Article can be accessed online at http://www.publishingindia.com

An Evolutionary and a Rule-Based Approach to String Transformation

Nandita Bhanja Chaudhuri*, D. Kamal Kumari, S. Ram Prasad Reddy*****

Abstract

Natural language processing focuses on analyzing and processing human languages using digital computers. String transformation is an important area of research in the field of natural language processing. String transformation maps a source string to a desirable form, which is related to various applications like spelling error correction, query reformulation, top k related candidate generation, and word stemming. Even though various traditional approaches are available for string transformation but they cannot be considered as optimal because accuracy and efficiency are the basic parameters to optimise. This paper proposes a novel model for string transformation which is up to 99% accurate, with an improved F-measure and G-measure. The model is intended to use evolutionary approach, which involves the methods to search the population of keywords, algorithm to find the distances between strings, and finally transforming the strings with and without using a dictionary. Our paper mainly focuses on the following: (1) spelling error correction which detects a wrong spelling and provides correct suggestion, (2) top k candidate generation which provides the most related suggestions for a keyword, (3) query reformulation which transforms a short form of a query into an elaborate form, and (4) word stemming which identifies a part of a word when it is also concatenated by grammatical stuffs. In short, it identifies redundant queries. It is rule-based system which is implemented without using a dictionary. Graphical comparisons are demonstrated for the candidates generated between the existing and the proposed system.Experimental results on large scale data shows that the proposed model is accurate and improved over the traditional approach.

Keywords: Natural Language Processing, String Transformation, Spelling Error Correction, Top K Candidate Generation, Query Reformulation, Stemming, F-measure, G-measure

1. Introduction

Several problems in natural language processing, information retrieval, data mining, and bioinformatics can be formalised as string transformation, which is a task. Given an input string, the system generates the *k* most likely output strings corresponding to the input string (Wang*et al.*, 2013). Natural language processing deals with word stemming, pronunciation generation, query reformulation, spelling error correction, and candidate generation which are allgeneralised as string transformation.This paper focuses on string transformation in the aspects of spelling error correction, top *k* candidate generation, query reformulation, and word stemming.

String transformation acts as a bridge between the source string and the desirable string. Enormous numbers of algorithms are proposedbut the task of investigation is still going on as accuracy and efficiency are the two basic parameters that need to be optimised. Although certain progresses have been made in this field, this paper also enhances the accuracy of the algorithm. In this context, accuracy precisely refers to the number or percentage of correctly generated candidates.

* Student, CSE Department, Vigana's Institute of Engg. for Women, Ranga Reddy, Telangana, India.
E-mail: nbhanja_chaudhuri@yahoo.com
** Asst. Prof., CSE Department, Vigana's Institute of Engg. for Women, Ranga Reddy, Telangana, India.
E-mail: dadhiraokamala@gmail.com
*** Assoc. Prof., CSE Department, Vigana's Institute of Engg. for Women, Ranga Reddy, Telangana, India.
E-mail: reddysadi@gmail.com

String transformation can be done in two ways i.e., by using a dictionary and without using a dictionary. Here, spelling error correction, top *k* candidate generation, and query reformulation is based on a dictionary but the word stemming is a rule based system which doesn't use a dictionary.

Often users end up entering wrong spellings while searching. So, candidates are generated from the dictionary which displays a number of nearest correct spellings corresponding to the entered word. The candidates are generated by index based searches, alphabetical searches, and by applying various operations like insertion, deletion, and swapping. For example, the candidates for "applle" are "apple", "app", "applet", etc.

Query reformulation aims to solve the mismatch problem in a document. Suppose a query is "ASAP" and the document contains only "As Soon As Possible", so the document will not be ranked high although the word is present in the document. So, the task of query reformulation is to transform "ASAP" to "As Soon As Possible", which directs to the high ranking of a document.

In linguistics, a stem is a part of a word. Word stemming is aimed at dealing with redundant queries i.e., it identifies a part of a word from the same word that is concatenated with inflectional suffixes and grammatical stuffs like "ing", "ed", "s", "es", "ies", "ship", etc. For example, the words like "calculate", "calculates", "calculating", "calculated", "calculation", "calculations", etc. are all stemmed to "calculate".

In this paper, an empirical model of string transformation is proposed by the evolutionary approach. This method is novel and unique since it confronts the following objectives. It employs evolutionary algorithm for spelling error correction, top *k* candidate generation, query reformulation, and rule-based algorithm for word stemming. An experimental result of the evolutionary approach with the Levenshtein edit-distance approach is demonstrated. Finally, the percentages of accuracy, F-measure, and G-measure are demonstrated.

2. Related Works

Traditional work on string transformation was mainly based on efficient generation of the strings, assuming that the model is given (Behm*et al.*, 2009).Others tried to learn the model with different approaches. Accuracy was considered as a prime factor in these methods. The proposed system enhances the accuracy of the string transformation.

2.1. String Transformation

String transformation maps a source string into the desirable string. Following data have been collected by mining enormous number of models.

Tejada *et al*. (2002) proposed an Active Atlas system, which applies a set of domain-independent string transformations to compare the objects' shared attributes in order to identify matching objects. In addition, this system learned to tailor the weights of a set of general transformations to a specific application domain through limited user input.Arasu*et al*. (2009)propose a greedy approximation algorithm which can learn a set of transformation rules that explain most of the given examples. The major factor of it was to elaborate the rule set. Li *et al*. (2007)developed a novel technique, called VGRAM, which improved the performance of approximate queries on string collections using variable length grams. Yang *et al*. (2010) proposed *q*-gram based framework that uses an efficient algorithm for top *k* approximate string matching. Yang *et al*. (2008)proposed a dynamic algorithm for computing a tight lower bound on the number of common grams shared by two similar strings in order to enhance query performance. They also put forward a method for automatically computing a dictionary of high quality grams for a workload of queries. Okazaki *et al*. (2008)proposed a discriminative candidate generator for string transformation. They used substitution rules as features and scored them using a L_1-regularised logistic regression model P (t | s), where s denote the input string, t denote the output string, and a feature represents a substitution rule.

$$fk(s,t) = \begin{cases} 1 \text{ rule rkcan convert s to t (1)} \\ 0 \text{ otherwise} \end{cases}$$

This model uses all the rules to transform the input string to the output string but applies only one rule at a time. Ristad&Yianilos(1997) proposed a stochastic model that allowed learning a string edit distance function from a corpus of examples.The distance $d_c(x^t,y^v)$ between two strings $x^t \in A^t$ and $y^v \in B^v$is illustrated recursively as

$$dc(x^t, y^v) = \min = \begin{Bmatrix} c(x_t, y_v) + dc(x^{t-1}, y^{v-1}), \\ c(x_t,) + dc(xt-1, yv), \\ c(, y_v) + dc(x^t, y^{v-1}) \end{Bmatrix} \quad (2)$$

where, $d_c(\square, \square) = 0$, A and B are finite alphabets, c is a cost function, t and v are string length. The edit distance may be computed in O(t.v) time dynamic programming (Wagner, 1980; 1974). Brill (1995) proposed a transformation-based error driven learning which is a rule based approach to automate learning of linguistic knowledge. Dreyer *et al.* (2008) developed a log-linear model for string transformation and had features that represent latent alignments between the source and the destination strings. Finite-state transducers were used to generate the candidates. Efficiency was not a major factor for this model.

