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

1 Introduction

In today's global village, 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 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[1]. 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`.

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

Fig. 2. Sample LexEQUAL Query

The query given in Figure 2, when issued on *Books.com*, may retrieve a set of answers, as given in Figure 3. The returned tuples have in Author column the multilexical strings that are *phonemically close* to the query string in English, namely, `Nehru`. 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.

We refer to such matching as **Multilexical Phonemic Matching**, which allows *matching on multilexical text strings, based on their phonemic equivalence.* Though restricted to proper names, such searches 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* [5].

[1] The error rate for name attributes in English is estimated to be approximately 3% [5].

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

Fig. 3. LexEQUAL Query Result

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 of the form,

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

Fig. 4. Sample SemEQUAL Query

wherein the new operator, SemEQUAL, retrieves the set of tuples, as shown in Figure 5, would be desirable. 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`[2] are also retrieved, due to the ALL clause in the query.

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. 5. SemEQUAL Query Results

We refer to such matching as **Multilingual Semantic Matching**, which *matches multilingual text strings based on their meanings, irrespective of the languages of storage.*

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

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 `Data Storage Devices` and `Computer Peripherals`.

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.

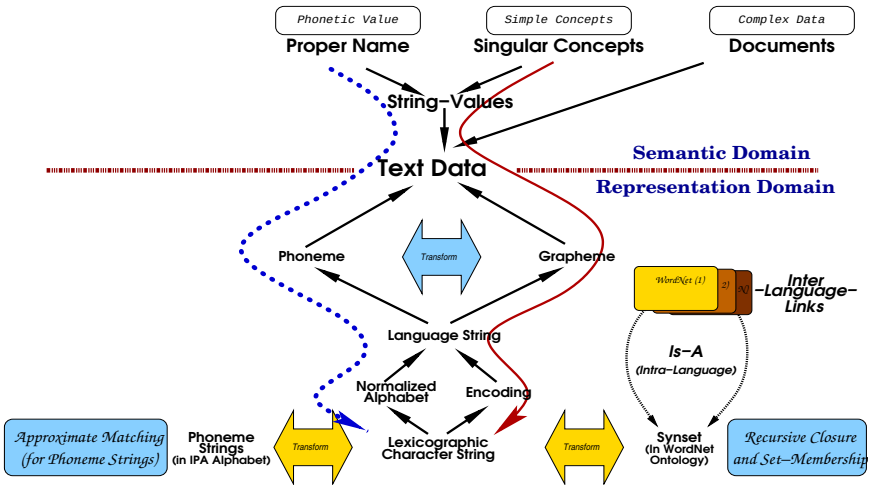

Fig. 6. Ontology for Text Data

Our view of storage and semantics of textual information in databases is shown in Figure 6. 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. 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 *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.

Multilingual text strings are stored in a normalized alphabet (such as *English, Hindi, Arabic, Chinese*, etc.) and in a specific encoding (such as *ASCII, Unicode*, etc.). Currently, *all* matching in database systems happens only as lexicographic matching. Such matching is not possible when the strings are in different scripts.

3.1 Phonemic Matching Strategy

In this section we briefly sketch our phonetic matching approach that extends earlier works in monolingual world [12] to matching of multilingual names. We further enhance the performance of such matching by defining a phonemic index based on the classic *Soundex* algorithm [6] and by adopting *q-gram* techniques [4] that has been successfully used in approximate matching of monolingual names to multilingual world. Interested readers are referred to [9] for details of the matching algorithm and its performance.

We propose a *phonemic* matching strategy (shown as dotted line in Figure 6) in LexEQUAL operator, as follows: First, the multilingual text strings are transformed to their equivalent *phonemic* representations in *International Phonetic Alphabet (IPA)*[3] [3], obtained using standard *text-to-phoneme (TTP)* converters. The phoneme strings are stored in the Unicode [10] encoding format, as Unicode has allocated character code for all IPA alphabets, 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 7, 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.

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. While the *User Match Threshold* parameter may be tuned at the user or application level, another parameterization – *Cost Matrix* to compute the edit distance, may be tuned only based on the characteristics of the domain. We detail our experiments and a methodology for tuning the parameters for optimal matching in [9].

[3] 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.

