J. Sci. Technol. (Dinajpur) Vol. 8: 30-36 (2010)
ISSN 1994-0386

DEVELOPING AN EFFICIENT SEARCH SUGGESTION GENERATOR, IGNORING SPELLING ERROR FOR HIGH SPEED DATA RETRIEVAL USING DOUBLE METAPHONE ALGORITHM

Ashis Kumar Mandal[1], Md. Delowar Hossain[2] and Md. Nadim[3]

ABSTRACT

Finding desire information from a large text database is one of the most important issues of modern information processing systems. In this regard different types of searching techniques are used. Though some of them are vary useful, they fail to show appropriate performance when user enters a misspelled data as the searching keyword. In this paper we have developed an efficient search suggestion generator using Phonetic algorithm namely, Double Metaphone Algorithm. Here we use a technique to reduce total searching complexity by creating an index on a specific field, we have defined it as keyCode field, in a table of our database where all of the values of keyCode field are produced by that algorithm acted on records. Results show that generator not only quickly find the required information but provide possible search suggestion avoiding the misspelled words entered as search key.

Key words**:** *Phonetic algorithm, Double Metaphone Algorithm, Search Comparisons, Indexing.*

INTRODUCTION

With the increasing amount of information available on the Database, it is not easy to find quickly what one needs among the overwhelming data. For this reason, usually indexing in a database is used to speed up retrieval of data. However, it seems to be ineffective when data are incorrectly enter as searching keywords, for much more records have to be searched and any desirable information can not be found. It is also noticed that more mistakes in spellings are due to faulty pronunciation than to any other cause (Davis, 1952).In this case, phonetic encoding can be used to provide suggestions for misspelled words. We use Double Metaphone encoding for our suggestion generator. Moreover, this generator effectively and quickly identifies all of the possible suggestions for particular data entered, and user can easily find the required information from it. This is the main objective of our paper. For this purpose, we use indexing concept in such a way that with this algorithm records (what data we want to search) in the database are converted into codes .This codes are stored in a particular field that we named keyCode . We index this field and in this case there will be found solution whether entered data is spelled correct way or not. The reason is that same code is generated for similar sound. As a result, this generator quickly finds out suggestions.

There are different techniques to implement indexing and phonetic matching. Among them, Stemming is the process for reducing inflected (or sometimes derived) words to their stem, base or root form – generally a written word form. This process is useful in search process for query expansion or indexing and other natural language processing problems, but it does not show a good performance in the case of misspelling of words. Another process of eliminating the spelling is matching the word using Q-Gram based algorithms**.** For example, for $q = 2$, the word Nelson has the following q-grams: NE EL LS SO ON. By comparison, Neilsen breaks down into these q-grams ($q = 2$): NE EI IL LS SE EN. Clearly, Nelson and Neilsen share the NE and LS q-grams in

[1]Lecturer, Department of CEN; [2]Assistant Professor, Department of CIT, Hajee Mohammad Danesh Science and Technology University, Dinajpur; [3]Web Programmer, KENTO STUDIOS LTD, Bangladesh

common. This process is useful for finding spelling similarity between words but it cannot show speed in large collection of data. Soundex (Repici, 2002) or PHONIX algorithm show advantage in phonetic indexing of words, but it also can not show desire performance in a very large database because of having same soundex or phonix code of large number of words.

Metaphone algorithm, originally published by Phillips (1990, 2000) that brings great improvement in phonetic code and a modification of this algorithm named Double-Metaphone Algorithm that eliminates different types of ambiguities of metaphone Algorithm. This algorithm produces multiple variants of surnames with common ancestry. For example, encoding the name "Smith" yields a primary code of SM0 and a secondary code of XMT, while the name "Schmidt" yields a primary code of XMT and a secondary code of SMT--both have XMT in common. It is very useful in using indexing for not only speed in searching process but also for ignoring the spelling error in the input keyword.

Causes of choosing Double Metaphone: For minimizing the searching comparisons fuzzy indexing is required, and Double Metaphone algorithm is chosen for this process. For example, indexing the Apache docs (Moseley, 2002) results in these word counts:

Table I: Comparison between techniques

Technique	**Unique Words**
None	7270
Stemming	5149
Soundex	2651
Metaphone	2957
Double Metaphone	3189

The above table expresses the following facts:-

- The non-indexed document requires maximum number of comparisons.
- Stemming minimizes comparisons but not so significant.
- Soundex algorithm generates less number of unique codes but it becomes ambiguous if the number of data is very large.
- Metaphone algorithm is more reliable than Soundex algorithm, but it also shows some limitations for some ambiguous words
- Double Metaphone is the modifications of the Metaphone algorithm and it is eliminates the limitations of Metaphone. It is comparatively more reliable than any of the above algorithm for making fuzzy indexing.

