A Framework for Name Search and Spell Check for Social Network Application using Regular Expression & Hashing

Ritesh Kothari¹, Nishchol Mishra², Sanjeev Sharma³, Jitendra Singh Verma⁴

¹Research scholar, School of IT, RGPV Bhopal, India  
²Assistant Professor, School of IT, RGPV Bhopal, India  
³Associate Professor, School of IT, RGPV Bhopal, India  
⁴Assistant Professor, School of IT, RGPV Bhopal, India

Abstract: String matching takes place an important role in the field of information retrieval, spell checking, dictionary and many more. Some of them can be either web based applications or desktop based applications. These different applications can use the concept of two types of string matching techniques one is exact and another is approximate string matching techniques. This paper deals with both the string matching techniques mentioned above. Objective of this paper is to shows the use of string matching techniques in retrieving the relevant information efficiently. Therefore to fulfill this requirement we have proposed a name search framework regular expression and hashing based name search (REHNS) including the spell check feature. Proposed names search framework will not only give the relevant names while it will also suggest the correct spelling of name, if found incorrect. For getting better results we have applied regular expression, hashing technique and edit method. Proposed system suggests the best similar name for mistyped names as well as gives the names in an alphabetical order in quick time. This is very helpful to people if they are searching the names starting with a particular letter.

Keywords: string matching, spell correction, name search, social networking

I. INTRODUCTION

We are introducing here about string matching and its benefits in some application areas. We are well-known about the word “string matching”. [5] But behind its popularity it is very much useful also in the real life applications. As we know about the working behind string matching, that it is the study of comparing two words. Whenever a pattern string is matched to the text string, it is called string matching and when a pattern string is searched in another string say text string then it is called string searching. In some another way we can define string matching as, let we have a query string “Q”, output strings “M” and a database dictionary “S”. So according to the definition we have to approach for getting all the output strings “M” found in the database “S” which are exactly or approximately matched to input string “Q”. [5] String matching has two important categories one is known as exact string matching and another one is approximate string matching. If string is found in a particular pattern exactly then it is called exact string matching and if similar strings are found then it is called approximate string matching technique. [4][5] May be Some people are not aware of different categories of string matching techniques as discussed above, while these are very important in our daily life applications. For example the documents searching in an offline or online mode etc. Here spell checking takes place an important role, when human does mistake in spelling of words. Approximate string matching is an area where a pattern is found in text string with at most k errors.[3][4] Approximate string matching is applied in spell checking with the help of some ‘similarity’ mechanisms are applied in spell checking like edit distance, Leveshtein distance etc. Further some algorithms are based in n-gram theories etc. [2]. This paper works with techniques like finite automata, hashing and edit method for retrieving names. We have focused here for retrieving names in an alphabetical order. Spell checking is also applied here for misspelled name.

Study and researches on approximate string matching and spell checking has become very important because in this present scenario where a huge percentage of population is using internet and they are connected to people on social networking sites, dealing with emails, searching the information on search engines, and many more. A study on string matching and spell check has been increased because the above mentioned activities require string matching technique and some of them needs spell checking as well. For example if somebody wants to
search the meaning of his name on internet then he or she should get the proper results even he types misspelled names. Another example of importance of string matching we can see on social networking sites where people always want to search names for their old friends, relatives etc.

II. PRELIMINARY

In this section we can discuss the methods and techniques used in the proposed work.