2.2. Approximate String Searching

Hadjieleftheriou&Li (2009) presented an overview of the progress in approximate search in string collections. Given a collection of strings, it efficiently identifies the ones similar to a given query string. Such a query is termed as approximate string search. Several selectivity-estimation algorithms have been proposed for different similarity functions and employed by various techniques like histogram, clustering, and sampling. Li *et al.* (2008) proposed algorithms that could enhance the performance of the existing algorithms for approximate string searches. Moreover they integrated existing filtering techniques to have a further enhancement. Ji *et al.* (2009)proposed a novel algorithm to answer single-keyword fuzzy queries. This algorithm uses cached results of previous queries to compute the answers which direct to achieve an interactive speed on large data sets.

In approximate string search, the model is based on similarity and it searches the string within the dictionary. Most existing methods uses trie based algorithms (Ji*et al.*, 2009), or n-gram based algorithms (Behm*et al.*, 2009; Li *et al.*, 2007; Yang *et al.*, 2008; Li *et al.*, 2008). The top *k* candidates are also generated by the n-gram methods (Yang *et al.*, 2010; Vernica & Li, 2009).

2.3. Query Spelling Error Correction

Enormous number of research has been done on spelling error correction but only some algorithms have focused on the parameters of accuracy and efficiency. Candidate generation is also a part of spelling error corrections. Mainly edit distance is used for candidate generation which involves three operations like insertion, deletion, and swapping(Damerau, 1964). The calculation method of the Levenshtein distance between two strings $X = x_1x_2\ldots x_m$ of length m and $Y = y_1y_2\ldots y_n$of length n, involved in calculating recursively the edit distance between different substrings of X and Y. The edit distance between the substrings $X_1i = x_1x_2\ldots x_i$ and $Y_1^j = y_1y_2\ldots y_j$is demonstrated by the following recursive relationship.

$$D(i, j) = D(X_1i, Y_1^j) \quad (3)$$

$$D(i, j) = \min\{ D(i-1, j) + 1, D(i, j-1) + 1, D(i-1, j-1) + \text{cost} \} \quad (4)$$

$$\text{With cost} = \begin{cases} 0 \text{ if } x_{i-1} = y_{j-1} \\ 1 \text{ otherwise} \end{cases} \quad (5)$$

$D(i, Ø) = i$ and $D(Ø, j) = j$ are initialised, where Ø denote empty string. Hicham et al. (Hicham, 2012) modified the Levenshtein algorithm by the following measure M(i , j).

$$M(i, j) = \min\{ M(i-1, j) + 1 - F_{aj}(x_{i-1}), M(i, j-1) + 1 - F_{sup}(y_{j-1}), M(i-1, j-1) + \text{cost} \} \quad (6)$$

$$\text{With cost} = \begin{cases} 0 \text{ if } x_{i-1} = y_{j-1} \\ 1 - F_{permut}(x_{i-1} / y_{j-1}) \text{ otherwise} \end{cases} \quad (7)$$

where, $F_{aj}(x_i)$ = the error frequency of adding the character $x_i$in the word, $F_{sup}(y_j)$ = the error frequency of deleting the character $y_j$in a word, $F_{permut}(x_i / y_j)$ = the error frequency of the permutation x_i with the character y_j.Following initialisations were made, M(0 , 0) = 0, M(i , 0) = M(i-1 , 0)+ $F_{aj}(x_{i-1})$, and M(0 , j) = M(0 , j-1) + $F_{sup}(y_{j-1})$.

Li *et al.* (2006) proposed distributional similarity based models for query spelling error correction. Golding and Roth (Golding, 1999) proposed an algorithm combining variants of winnow which is a multiplicative weight update algorithm, and weighted-majority voting which are applied to a large class of machine learning problems in natural language. A context sensitive spelling correction was the goal of this algorithm.Brill &Moore (2000) proposed a unique channel model for spelling correction which is based on generic string to string edits. Whitelaw *et al.* (2009)implemented a system for spellchecking and auto correction system which uses World Wide Web as a noisy corpus. Oncina&Sebban (2005)aimed at learning an unbiased stochastic edit distance in the form of a finite-state transducer from a corpus of pair of strings based on an expectation maximisation algorithm.

McCallum *et al*. (2005a) proposed discriminative edit distance CRFs, a finite-state conditional random field model for edit sequences between strings. It is advantageous over generative models since they enable complex, arbitrary actions, and features in the input strings. Ahmad &Kondrak(2005)utilise the expectation maximisation algorithm to learn an accurate error rate model without relying on a corpus of paired strings for spelling correction.Toutanova&Moore (2002) proposed a combined pronunciation and letter-based model for incorporating word pronunciation information in a noisy channel model for spelling correction. Duan&Hsu (2011) proposed a generative model for spelling correction based on a noisy channel transformation of the intended queries. A Markov n-gram transformation model that captures user spelling behaviour in an unsupervised fashion is trained by utilising spelling correction pairs. The A* search algorithm and several heuristics are applied to acquire the top spelling correction suggestions. Chen *et al*. (2007)utilised maximum entropy model for spelling error correction based on web search results instead of lexicons. Islam &Inkpen (2009) proposed a method for detecting and correcting multiple real word spelling errors using the Google Web 1T 3-gram data set and a normalised and modified version of the Longest Common Subsequence string matching algorithm. Their aim was to improve detection recall and correction recall with a possible high precision. Islam &Inkpen (2011)proposed an unsupervised approach that automatically detects and corrects a text containing multiple errors of both syntactic and semantic types like typographical errors, unwanted words, prepositional errors, missing words, punctuation errors, and many of the grammatical errors.