Zobel and Dart (1996) explain Parallels between information retrieval and phonetic matching. They have developed several new methods for phonetic matching, and have used measurement techniques developed for information retrieval to compare them .Zaman and Khan (2004, 2005) use phonetic encoding for Bangla to provide better suggestions for misspelled words. They propose a Double Metaphone encoding for bangla name that can be used by applications to search for and match names. Most of the cases, there is little discussion on effectiveness of searching processes.

METHODS AND MATERIALS

As stated earlier our searching process keeps an idea about the location of data. For example, in our system we have implemented the searching process in a table of books information **(named books)** of a Library Management System. For any query the searching process displays searched result without searching the original database. It first makes a decision about the presence of the data and then if user selects any displayed result; a query is executed to retrieve the original data from the original table. The tables for this implementation are **savedSearch** (keeping the pre-

established knowledge about data location)**, codeIndex** (Keeping Metaphone codes of words in the database), **temporaryResult** (temporary storage of the searched result)**.** The relation among the tables is as follows:

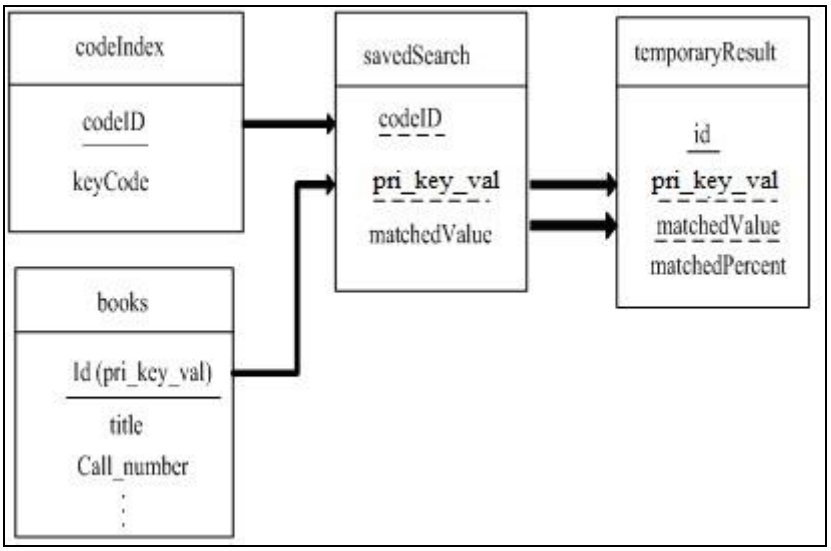

Figure 1. Tables used in Searching Process

Phases of the Working Process: The whole process can be divided into two phases. Phase-1 works when data inserted into the database and phase-2 works during the searching process. The working procedures of the two phases are described as follows.

Phase-1:

- Take the input for inserting in the database.
- Select the words which may be searched.
- Make Metaphone code for the searchable words.
- Create map based on the Metaphone code.

Phase-2:

- Remove stop words(such as of, about, on etc.) from the input (Search) keyword.
- Create Metaphone code for each of the words in the search key.
- Retrieve **pri_key_val** from the table **savedSearch**.
- If more words have same **pri_key_val** merge them in a single line**.**
- Calculate percentage of matching with the keywords and the actual data in the database.
- Display retrieved data in the descending order of the Matched Percentage with the actual data.

From the display, if the user selects a data then **pri_key_val** associated with the data helps to retrieve the actual data from the database directly.

