International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395 -0056
Volume: 02 Issue: 05 | Aug-2015 www.irjet.net p-ISSN: 2395-0072

REFORMATION OF QUERY BASED APPROACH IN SEARCH ENGINE

Rohini.M

MTech, Computer Science and Engineering, Lourdes Matha College of Science and Technology, Kerala, India

---

Abstract - ***String Transformation is an essential problem in many applications. Some of which are data mining, natural language processing, and bioinformatics. In natural language processing, pronunciation generation, spelling error correction, word transliteration, and word stemming can all be formalized as string transformation. String transformation can also be used in query reformulation and query suggestion in search. In data mining, string transformation can be employed in the mining of synonyms and database record matching. In string transformation, the input strings are transformed into k most likely output strings by applying a number of operators. Each operator is a transformation rule that defines the replacement of a substring with another substring. String transformation can be conducted at two different settings, depending on whether or not a dictionary is used. Query reformation involves rewriting the original query with its similar queries and enhancing the effectiveness of search. Most existing methods manage to mine transformation rules from pairs of queries in the search logs. One represents an original query and the other represents a similar query (e.g., hotmail sign-on, hotmail sign-up).The weights of the transformation rules are calculated based on log likelihood ratio. A query dictionary is used in this case mined contextual substitution patterns and tried to replace the words in the input query by using the patterns. They created a set of candidates that each differ from the input query in one word. The existing methods mainly focused on how to extract useful patterns and rank the candidates with the patterns, while the models for candidate generation are simple.***

Key Words: *String Transformation, Spelling Error Correction, Query Reformulation.*

1. INTRODUCTION

Main task is to understand natural language given by the end user first, then must be processed to achieve human-computer interaction desired output and allowing computer systems to get the meaning of the transformations input. String human or natural language have been proposed as a solid game mechanism for such variations. However, in many areas, the identification of a suitable set of transformations is difficult that the space of possible transformations is great.

Many problems in natural language processing, data mining, information retrieval, and bioinformatics can be formalized as string transformation, which is a task as follows. Given an input string, the system generates the *k* most likely output strings corresponding to the input string. A novel and probabilistic approach to string transformation, which is applied to correction of spelling errors in string. The approach includes the use of a log linear model, a method for training the model, and an algorithm for generating the top *k* candidates, whether there is or is not a predefined dictionary. The proposed method is applied to reformulation of queries by word by word transformation in search engine. Query reformulation involves rewriting the original query with its similar queries and enhancing the effectiveness of search.

String transformation, which is an essential problem in many applications. In natural language processing, pronunciation generation, spelling error correction, word transliteration, and word stemming can all be formalized as string transformation. String transformation can also be used in query reformulation and query suggestion in search. In data mining, string transformation can be employed in the mining of synonyms and database record matching. As many of the above are online applications, the transformation must be conducted not only accurately but also efficiently.

String transformation can be defined in the following way. Given an input string and a set of operators, we are able to transform the input string to the k most likely output strings by applying a number of operators. Here the strings can be strings of words, characters, or any type of tokens. Each operator is a transformation rule that defines the replacement of a substring with another substring. The likelihood of transformation can represent similarity, relevance, and association between two strings in a specific application. Although certain progress has been made, further investigation of the task is still necessary, particularly from the viewpoint of enhancing both accuracy and efficiency.

String transformation can be conducted at two different settings, depending on whether or not a dictionary is used. When a dictionary is used, the output strings must exist in

the given dictionary, while the size of the dictionary can be very large. Without loss of generality, specifically study correction of spelling errors in queries as well as reformulation of queries in web search. In the first task, a string consists of characters. In the second task, a string is comprised of words. The former needs to exploit a dictionary while the latter does not. Correcting spelling errors in queries usually consists of two steps: candidate generation and candidate selection. Candidate generation is used to find the most likely corrections of a misspelled word from the dictionary.