2.4. Query Reformulation

Query reformulation deals with transforming ill-formed queries into well-formed queries to enhance the search effect. It is implemented to achieve better search result so that the documents could be ranked high. Various systems try to derive rules for reformulation from search query logs. Huang &Efthimiadis (2009)proposed a high precision rule-based classifier to detect various types of reformulations like add or remove words, substitutions etc. They also analyzed query reformulations in the AOL query logs using metrics which are indicators of effectiveness. Jones *et al*. (2006)used a query corpus and detected the phrase based transformation rules from queries. They divided the queries into phrases and generated candidates depending on rules of substitutions of the phrases. The log likelihood ration is used to calculate the weights of the transformation rules.Wang &Zhai (2008)implemented pattern based techniques in query reformulation in which they extracted substitution patterns and substituted the words in the queries using these patterns. Guo*et al*. (2008) proposed conditional random field for query refinement (CRF-QR) model for predicting a sequence of refined query words as well as corresponding refinement operations. This model has the ability to perform multiple reformulation tasks simultaneously, thereby enhancing the accuracy of the system.

2.5. Word Stemming

A stem is a part of a word. Stemming is a method for transforming derived terms into corresponding root or stem words (Gupta, 2014).Word stemming identifies a part of a word when the word itself is concatenated with grammatical suffixes and inflectional suffixes. Many algorithms have been proposed but the research still continues for finding the best algorithm. Majumdar*et al*. (2007) proposed YASS (Yet Another Suffix Stripper) stemmer which is based on clustering technique depending on the parameter of string distance.Ramanathan&Rao (2003) proposed a light weight stemming algorithm whose objective wasto strip the suffixes based on maximum match. It was tested on various types of documents collected from politics, games, business etc. There were thirty five thousand nine hundred and seventy seven unique terms present in the documents. The algorithm identified fourteen percent over stemming error and five percent under stemming errors. Majgaonker&Siddiqui (2010)proposed an unsupervised stemming technique, in which they developed suffix rules depending on three techniques viz. statistical stripping, rule-based approach onstemming, and suffixes stripping. The unsupervised technique learns the suffixes automatically from different texts that are available. The rule-based technique applies various suffix stripping rules. Pandey &Siddiqui (2008) proposed an unsupervised stemmer where words were collected from a corpus- EMILLE, which was used for training. Further, the probabilities of stem and suffix were identified and split probability was obtained from them. Post processing techniques were applied for further refining the learned suffixes.

3. Proposed Models for String Transformation

We propose two approaches for string transformation-(1) evolutionary approach and (2) rule-based approach. The methods are highly accurate and arebased on or without a dictionary. The performance of the system increases as the data size increases in the dictionary.

In the first method, we have considered a large population of strings from the dictionary for generating the likely candidates for the model. Fitness of the candidates is measured and mutations are performed in order to achieve the best candidates. In the second method, we have considered a large number of unambiguous rules for transforming the input string into the desired string.

3.1. Model Details

The overviews of our models are shown in Figure 1 and Figure 2.

Evolutionary approach is an optimisation technique or a heuristic solution search. It applies a fuzzy version of evolutionary processes to evolve solutions to the problems. It works on a population of artificial chromosomes which are strings of finite alphabets. Each chromosome acts as a solution to a problem which has fitness. The fitness is a real number which is a measure of the degree of goodness of the solution to a particular problem (McCallum, 2005). In this model, there are two processes in the evolutionary approach i.e.,learning and generation. In the learning process, a population of string is initialised for the system. The fitness function determines the fitness of the strings based on the distance parameter. Again, a sub-population is considered based on the fitness of the strings. Mutation

Figure 1: Overview of Evolutionary Approach

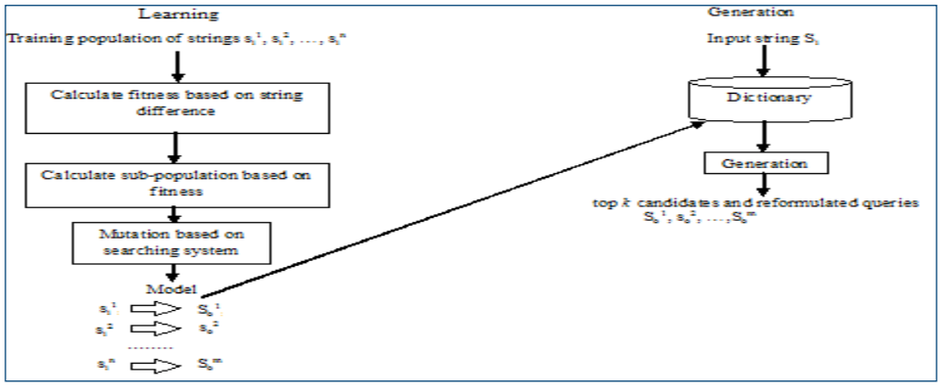


Figure 2: Overview of Rule-Based Approach

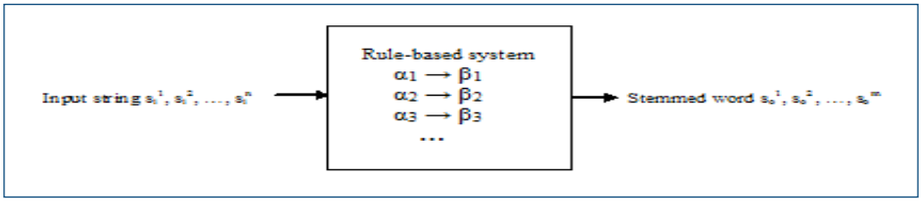
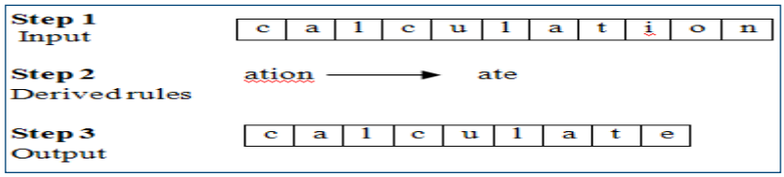


Figure 3: Example of Rule Derivation



is performed based on searching system. Then the model of string transformation consists of a set of input strings with the corresponding set of output strings, $s_i \rightarrow s_o$. For various applications, we consider character-level or word-level transformations. In the generation process, given an input string, the generation system produces the top *k* candidates and the reformulated queries aided by a dictionary and depending on the model.During the generation process, likely candidates or schema are generated. Within a chromosome, a schema is a pattern defined by fixing the values of specific chromosome loci. The set of chromosomes that indicates any pattern is said to be a schema. The schema representations in high fit chromosomes have more expectations to propagate through successive iterations of a population as the evolutionary approach proceeds gradually.