Flow Diagram: The whole searching process is explained by the following

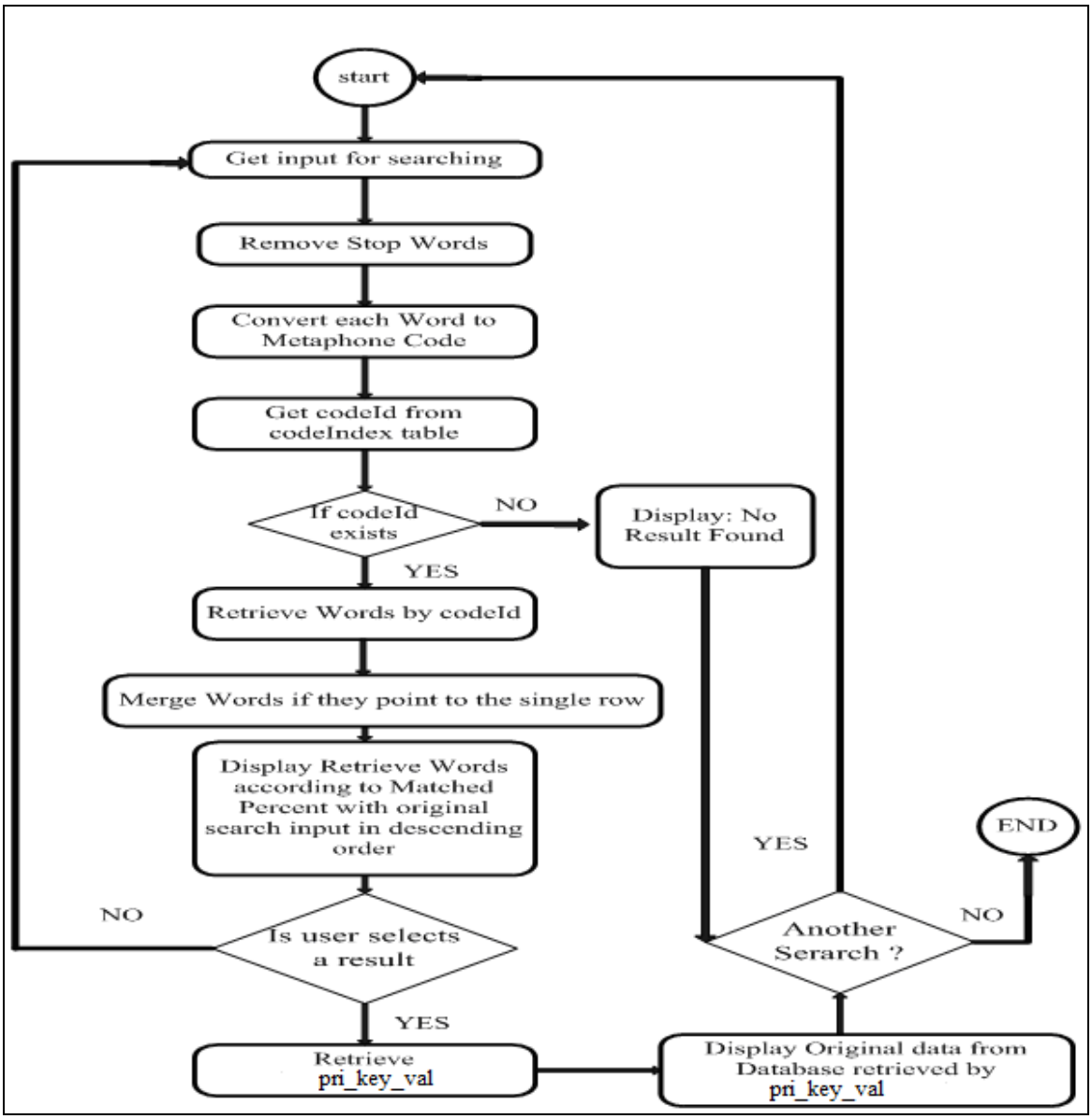

Figure 2. Flow Diagram of Searching Process

Explanation by example: The searching process for a particular input (Here spelling error occurred and user desiring output for **Networking Phenomenon**) will be performed as follows:

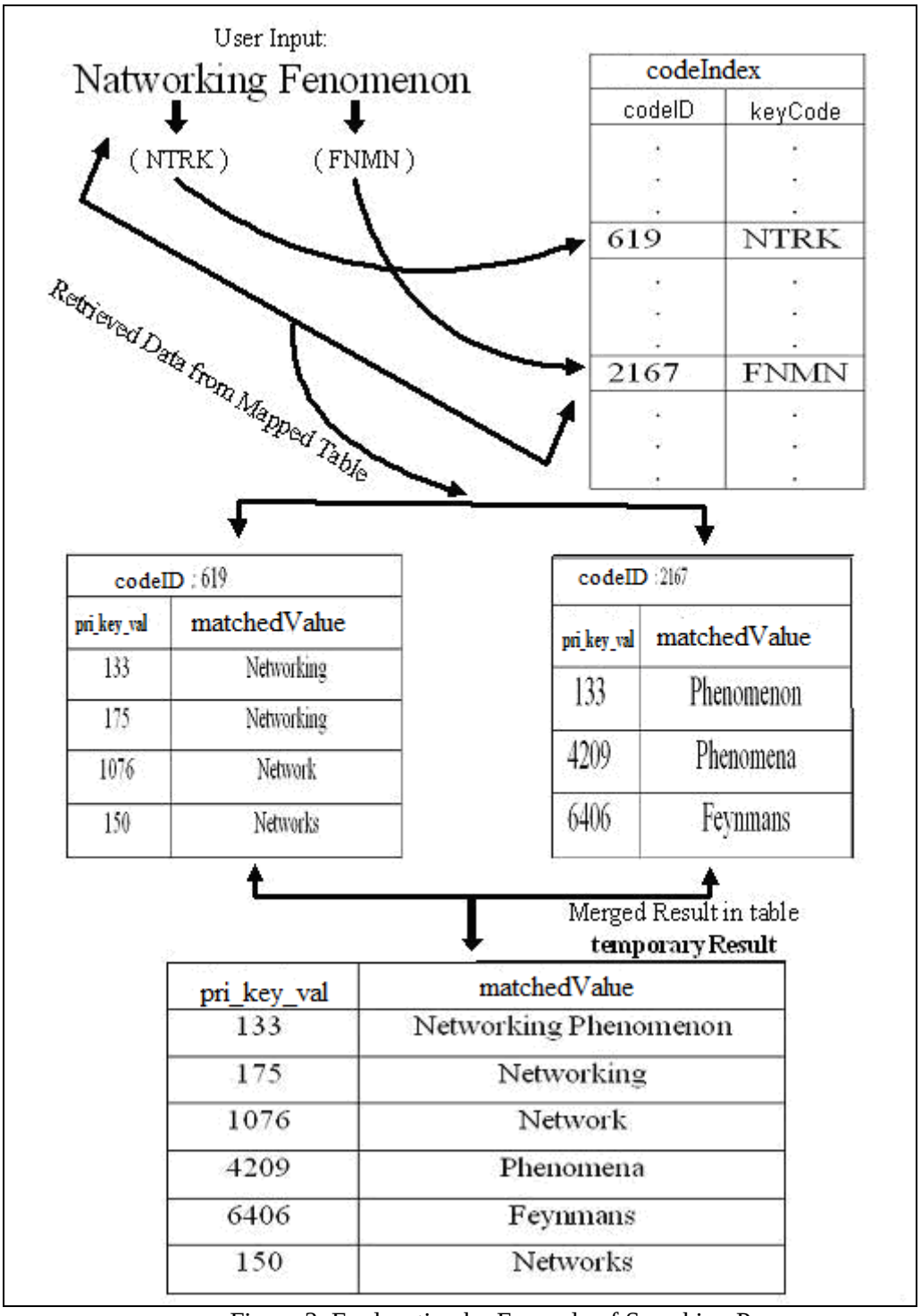

Figure 3. Explanation by Example of Searching Process

Now if the user selects the first displayed data from temporary result query to retrieve the original data (in books table) is: **SELECT * FROM books where id= 133;**

RESULTS AND DISCUSSION

If we do not have any information on how the data are organized in the data structure, we have to sequentially examine each element of the data structure. This is known as linear search and would have a time complexity of O(n) in the worst case.

However, if the elements of the data structure are ordered, let us say in ascending order, and we wish to find out the position of an integer target K in the array, we need not make a sequential

search over the complete data structure. We can make a faster search using the Binary search method. Time complexity of the Binary search method is O(logn), which is much more efficient than the Linear Search method.

But what we will do if the input keywords have error in spelling, then it is impossible to implement binary search to retrieve the required data. Then, we have no choice to implement leaner search which will results in a very slow process. Our generator shows excellent performance in such a situation. Because it not only implements binary search in such situation but also reduces the search domain depending on the input keywords.

We have performed an experiment by taking random book information, where increase of comparison is expressed with the increase of number of books.