LexEQUAL (S_l, S_r, e)

Input: *Multilingual Strings* S_l, S_r, *User Match Threshold,* e
Languages with Text-to-Phoneme transformations, $\mathcal{S}_\mathcal{L}$

Output: TRUE, FALSE or NORESOURCE

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) $\leq (e * Smaller)$ **then**
return TRUE **else return** FALSE;
6. **else return** NORESOURCE;

Fig. 7. The LexEQUAL Algorithm

3.2 Semantic Matching Strategy

Our strategy for matching multilingual data based on *semantics* in SemEQUAL operator is as shown in Figure 8. First, we convert the query string to a set of *concepts* using a standard linguistic, ontological resource, such as WordNet [11]. The WordNet is a *lexico-semantic* database that provides, in addition to other things, the context for all noun forms in standard taxonomical hierarchies covering all concepts expressible in English 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 may be on specializations of the *meaning* of the query string, using semantic closure[4] computed using WordNet. In addition to English WordNet, there are several initiatives such as *Euro-WordNet* and *Indo-WordNet* around the world, to interlink concepts of WordNets in different languages. Once the multilingual word forms are mapped onto concepts using WordNet of the appropriate language, the resulting concepts may be compared for equivalence, generalization or specialization based on the common concept hierarchy between the languages.

The SemEQUAL function takes as input a multilingual string, $Q_{Category}$, and matches it with attribute values from a specified table column – $CategoryAttribute$, storing category values, potentially in different languages. The output is `TRUE` if the category value appears in the semantic closure of the $Q_{Category}$, or `FALSE`, otherwise. The function TransitiveClosure returns the semantic closure of a given category; for example, in linguistic domain, it is computed using `Is-A` relationships in WordNet within a language and using cross linking of concepts across languages in an inter-linked WordNet. For a domain-specific ontology, we use the *parent-child* relationships for tracing the hierarchy.

[4] 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.

SemEQUAL ($Q_{Category}$, $CategoryAttribute$)

Input: *Multilingual Strings* $Q_{Category}$, $CategoryAttribute$

Output: TRUE or FALSE
[*Optional*] Gloss of Matched Synset

1. $\mathcal{W}_\mathcal{Q} \leftarrow$ WordNetOf (LanguageOf ($Q_{Category}$));
2. $\mathcal{S}_\mathcal{Q} \leftarrow$ Synset of $Q_{Category}$ in $\mathcal{W}_\mathcal{Q}$
3. $\mathcal{S}_\mathcal{C} \leftarrow$ TransitiveClosure($\mathcal{S}_\mathcal{Q}$);
4. **if** $CategoryAttribute \in \mathcal{S}_\mathcal{C}$
5. **then return** `TRUE` **else return** `FALSE`;
6. [*Optionally*] **return** Gloss of the Matched Synset;

Fig. 8. The SemEQUAL Algorithm

The recursive SQL feature defined in SQL:1999 [2] 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), we are currently experimenting with various options for storage of hierarchies and efficient indexing structures for accessing them, for efficient closure computation. 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.

3.3 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 [8]. 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 the Unicode storage size and its effect on in-memory processing.

We propose Cuniform, a compressed format that is trivially convertible to Unicode, to alleviate this problem. 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*. 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.

4 MIRA Architecture

Our proposed architecture for multilingual query processing is shown in Figure 9. The shaded boxes emphasize the new processing modules, and the iconized boxes represent resources (lexical or semantic) that are to be installed.

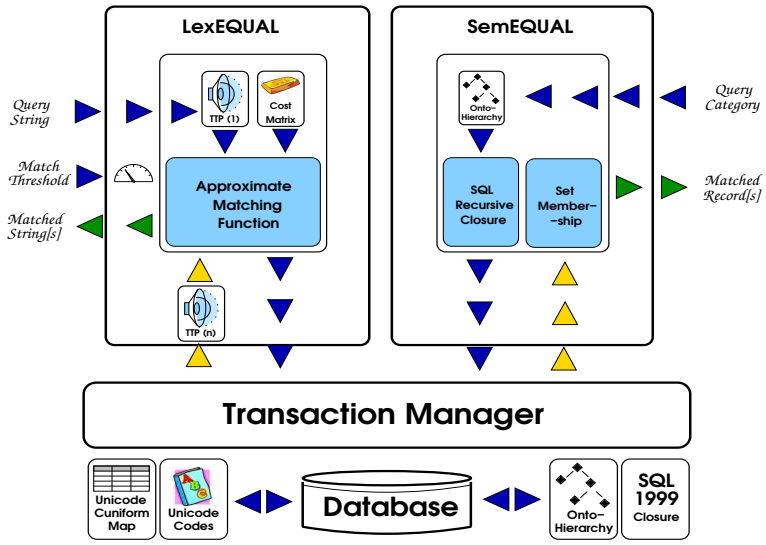

Fig. 9. MIRA Architecture

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. Our proposal employs standard components and hence may be easily adopted to any query processing or information retrieval systems.