Let S be a schema and nS(j) be the number of chromosomes that belongs to S in a population j of proceeding evolution process. This model is inspired by the schema theorem and so the expectation of the number of chromosomes belonging to S in population j+1, denoted as nS(j+1), is given by the formula,

lS

$$nS(j+1) = FS(j)\, nS(j) 1 - pC1 - pn^{oS} \quad (8)$$

l - 1

where FS(j) is the relative fitness of S, pC is the crossover probability, and pn is the mutation probability. It allows determiningthe paths in which patterns are likely to propagate. So, one may design the pattern and check the effectiveness of it on the evolutionary approach. The schemata which have fitness greater than one will be present in the successive population and the fittest ones remains as a result in the final output.

The rule-based system consists of a set of rules. A rule is formally represented as $\alpha \rightarrow \beta$ which indicates the task of replacing substring α of the input string with the string β in the rule set, where $\alpha, \beta \in \{ s|s = t \}$ and $t \in \sum^*$ is the set of all possible strings over the alphabets. Given an input string, the system uses the rule set and generates the corresponding stemmed output. For various scenarios, we consider character-level or word- transformations and can apply word-level or character-level rules. There is a constraint of the number of that is applicable to a string,here we have considered one at a time. Figure 3 demonstrates the rule derivation for a word-level transformation. Given an input string "calculation", the substring "ation" is extracted and converted into "ate" based on the rule-based system which is transformed into "calculate". If the input string s_i is transformed to the output string s_o by the application of a set of rules, then the rule set is implied to form a transformation for the string pair s_i and s_o.

In the evolutionary approach, the model consists of a set of input strings and corresponding set of output strings, the population is driven from a large data size dictionary, and the generation system produces the top *k* candidates and the reformulated queries based on ranking. In the rule-based approach, the model utilises the rule-based system and generates the desired output.

3.2. String Generation

In the process of string generation, given an input string s_i, it generates the top *k* likely candidate s_o. During the evolutionary process, the fittest candidates are chosen from each successive population and the weaker candidates are pruned. The survivals of the process prove to be the final output and needs to be ranked based on their fitness. Therefore, a scoring function is needed to assign ranking to the candidates of output string s_o given an input string s_i.The score function is as follows:

$$score(s_o \mid s_i) = \max \sum_{R(s_i, s_o) F \in R(s_i, s_o)} F \quad (9)$$

wheres_i and s_o are input and output strings respectively, $R(s_i, s_o)$ denotes a string transformation that can convert s_i to s_o, and F is the fitness of the successive population. A candidate is ranked high, if it has a higher score. The score is calculated as a summation of the fitness for each transformation.

4. String Transformation Algorithms

In this section, we demonstrate some of the string transformation algorithms like evolutionary approach, fitness, string distance, and word stemming. For the evolutionary approach we mined the dictionary whereas for the rule-based approach, a dictionary is not utilised.

4.1. Evolutionary Approach

The evolutionary approach is a method to search for optimal solution in a population of candidates. It uses some standard components thus enhances reusability. The important components are population, fitness function, and the evolution scheme. The procedure is depicted in Algorithm 1.

Algorithm 1: Evolutionary Approach

Input: User input String (S_{ip}) ,D(k_1,k_2,…., k_n)

Where k1,k2….k_n are keywords in dictionary

Output: Possible output strings

Method:

1: Read the input string *Sip* from the user
2: temp =””
3: k = 0
4: for all k less than length*jsipj*do
5: count = 0
6: temp+= input[k]
7: count = match temp% with the dictionary
8: if count == 0 then
9: temp = temp.substring(0,temp.length-1)
10: dr = math temp % with the dictionary
11: likelykeywords = dr
12: end if
13: end for
14: Extract likelykeywords from the dictionary
15: Extract likelykeywords from the dictionary of same size
16: Return output strings

The algorithm picks up random population of individuals and evaluates its fitness based on string distance (step 1-2). Then a termination criterion is set in which the weaker candidates are pruned and the fit candidates are passed on to successive evolutions. Again, a sub-population is considered, and then index and alphabetical searches are performed on it (step 5-6). Again a fitness function is applied on the resultant candidates and the survivors are collected.

4.2. Fitness

Fitness is the degree of goodness or the similarity between the candidates. A candidate is considered fit if it has more similarity between the individuals of population. At the beginning, all the individuals of population have random values, so the best match has a low value. As successive evolving populations are generated, the best match values also rises gradually. The procedure is depicted in Algorithm 2.

Algorithm 2: Fitness

Input: Incoming instances

Output: Candidates with less distance or best match

Method:

1: for all individuals in the population, go to step 2, 3 do
2: Calculate the best fuzzy match to the incoming individuals
3: if match is good enough or has less string distance then
4: stop the search
5: end if
6: end for
7: Fitness = best match found
8: Extract the best individual
9: Return output string

The input to the algorithm is the incoming instances and the outputs are the candidates that have less distance with corresponding to the input. For each individual the best match is calculated in accordance of the incoming individuals. If the match is good enough (>=90%), the searching process stops. Fitness is considered as the best match found and the best individuals are selected.

4.3. Edit Distance

Given two character strings s and t, the edit distance is the minimum number of operations required to transform s into t. Generally, the

operations that are permitted by it are as follows: (1) insertion of a character into a string, (2) deletion of a

character from a string, and (3) substitution of a character by another character. For example, the edit distance of the word "Saturday" and "Sunday" is three. The procedure is depicted in Algorithm 3.

Algorithm 3: Edit Distance

Input: Strings s1, s2with lengths l1, l2 respectively

Output: Minimum number of operations d, required for string transformation

Method:

```
1:  for i from 0 to l1 do
2:  d[i , 0] := i
3:  end for
4:  for j from 1 to l2 do
5:  d[0 , j] := j
6:  end for
7:  for i from 1 to l1 do
8:  for j from 1 to l2 do
9:  if s1[i] equals s2[j] then
10: cost := 0
11: else
12: cost := 1
13: end if
14: distance[i, j] = minimum ( distance[i-1, j] + 1,
    distance[i, j-1] + 1, distance[i-1, j-1] + cost
15: return distance[l1, l2]
16: end for
17: end for
```

The inputs of the algorithm are strings, of any length. While looping through the strings, if the characters are equal then zero is assigned to cost, otherwise cost equals one. The algorithm permits three types of operations which are deletion, insertion, and substitution. The minimum numbers of operations are considered and returned as a result.

4.4. Word Stemming

The word stemming conflates inflected or derived words to a stem or root. Stem is not necessarily the morphological root. There are several approaches to stemming like parts-of-speech recognition, corpus-based stemming, context sensitive stemming, suffix stripping, affix stripping, statistical measures, brute force look up etc. Here, we are concentrating on rule-based suffix stripping. The procedure is depicted in Algorithm 4.