A. Regular Expression

Regular expression is a text string for describing the search patterns in text string. For a string search and match regular expression is used. It is also known as regex. Basically, a regular expression is defined as a pattern finding in a given text. We have applied regex in our proposed work for string matching in an efficient way. Regular expression is a name comes from the mathematical theory. A "match" can be a piece of any text or sequence of bytes or characters where pattern was found by applying some regex expressions or operations. [8]

B. Edit Distance

We introduce here about an important feature used in implementation of approximate string matching techniques which is edit distance. Edit distance, introduces the spelling check between two strings. Edit distance works on four distance parameters. These parameters are insertion, deletion, substitution and transposition. We can simply explain the edit distance by calculating the character differences. [6][10]

C. Hash map

Hashing provides an easy way to searching patterns string by avoiding quadratic number of character comparisons. Hashing uses hash function. [5] A hash function is used to map the data of any arbitrary size to the data of fixed length size. [9] Hash functions are primarily used in hash tables or hash map. Hash map is a kind of data structure. Hashing is performed efficiently by using hash function. [1]

III. LITERATURE SURVEY

This section represents the work done in the field of approximate string matching with allowing spell checking. This section contains work done regarding spell checking technique using approximate string matching algorithms. Different works have been proposed in the field of string matching regarding information retrieval. Today, huge amount of data is stored online over a cloud daily so there have been more researches and work in this area.

Author Youssef Bassil [10] has proposed a novel shared memory parallel spell check algorithm based on Yahoo n Grams dataset (comprises trillions of word sequences and n-grams)[7], for detecting and correcting spelling errors in computer text. During an experiment Youssef Bassil used 300,000 misspelled word articles and found that, 94% errors are corrected, which is a big success in approximate string matching technique.

BurakYildiz, FatihEmekci [6] have proposed the solution for fuzzy query where approximate string matching takes place an important role. BurakYildiz, FatihEmekci proposed an algorithm ‘edit distance aware name filtering’ for solving the approximate string search problems in spell checking. Authors in [6] developed a spellcheck engine it is very useful for social networks. The study of their results shows that the query processing time and throughput is improved by almost 30%. Further name spell check engine improved the name spell check response time, and throughput by using filtering schemes and some other parameters. [6]

IV. PROBLEM DEFINITION

The problem solved in this paper is to find all the names in alphabetical order, which are available into the name database, which starts from a particular words or a character and filtered other names which contain the same words at some other positions. This paper also finds the correct name for misspelled name up to two misplaced characters.

V. PROPOSED WORK

In section 3, we have discussed that the work is based on spell correction which retrieves all the similar names corresponding to the misspelled name at ‘e’ edit distance away and very helpful to user if they make a mistake in spelling. We have discussed literature related to the spell correction. These literatures worked only on spell correction. In this paper we proposed a search framework ‘REHNS’ for efficient name search and spell check. In this section we have described proposed framework with the help of ‘REHNS’ algorithm and flow chart diagram in 5.1 and 5.2 respectively. We have applied the regular expression, hashing technique and edit method in ‘REHNS’ framework.
A. Proposed algorithm

REHNS(S=string, N=dataset, P= array list)
1. Input (S, N);
2. Output (suggested name, all the names in an alphabetical order);
3. S  String (alphabets);
4. P  dataset (N);
5. Apply finite state automata(S);
Implement RE (S); // RE=regular expression
Make patterns using ‘+’ regex meta character;
//S+ ε strings consist (S).
W  (S); // W = temp array list use to store patterns
6. Implement hashing associated with tree (W);
Suffix tree (W);
while (P [] = = W);
W¹  patterns matched to string(S); // W¹is temp array list use in algorithm
Hash map (i)  W¹
X  hash map (i);
7. Filtering (X);
sort names (X);
Check spelling (S);
If correct;
Return X;
Else apply edit_e using substring (S);
Arraylist [ ]  substring (S);
Check (Arraylist [ ] (S));
If found (s); //s =correct word
Return s;
else return no;
8. Database (N)  word (S);
9. Display (M);

Algorithm 1 REHNS Framework

B. Flow Chart Diagram

Figure 1 shows the working process of proposed ‘REHNS’ framework. The flow chart diagram contains several steps to run REHNS framework. First step is ‘start’ phase. Here user enters a ‘web address’ after that user enters into the second phase of REHNS framework, which is ‘homepage’. At the homepage user is allowed to perform operations. Here user can either search name or add some names. If he wants to increase the dataset then he should go for ‘login’ phase after making registration. Now if user wants to the search name, he can go to the search box and then enters the name.