Idea behind Experiment: The main idea behind our experiment is approximating the number of comparisons with the increase of number of books in a certain database. Following steps are performed for such approximation:-

- We Take 500 random books information.
- From that information the words whose length is greater than 2 are treated as searchable words.
- Number of such searchable words are determined by C-program for different number of books information such as 0, 50, 100… 500.
- These numbers denotes the number of comparisons if we perform a normal search over the data.
- Number of Unique Metaphone codes are determined by php-program for different number of books information such as 0, 50, 100… 500.
- These numbers denotes the number of comparisons if we perform search taking the help of Metaphone code.
- The values plotted in graph for clear explanation.

Table II: Data for analysis.

Books Information	Metaphone Code	Searchable Words
0	0	0
50	205	360
100	480	880
150	550	1040
200	755	1590
250	895	2020
300	985	2580
350	1075	3115
400	1155	3605
450	1255	4180
500	1345	4640

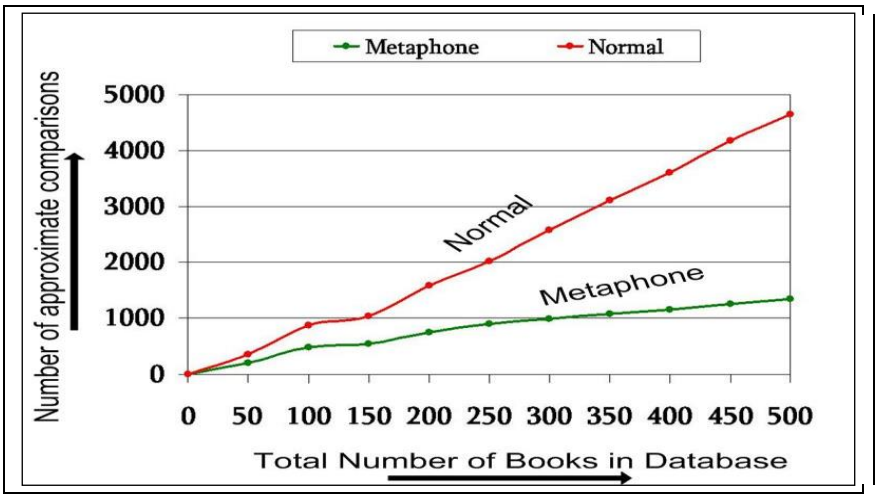

Figure 4. Comparison between Normal and Phonetic Indexed Search

Seeing above figure we can easily conclude that, the rate of increase of number of comparisons for Metaphone coded search is much slower than the normal search. Thus the speed of Metaphone searching technique is very high than the normal linear search technique. We can minimize the searching comparisons by looking in the Metaphone Coded Look-up table. The minimization of number of comparisons is more significant when the amount of data is very large.

CONCLUSION

We always look for easy, efficient and effective way to access right information whereas we intentionally or unintentionally misspelled word as a search keyword. In this paper we have developed an efficient search suggestion generator with the concept of phonetic encoding and indexing techniques of databases. From the experimental result, it has been proved that our developed suggestion generator works fine even in large database. We have confined our discussion on English contents .So, further work can be done to enhance the system, for example, features like database with multilingual contents.

REFERENCES

Davis LE. 1952. College English.13. No 4. pp 221-223. 1952.

Moseley B. 2002. Fuzzy Indexing with Double Metaphone.
http://swish-e.org/archive/2002-08/4381.html

Phillips L.1990. Hanging on the Metaphone.Computer Lang. 7:12

Phillips L. 2000. The Double Metaphone Search Algorithm. C/C++ Users J.18:6

Repici D.2002.Understanding Classic SoundEx Algorithms.
http://creativyst.com/Doc/Articles/SoundEx1/SoundEx1.htm

Zaman N and Khan M. 2004. A Bangla Phonetic Encoding for Better Spelling Suggestion. Proce 7th Int. Conf. on Computer and Inform. Tech, Dhaka.

Zaman N and Khan M. 2005. A Double Metaphone Encoding for Approximate Name Searching and Matching in Bangla. Proce. 4th IASTED Int. Conf. Computational Intelligence. pp. 108-113, Calgary, Alberta, Canada.

Zobel J and Dart P.1996. Phonetic String Matching: Lessons from Information Retrieval. Proce. 19th annual international ACM SIGIR conf. on Res. Dev. in inform. retrieval, Switzerland.