LexEQUAL: Supporting Multilexical Queries in SQL

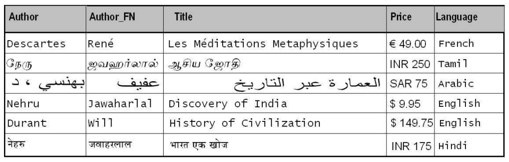
A. Kumaran *   Jayant R. Haritsa *

1. Introduction

Current database systems offer support for storing multilingual data [2], but are not capable of querying *across languages*, an important consideration in today's global economy. We therefore propose a new multilexical operator called LexEQUAL, that extends the standard lexicographic matching in database systems to matching of text data across languages, specifically for names, which form close to twenty percent of text corpora.

Our implementation of the LexEQUAL operator is based on transforming matches in language space into parametrized approximate matches in the equivalent *phoneme space*. A detailed evaluation of our approach on a real data set shows that there exist settings of the algorithm parameters with which it is possible to achieve both good *recall* and *precision*.

2. The LexEQUAL Operator



Author	Author_FN	Title	Price	Language
Descartes	René	Les Méditations Metaphysiques	€ 49.00	French
நேரு	ஜவஹர்லால்	ஆசிய ஜோதி	INR 250	Tamil
بهنسي ، د	عفيف	العمارة عبر التاريخ	SAR 75	Arabic
Nehru	Jawaharlal	Discovery of India	$ 9.95	English
Durant	Will	History of Civilization	$ 149.75	English
नेहरु	जवाहरलाल	भारत एक खोज	INR 175	Hindi

Figure 1. A Multilingual *Books.com*

Consider a hypothetical *Books.com* that sells books in different languages, with a sample product catalog as shown in Figure 1, and a user query to retrieve all the works of an author, in a set of specified languages. This can be easily achieved using LexEQUAL as shown below:

```
SELECT * FROM Books
WHERE Author LexEQUAL ‘Nehru’ Threshold 0.3
IN { English, Hindi, Tamil, Arabic }
```

The output for this query will contain all records of the table in Figure 1, for whom the `Author` attribute contains names that are *phonemically close* to `"Nehru"`, with the Threshold parameter in the query determining the match quality tradeoff between precision and recall.

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

Figure 2. The LexEQUAL Algorithm

The pseudocode of our implementation of LexEQUAL is shown in Figure 2. The algorithm transforms the input multilingual strings to their equivalent phoneme strings and flags a match if the edit distance between them is less than the user-specified error limit, `Threshold`. The transform function uses standard *Text-to-Phoneme (TTP)* converters that convert to a canonical *International Phonetic Association's* phonemic alphabet. Further, the LexEQUAL implementation is parameterized for different cost functions for editdistance computation and for domain-specific clustering of similar phonemes.

We have currently implemented LexEQUAL as a *user-defined function*, incorporating *Q-Grams* [1] to filter out *non-matches* inexpensively and *Phonemic Indexes* [4] to narrow the search to *potential matches* using standard database index structures. On a commercial DBMS, this implementation produced runtimes comparable to traditional uni-lingual matching.

In summary, the LexEQUAL operator employing phonetic matching can complement the standard lexicographic operators, representing a first step towards achieving complete multilingual functionality in database systems. The full version of this paper is available in [3].

References

[1] L. Gravano et al. Approximate String Joins in a Database (almost) for Free. *Proc. of 27th VLDB Conf.*, 2001.

[2] A. Kumaran and J. Haritsa. On the Costs of Multilingualism in Database Systems. *Proc. of 29th VLDB Conf.*, 2003.

[3] A. Kumaran and J. Haritsa. Supporting Multiscript Matching in Database Systems. *Proc. of 9th EDBT Conf.*, 2004.

[4] J. Zobel *et al*. Phonetic String Matching: Lessons from Information Retrieval. *Proc. of 19th ACM SIGIR Conf.*, 1996.

* Dept. of Computer Science and Automation, Indian Inst. of Science, Bangalore 560012, INDIA. {*kumaran, haritsa*}*@csa.iisc.ernet.in*