Algorithm 4: Word Stemming

Input: Strings

Output: Stemmed word

Method:

```
1: for all individuals in the population, go to step 2, 3
   do
2: Calculate the best fuzzy match to the incoming
   individuals
3: if match is good enough or has less string distance
   then
4: stop the search
5: end if
6: end for
7: Fitness = best match found
8: Extract the best individual
9: Return output string
```

Algorithm (3) demonstrates a rule-based system to stem a word. It must be build bit by bit, trying to eradicate small number of suffixes at a time. For each kind of new suffixes that are encountered, a generalised rule must be built and checked whether it improves or degrades the stemming process. If a particular generalised rule improves the process, then it is added to the rule-based system, otherwise the rule is discarded.

5. Experimental Results

We have experimentally evaluated our method to solve three problems i.e. spelling error correction, query reformulation, and word stemming. Spelling error correction aimed at string transformation at character level whereas the string transformation in query reformulation has taken place at word level. These methods are dependent on a dictionary.Word stemming is based on suffix stripping which is a rule-bases system and does not use a dictionary. We have implemented these methods in C#.NET at front end and MySql at back end.

5.1. Word Mining

An artificial dataset is collected from web. Data have been also collected from 100 news documents collected from popular English newspaper like www.timesofindia.indiatimes.com,www.newindianexpress.com, www.thehindu.com, www.telegraphindia.com, and www.hindustantimes.com. Thus the dictionary consistsof correct spelled words and a total of 1, 47,482 records. The records comprise the words from English dictionary. Moreover, an interface is also created for inserting new terms of science, mathematics, politics, sports, entertainment, social issues, etc.

5.2. Experiments on Spelling Error Correction

The process of spelling corrections suggests correct spelling along with various other keyword options related to the search. Accuracy is vital for this process, so a large dictionary is considered over here. Here we have aggregated some of the wrong entries of the user and the correct spelling generated by this system which is shown in Table 1.

Table 1: Assembled Words for Spelling Error Correction

Misspelled	*Correct*	*Misspelled*	*Correct*
applle	Apple	computr	computer
speling	Spelling	dictionry	dictionary
jornal	Journal	allgorithm	algorithm
artikle	Article	managemnt	management
buillding	Building	candedate	candidate
efficeent	Efficient	approximaate	approximate

From Table 1, it seems the spellings that are generated by this system, in correspondence to misspelled words are true corrections. The experimental result of spelling error correction is shown in Figure 4.

The experimental shows that when a misspelled word “country” is encountered, it is transformed into “country” which seems to be a true correction. Similarly, it has been tested with large number of words, and each time the outcome is a true correction.

5.3. Experiments on Candidate Generation

With a user input, top *k* candidate sets are generated with respect to the input. The top *k* candidate sets comprises the most relevant keyword whereas the rest have a chance of usability.We have compared our system with the edit distance algorithm used in log-linear model [1] with respect to number to relevant top *K* candidate generation. To the best of our knowledge, this system outperforms in the generation process. Dictionary is utilised for this process. In Figure 5we demonstrate the experimental result of top *k* candidate generation of both the approaches.

By the above experimental results, it seems that our method generates more number of relevant candidates. It is conducted by utilising a keyword. Similarly, it has been tested with several other keywords, and most of the time more number of candidates were generated by our process.

It seems that that our method generates more number of candidates compared to the baseline as shown in Figure 6. As per our knowledge, the generated candidates are likely to be compatible with the input string. Next, we recorded

Figure4: Experiments on Spelling Error Correction

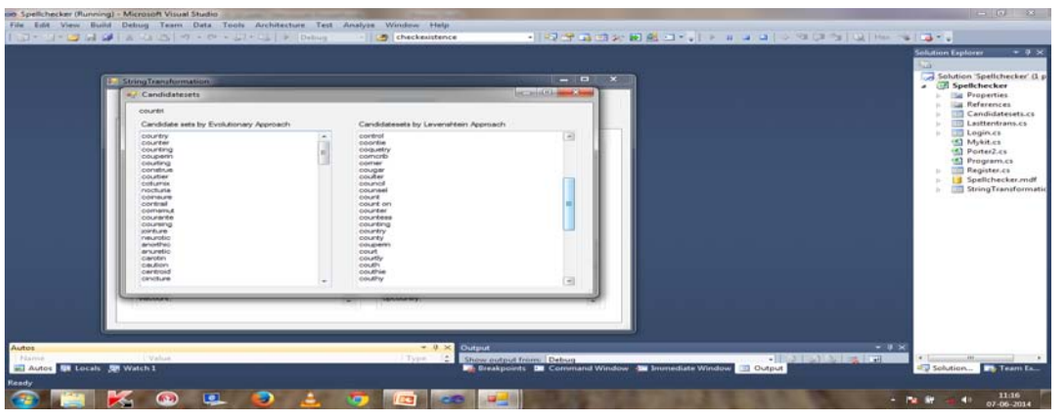


Figure 5: Top *k* candidate generation by our process (M1) and the baseline (M0).

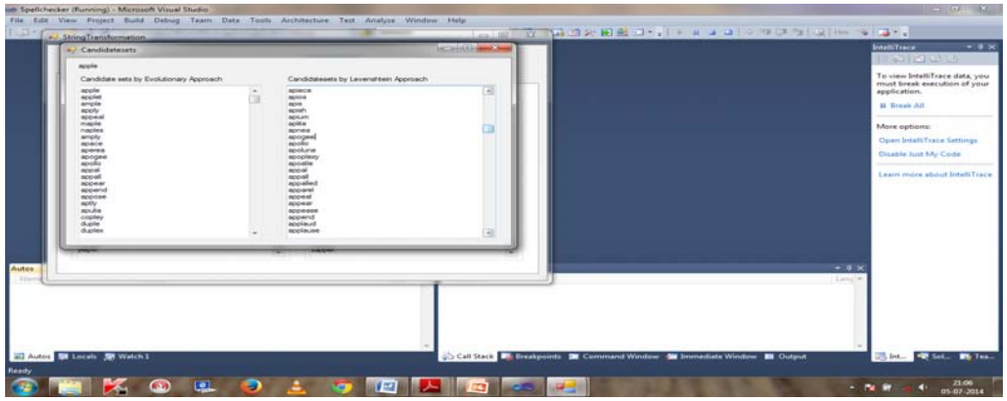
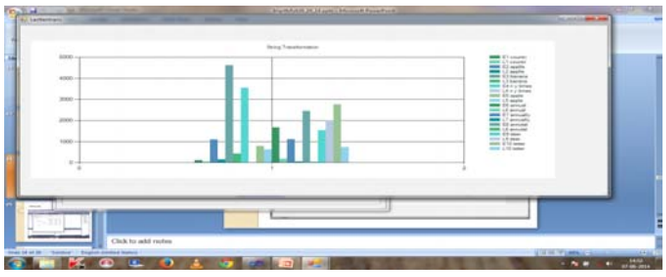


Figure 6: Comparison between Baseline (M0) and Our Method (M1) with respect to Number of Candidate Generation

the last ten transactions of input keywords and compared the candidate generation process.