Now the matching and searching process starts. The internal process consists the phase ‘regular expression’, where the regex function makes the possible patterns of query name and then matches the relevant names to the query name. After this process ‘hashing’ phase comes, where all the names are stored into the hash map. After this, ‘filtering’ phase comes, where the filtering process of spell checking and correction applied. Here a method ‘edit’ is used to suggest the misspelled name. Here query name generates the substring and then matches to the names. Finally the suggested name is sent to the search box and all the relevant names are displayed to user.

VI. RESULTS AND ANALYSIS

We have implemented and run REHNS framework in java. For creating the database we have used Postgres 9.0. For web service we have used apache tomcat server 6.0 versions. We have analysed our result over 1000 names. In the result analysis section we have analysed our results in different categories.
and different ways. Table 1 shows the performance of proposed framework in terms of user input, actual input name, relevant names and suggested name. According to the table 1 we can say that when user searches for Ram then he gets Ram and all the names starting with Ram. According to the second row of table 1, we can say that when user enters Chetanya in place of ‘Chaitanya’, then REHNS framework checks the input name and corrects as Chaitanya. Therefore we can say that REHNS framework suggests the best similar name. In the second row character ‘a’ is substituted by ‘e’ and ‘i’ is deleted. Similarly in last row transposition of characters are performed in spell check.

<table>
<thead>
<tr>
<th>User Input</th>
<th>User wants to search</th>
<th>Suggested result</th>
<th>Final results</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ram</td>
<td>Ram</td>
<td>Ram, Raman…</td>
<td>Ram, Ram, Ramshankar ……</td>
</tr>
<tr>
<td>Rem</td>
<td>Ram</td>
<td>Ram</td>
<td>Ram, Ram, Ramshankar</td>
</tr>
<tr>
<td>Chetanya</td>
<td>Chaitanya</td>
<td>Chaitanya</td>
<td>Chaitanya</td>
</tr>
<tr>
<td>Mosumi</td>
<td>Mausumi</td>
<td>Mausumi</td>
<td>Mausumi</td>
</tr>
<tr>
<td>Cahitanay</td>
<td>Chaitanya</td>
<td>Chaitanya</td>
<td>Chaitanya</td>
</tr>
</tbody>
</table>

Table 1 Result analysis of REHNS framework

Figure 2 shows the graph of time analysis based on the datasets. According to figure 2 we have analysed the results based on time. We have performed search for five different names and analysed the time in seconds.

Figure 3 shows the relation between the time and dataset. Figure 3 shows the time for searching the names containing the datasets. The result we analyse from this graph is that the proposed REHNS framework gives the names in quick time as we can see.

Further we can analyse our result in terms of time and different no of datasets. This dataset contains names. We have added 1000 names to our data dictionary and then performed a search for four stage of dataset. We partitioned our dataset in four categories and then performed a search for query name at these different levels and found that at each level the time is not being so much changed. It means if dataset is increased the time does not suffer so much in searching. Figure 3 shows the relation between the time and dataset.

VII. CONCLUSION AND FUTURE WORK

String matching is a useful technique which is used in our daily life applications. We have applied the combination of exact and approximate string matching techniques in searching the names. Result shows that REHNS framework works efficiently and gives the relevant names including spell check feature. We have also added a drop down list which gives the desired word character by character and suggestion for misspelled name. We have developed a framework which can help in such user friendly activities, which require either alphabetically words or words starting from any particular letter in quick time. To making it informative we have also added an extra field as meaning of names where we can add meanings of names. In future this web application can be designed having some more
features as mobile application and also this REHNS framework can give more suggestions for misspelled name, so that user could get desired result. With more efficient algorithms and techniques we can reduce the time. We can also add some another fields like meaning for getting the information.

REFERENCES


