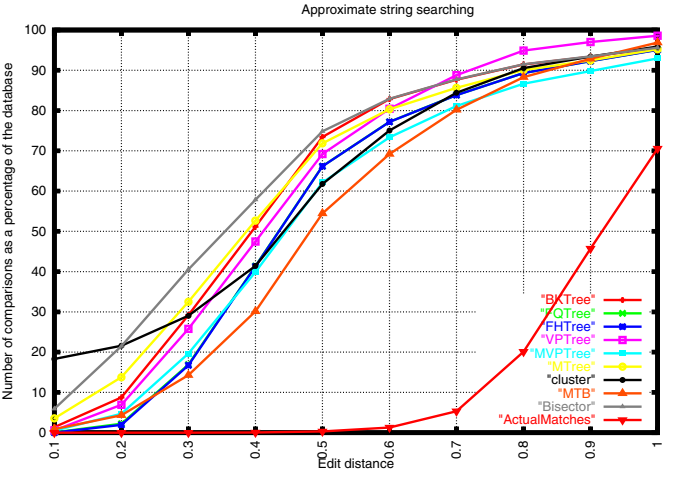

Fig. 8. Search Efficiency of Approximate Indexes

We are exploring the better partitioning and clustering techniques for building effective approximate indexes, to improve the search efficiency.

Domain-Specific Ontologies. While our performance experiments in semantic matching with WordNet taxonomic hierarchies had established performance characteristics of SemEQUAL, we expect the domain-specific ontologies to be more useful in semantic searching applications. Experiments (for performance and tightness) must be conducted using smaller and more precise domain specific ontologies, to ascertain the value of SemEQUAL operator.

References

1. The Computer Scope Limited. http://www.NUA.ie/Surveys.
2. The Web Fountain Project. http://www.almaden.ibm.com/WebFountain.
3. The International Phonetic Association. http://www.arts.gla.ac.uk/IPA/.
4. The Unicode Consortium. http://www.unicode.org.
5. The WordNet. http://www.cogsci.princeton.edu/~wn.
6. The Global WordNet Association. http://www.globalwordnet.org.
7. L. Gravano, P. Ipeirotis, H. Jagadish, N. Koudas, S. Muthukrishnan, and D. Srivastava. Approximate string joins in a database (almost) for free. In *Proc. of the 27th VLDB Conf., Rome, Italy*, 2001.
8. ISO/IEC. *Standard 9075-1-5:1999, Information Technology – Database Languages – SQL*. International Organization for Standardization, 1999.
9. D. Jurafskey and J. Martin. *Speech and Language Processing*. Pearson Education, 2000.
10. D. E. Knuth. *The Art of Computer Programming (Vol 3: Sorting and Searching)*. Addison–Wesley, Reading, Massachusetts, United States, 2nd edition, 1993.
11. A. Kumaran and J. R. Haritsa. On database support for multilingual environments. In *Proc. of the 13th IEEE Research Issues in Data Engineering Workshop (held in conjunction with 19th IEEE Intl. Conf. on Data Engineering), Bangalore/Hyderabad, India*, 2003.
12. A. Kumaran and J. R. Haritsa. On the costs of multilingualism in database systems. In *Proc. of the 29th VLDB Conf., Berlin, Germany*, 2003.
13. A. Kumaran and J. R. Haritsa. LexEQUAL: Multilexical matching operator in SQL. In *Proc. of the 23rd ACM SIGMOD Intl. Conf. on Management of Data, Paris, France*, 2004.
14. A. Kumaran and J. R. Haritsa. Supporting multilexical queries in SQL. In *Proc. of the 20th IEEE Intl. Conf. on Data Engineering, Boston, United States*, 2004.
15. A. Kumaran and J. R. Haritsa. Supporting multiscript matching in database systems. In *Proc. of the 9th Extending Database Technology Conf., Heraklion-Crete, Greece*, 2004.
16. J. Zobel and P. Dart. Phonetic string matching: Lessons from information retrieval. In *Proc. of 19th ACM SIGIR Conf., Zurich, Switzerland*, 1996.