Figure 7: Comparison between Baseline (M0) and Our Method (M1) with respect to Number of Candidate Generation of Last Ten Transactions of keywords



It can be observed from Figure 7 that the candidates generated for the last ten transactions shows better result i.e. more number of candidates are generated by our method.

5.3. Experiments on Query Reformulation

Experiments are conducted for query reformulation, in which the input is a short form of a text and the output is an equivalent full form. This directs to a better ranking of a document. Dictionary is utilised in this experiment. Here we have collected some of the short forms of a text which are transformed into similar elongated text during the experiment as shown in Table 2.

Table 2: Assembled Queries for Query Reformulation

Query	*Reformulated Query*
SBI	State Bank of India
N Y Times	New York Times
TOI	The Times of India

Figure 8: Experiment on Query Reformulation

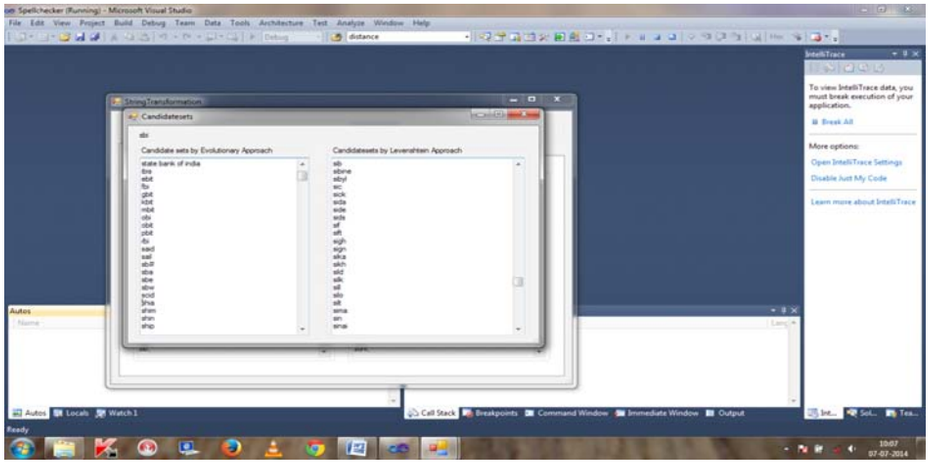


Figure 9: Number of candidates generated during query reformulation

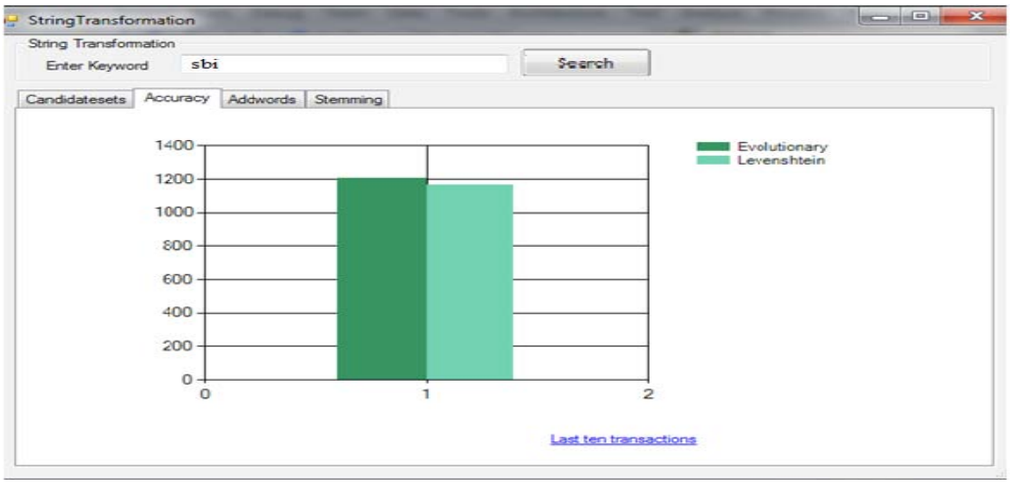


Figure 8 shows the process of query reformulation in which an input string "sbi" is entered which is transformed into "state bank of india". Therefore, if a document doesn't have the keyword "sbi" then also it can be ranked high if it have "state bank of india". The experiment is aided by the dictionary. Figure 9 shows the number of candidates that are generated during the query reformulation process. It seems that out method generates a slight more number of candidates.

5.4. Performance Analysis

The performance of this system is recorded and analyzed with other models. For experimental setup, a gold standard data set is constructed for evaluation. The training and the testing set consists of 8, 318 and 2, 889 terms respectively collected from queries which is shown in Table 3.

Table 3: Training and Testing Set

	#queries	*#terms*
Training Set	2500	8318
Testing Set	1090	2889

In the next section, we will show the evaluation metrics that utilises the training and testing data set for performance analysis.

5.4.1. Evaluation Metrics

Here, we have considered the following five types of evaluation metrics.

- Accuracy

Accuracy is computed for this system which is the true corrections that are made. It is calculated by the number of correct outputs by the spelling error correction system divided by the total number of queries in the test set. We compared our method with the baseline along with other models with respect to accuracy, which is the ratio of the true corrections in the candidate generation process.It is mathematically expressed as,

1 (true correction in the top k) (10)

Accuracy of k =

0 (otherwise)

- Precision

It is calculated as the number of correctly generated candidates for misspelled keywords divided by the total number of generated candidates made by the system.

- Recall

It is calculated as the number of correctly generated candidates for misspelled keywords divided by the total number of misspelled keywords in the test set.

- F-measure

It is the harmonic mean of precision and recall. It measures accuracy using the statistic precision P and recall R. The formula is as follows,

$$F = 2PR / (P + R). \quad (11)$$

- G-measure

It is the geometric mean of precision and recallIt measures accuracy using the statistic precision and recall. The formula is as follows,

$$G = \sqrt{precision} .recall \quad (12)$$

One metric alone cannot judge the performance of the system. In most system tradeoff between precision and recall are done. There are no constraints that the increase of one may lead to the increase of another measure. Often it can be noticed that with the rise of certain measure, there exists a drop in another measure.