5.1 Research Issues

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

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 efficient search capability on pre-generated phonemic strings corresponding to names. However, we find that most approximate indexes are inefficient in searches, as shown in Figure 10; 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. We are exploring the better partitioning and clustering techniques for building effective approximate indexes, to improve the search efficiency.

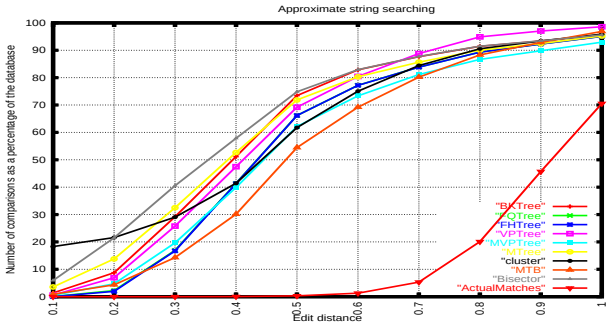

Fig. 10. Search Efficiency of Approximate Indexes

Semantic Match Quality For SemEQUAL operator, establishing the usefulness of the result on variety of ontological hierarchies is a paramount need. While we have evaluated and improved performance of semantic matching, the tightness and usefulness of the results are to be established, with proper experimentation. It should be mentioned here that while such evaluations had been done extensively in the IR community, the data set consists primarily of documents, and not attribute level information.

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.

Dynamic Resource Management The databases should be as transparent to multilingual processing, as the word processors are to multilingual scripting. Just as installation of font resources and a *language-specific* rendering module is sufficient for language specific scripting in an editor, the installation of a few lexical and linguistic resources – *language-specific* character charts, TTP converter and WordNet, must be sufficient for multilingual information processing of new language data, along with the existing data. We are investigating a dynamic linguistic resource management environment to bring about such transparency in database systems.

Real-life Application and Multilingual Performance Suites We need 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, along the lines of TPC benchmarks for OLTP applications.

References

1. The Global WordNet Association. *http://www.globalwordnet.org.*
2. International Organization for Standardization. ISO/IEC 9075-1-5:1999, Information Technology – Database Languages – SQL. 1999.
3. The International Phonetic Association. *http://www.arts.gla.ac.uk/IPA/ipa.html.*
4. L. Gravano, P. Ipeirotis, H. Jagadish, N. Koudas, S. Muthukrishnan and D. Srivastava. Approximate String Joins in a Database (almost) for Free. *Proc. of the 27th VLDB Conf., Rome, Italy,* 2001.
5. D. Jurafskey and J. Martin. Speech and Language Processing. *Pearson Education,* 2000.
6. D. Knuth. The Art of Computer Programming, Volume 3: Sorting and Searching. *Addison-Wesley,* 1993.
7. A. Kumaran and J. R. Haritsa. On Database Support for Multilingual Environments. *Proc. of the 13th IEEE Research Issues in Data Engineering Workshop (held in conjunction with 19th ICDE Conf.), Hyderabad, India,* 2003.
8. A. Kumaran and J. R. Haritsa. On the Costs of Multilingualism in Database Systems. *Proc. of the 29th VLDB Conf., Berlin, Germany,* 2003.
9. A. Kumaran and J. R. Haritsa. Supporting Multiscript Matching in Database Systems. *Proc. of the 9th EDBT Conf., Heraklion-Crete, Greece,* 2004.
10. The Unicode Consortium. *http://www.unicode.org.*
11. The WordNet. *http://www.cogsci.princeton.edu/wn.*
12. J. Zobel and P. Dart. Phonetic String Matching: Lessons from Information Retrieval. *Proc. of 19th ACM SIGIR Conf., Zurich, Switzerland,* 1996.