5.4.2. Overall Results

By utilising the procedures described in previous sections, we conducted several experiments to evaluate the performance of the models. We have considered the default setting as discussed above for the experiments. The detailed results are shown in Table 4.

Table 4: Performance results of models (in percentages)

	Precision	*Recall*	*Accuracy*	*F-Measure*	*G-Measure*
M0	90.8%	80.48%	97.24%	85.32%	87.05%
M1	91.52%	83.07%	99.08%	87.0%	87.18%

From Table 4, we can observe that M1 has a significant improvement of precision of about 0.72 percentages over M0. Similarly, there is a boost of recall of about 2.59 percentages over the baseline. The accuracy has increased by 1.84 percentages in our model. It indicates that this model is more efficient in candidate generation, spelling error correction, and query reformulation. Similarly, there is an improvement of 1.68 and 0.13 percentages for F-measure and G-measure respectively.

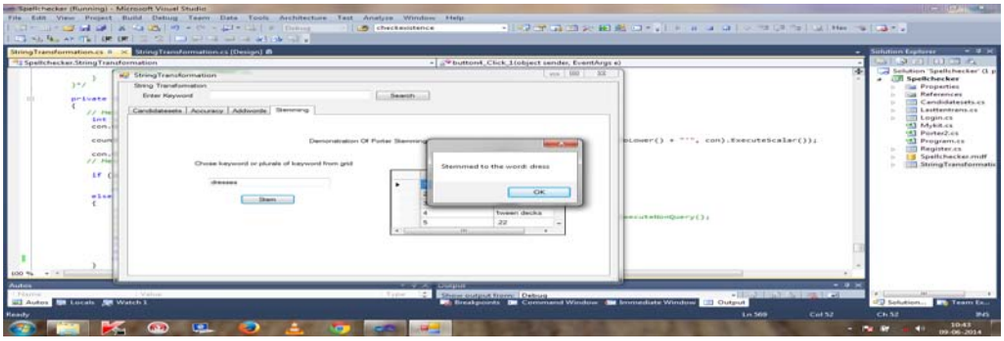
5.5. Experiments on Word Stemming

Here, word stemming is implemented by using rule-based system which doesn't utilise a dictionary. Given an input string, the system generates its equivalent stemmed word. Here we have collected few of the words along with its equivalent stemmed word which isused during the experiment are shown in Table 5.

Table 5: Assembled Words for Performing Word Stemming

Word	*Stemmed Word*
Apples	apple
calculation	calculate
countries	country
Boxes	box
summarization	summarize
continuing	continue
jumping	jump
Started	start
Helpful	help
consciousness	conscious

Figure 10: Experiments on Word Stemming



Our experiments are based on suffix stripping. Figure 10 demonstrates that when a user enters a keyword, it suffixes are stripped and an equivalent stemmed word is displayed. We have collected data from various sources like web, newspapers, books, etc. and tested the system several times. The outcome of it is predictable and a true stemmed word.

6. Conclusions and Future Work

In this paper, we have proposed a unique and novel model for string transformation. First, we adopted the evolutionary approach which addresses the applications like spelling error correction to correct the incorrect user entries, top *k* candidate generation for better suggestion for a keyword, and query reformulation. This model is aided with a large dictionary with data collected from web, newspapers, books, etc. Next, we considered five evaluation metrics viz. precision, recall, accuracy, F-measure, and G-measure which are employed for comparing out model with the baseline. In all the circumstances, our model seemed to perform better. Second, we adopted the rule-based approach which addresses the application of word stemming. Here, we specifically demonstrated suffix stripping. A corpus is not used for this process. For testing purpose, data is collected from web, newspapers, books, etc.As per our knowledge, all the applications are well tested and the outcomes are true.

There is still potential for further research in this direction. The algorithms of spelling error correction, top *k* candidate generation, query reformulation, and word stemming can be improved so that it can offer better results in terms of precision, recall, accuracy, F-measure, and G-measure. To improve the algorithms, one may utilise the machine translation techniques. Query reformulation may be termed as a machine translation problem, in which the text expansion may be done depending on the type of application. Further, there is a chance to build a unified model that can return the page ranking information, pronunciation generation, and word transliteration.The model can be implemented to perform string transformation for different types of languages. Conversion of a query form one language into an equivalent another language is also a vast area of research.

References

Ahmad, F.,&Kondrak, G. (2005).*Learning a Spelling Error Model from Search Query Logs*. Paper presented at the Conference on Human Language Technology and Empirical Methods in Natural Language Processing(pp. 955-962).

Arasu, A., Chaudhuri, S., &Kaushik, R. (2009).*Learning String Transformations from Examples*.Proceedings of the VLDB Endowment, August, 2, 514-525.

Behm, A., Ji, S., Li, C., & Lu, J. (2009).*Space-constrained Gram-based Indexing for Efficient Approximate String Search.* Paper presented at the 2009 IEEE International Conference on Data Engineering, ser. ICDE '09. Washington, DC, USA: IEEE Computer Society(pp. 605-615).

Brill, E. (1995).Transformation-based error-driven learning and natural language processing: A case study in part of speech tagging.*Journal of Computational Linguistics*, December, 21(4), 543-565.

Brill, E., & Moore, R. C. (2000).*An Improved Error Model for Noisy Channel Spelling Correction*. Paper presented at the 38th Annual Meeting on Association for Computational Linguistics, ser. ACL `00. Morristown, NJ, USA: Association for Computational Linguistics (pp. 286-293).

Chen, Q., Li, M., & Zhou, M. (2007). *Improving Spelling Query Correction using Web Search Results*. Paper presented at the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, Prague, 2007, ©2007 Association for Computational Linguistics (pp. 181-189).

Damerau, F. J. (1964). *A Technique for Computer Detection and Correction of Spelling Errors.*Paper presented at the Communications of the Association for Computing Machinery, March, 171-176.

Dreyer, M., Smith, J. R., & Eisner, J. (2008).*Latent-variable Modeling of String Transductions with Finite-state Methods.*Paper presented at the Conference on Empirical Methods in Natural Language Processing, ser. EMNLP `08. Stroudsburg, PA, USA: Association for Computational Linguistics (pp. 1080-1089).

Duan, H., & Hsu, B. J. (2011).*Online Spelling Correction for Query Completion.* Paper presented at the 20th International Conference on World Wide Web, ser. WWW `11. New York, NY, USA: ACM (pp. 117-126).

Golding, A. R., & Roth, D. (1999).A winnow-based approach to context-sensitive spelling correction. *Machine Learning*, February, 34(1-3), 107-130.

Gupta, V. (2014). Hindi rule based stemmer for nouns. *International Journal of Advanced Research in Computer Science and Software Engineering*, January, 4(1), 62-65.

Guo, J., Xu, J., Li, H., & Cheng, X. (2008).*A Unified and Discriminative Model for Query Refinement*. Paper presented at the SIGIR `08, Singapore.

Hadjieleftheriou, M., & Li, C. (2009). Efficient Approximate Search on String Collections. Proceedings of the VLDB Endowment, August, 2(2), 1660-1661.

Hicham, G., Abdallah, Y., & Mustapha, B. (2012). Introduction of the weight edition errors in the levenshtein distance.*International Journal of Advanced Research in Artificial Intelligence*, 1(5), 30-32.

Huang, J., &Efthimiadis, E. N. (2009).*Analyzing and Evaluating Query Reformulation Strategies in Web Search Logs.* Paper presented at the 18th ACM Conference on Information and Knowledge management (pp. 77-86).

Islam, A., &Inkpen, D. (2009).*Real-word Spelling Correction using Google Web 1T 3-grams*. Paper presented at the 2009 Conference on Empirical Methods in Natural Language Processing (pp. 1241-1249).

Islam, A., &Inkpen, D. (2011).*Correcting Different Types of Errors in Texts.* Paper Presented at the 24th Canadian Conference on Advances in Artificial Intelligence (pp. 192-203).

Ji, S., Li, G., Li, C., & Feng, J. (2009).*Efficient Interactive Fuzzy Keyword Search*. Paper presented at the 18th International Conference on World Wide Web (pp. 371-380).

Jones, R., Rey, B., Madani, O., & Greiner, W. (2006). *Generating Query Substitutions.* Paper Presented at the 15th International Conference on World Wide Web (pp. 387-296).

Li, C., Lu, J., &Lu, Y. (2008).*Efficient Merging and Filtering Algorithms for Approximate String Searches.*Paper presented at the 2008 IEEE 24th International Conference on Data Engineering (pp. 257-266).

Li, C., Wang, B., & Yang, X. (2007).*Vgram: Improving Performance of Approximate Queries on String Collections using Variable-length Grams*. Paper Presented at the 33rd International Conference on Very Large Data Bases (pp. 303-314).

Li, M., Zhang, Y., Zhu, M., & Zhou, M. (2006). *Exploring Distributional Similarity based Models for Query Spelling Correction.* Paper Presented at the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics (pp. 1025-1032).

Majgaonker, M. M., & Siddiqui, T. J. (2010). Discovering Suffixes: A Case Study for Marathi Language. *International Journal on Computer Science and Engineering*, 2(8), 2716-2720.

Majumder, P., Mitra, M., Parui, S. K., Kole, G., Mitra, P., &Datta, K. (2007). YASS: Yet another Suffix Stripper. *Association for Computing Machinery Transactions on Information Systems*, 25(4), 18-38.

McCallum, A., Bellare, K., &Pereira, F. (2005).*A Conditional Random Field for Discriminatively-trained Finite-state String Edit Distance*. Paper presented at the 21st Conference on Uncertainty in Artificial Intelligence (pp. 388-395).

McCallum, J. (2005). Genetic algorithms for modeling and optimization.*Journal of Computational and Applied Mathematics*, 184, p. 205-222.

Okazaki, N., Tsuruoka, Y., Ananiadou, S., &Tsujii, J. (2008).*A Discriminative Candidate Generator for String Transformations.* Paper Presented at the Conference on Empirical Methods in Natural Language Processing (pp. 447-456).

Oncina, J.,&Sebban, M. (2005).*Learning Unbiased Stochastic Edit Distance in the Form of a Memory-less Finite-state Transducer*.Workshop on Grammatical Inference Applications: Successes and Future Challenges.

Pandey, A. K., &Siddqui, T. J. (2008). *AnInsupervised Hindi Stemmer with Heuristic Improvements*. Paper Presented at the Second Workshop on Analytics for Noisy Unstructured Text Data, 303, 99-105.

Ramanathan, A., & Rao, D. D. (2003). *A Light Weight Stemmer for Hindi*.Workshop in Computational Linguistics for South-Asian Languages, EACL.

Ristad E.S., &Yianilos, P.N. (1997).*Learning String Edit Distance*.Princeton University Research Report.

Tejada, S., Knoblock, C. A., & Minton, S. (2002).*Learning Domain-Independent String Transformation Weights for High Accuracy Object Identification*.Paper Presented at the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (pp. 350-359).

Toutanova, K., & Moore, R. C. (2002).*Pronunciation Modeling for Improved Spelling Correction.* Paper Presented at the 40th Annual Meeting of the Association for Computational Linguistics (ACL), Philadelphia (pp. 144-151).

Vernica, R., & Li, C. (2009).*Efficient Top-k Algorithms for Fuzzy Search in String Collections.* Paper Presented at the First International Workshop on Keyword Search on Structured Data (pp. 9-14).

Wang, X., &Zhai, C. (2008).*Mining Term Association Patterns from Search Logs Ffor Effective Query Reformulation*.Paper Presented at the 17th ACM Conference on Information and Knowledge Management (pp. 479-488).

Wang, Z., Zu, G.,Li, H.,&Zhang, M. (2013).A Probabilistic Approach to String Transformation. *IEEE Transactions on Knowledge and Data Engineering*(pp.99).

Wagner, R. A.,& Fischer, M. J. (1980).The string-to-string correction problem. *Journal of Computer and System Sciences*, 20(1), 18-31.

Wagner, R. A. & Fischer, M. J. (1974).The string-to-string correction problem. *Journal of the ACM*, 21(1), 168-173.

Whitelaw, C., Hutchinson, B., Chung, G. Y., & Ellis, G. (2009).*Using the Web for Language Independent Spellchecking and Auto-correction.* Paper Presented at the 2009 Conference on Empirical Methods in Natural Language Processing (pp. 890-899).

Yang, Z., Yu, J., &Kitsuregawa, M. (2010).*Fast Algorithm for Top-K Approximate String Matching*.Paper Presented at the Twenty-Fourth AAAI Conference on Artificial Intelligence (AAAI-10).

Yang, X., Wang, B., &Li, C.(2008).*Cost-based Variable-Length-Gram Selection for String Collections to Support Approximate Queries Efficiently.*Paper Presented at the 2008 ACM SIGMOD International Conference on Management of Data(pp. 353-364).