In such a case, a string of characters is input and the operators represent insertion, deletion, and substitution of characters with or without surrounding characters, for example, "a"->"e" and "lly"->"ly". Obviously candidate generation is an example of string transformation. Note that candidate generation is concerned with a single word; after candidate generation, the words in the context (i.e., in the query) can be further leveraged to make the final candidate selection .Query reformulation in search is aimed at dealing with the term mismatch problem. For example, if the query is "NY Times" and the document only contains "New York Times", then the query and document do not match well and the document will not be ranked high. Query reformulation attempts to transform "NY Times" to "New York Times" and thus make a better matching between the query and document. In the task, given a query (a string of words), one needs to generate all similar queries from the original query (strings of words). The operators are transformations between words in queries suchas "tx"->"texas" and "meaning of"->"definition of.

2. LITERATURE SURVEY

According to winnow-based approach for context-sensitive correction of spellings**,** for **a** large variety of problems there is a requirement of characterization of linguistics. It focus on the concepts of techniques that are used for correcting the strings errors in of miss-placed so as to avoid these kinds of problems while searching on the net the earlier techniques used the winnow based approaches for correcting those errors. In this effort we are acquiring the properties of those methods to avoid such mistakes. if we consider the example string " to", suppose if we place the string „too" in place of "to" it leads to difference in meanings and may the sentence leads to wrong. This is the task of fixing spelling errors that happen to result in valid words, such as substituting to for too, casual for causal, and so on. A unified and discriminative method for query modification is a method used for providing the alternative substrings in place of original string, in the case when the original string is found to be misspelled.

3. EXISTING SYSTEM

String transformation has many applications in data mining, natural language processing, information retrieval, and bioinformatics. String transformation has been studied in different specific tasks such as database record matching, spelling error correction, query reformulation and synonym mining. The major difference between our work and the existing work is that we focus on enhancement of both accuracy and efficiency of string transformation. Spelling error correction normally consists of candidate generation and candidate selection. Candidate generation is usually only concerned with a single word. Candidate generation is usually only concerned with a single word. Other methods learn weighted edit distance to enhance the representation power [18], [19], [20], [21]. Edit distance does not take context information into consideration. For example, people tend to misspell "c" as "s" or "k" depending on context, and a straightforward use of edit distance cannot deal with the case. To address the challenge, some researchers proposed using a large number of substitution rules containing context information (at character level). For example, Brill and Moore [5] developed a generative model including contextual substitution rules. Toutanova and Moore [22] further improved the model by adding pronunciation factors into the model. Duan and Hsu [23] also proposed a generative approach to spelling correction using a noisy channel model. They also considered efficiently generating candidates by using a trie. In this paper, we propose using a discriminative model. Both approaches have pros and cons, and normally a discriminative model can work better than a generative model because it is trained for enhancing accuracy. Since users' behavior of misspelling and correction can be frequently observed in web search log data, it has been proposed to mine spelling-error and correction pairs by using search log data. The mined pairs can be directly used in spelling error correction. Methods of selecting spelling and correction pairs with a maximum entropy model [24] and similarity functions [25], [26] have been developed. Only high frequency pairs can be found from log data, however. In this paper, we work on candidate generation *at the character level*, which can be applied to spelling error correction for both high and low frequency words.

Step 1: Dictionary Creation

String transformation can be conducted at two different settings, depending on the dictionary. When dictionary is used, the output string must exist in the given dictionary, while the size of the dictionary can be very large and the response time is very short. To create a dictionary it follows these steps: add words to the dictionary, arrange

the words alphabetically and add description related to that word.

Step 2: Candidate Generation

Misspelling is a common phenomenon in search engine queries. Misspelling occurs for a variety of reasons. When typing quickly users may add or drop letters unintentionally. Accidently hitting an adjacent key on the keyboard make spelling mistakes. In addition to typographical errors, some errors result from the challenge of spelling itself with inconsistent spelling rules, ambiguous word breaking boundaries, and constant introduction of new words. To assist users in expressing information needs, it is important for search engines to automatically generate corrections for misspelled queries. Candidate generation is concerned with a single word and based on rule based approach. Edit method is the typical method which is used here.

Step 3: Candidate Selection

In candidate selection, top k pruning technique is used to select the top k candidates. The string generation problem amounts that of finding top k output strings given the input string. The algorithm uses top k pruning strategy to eliminate unlikely paths and thus improve efficiency. Finally calculate the weight of the generated candidates and thus produces a rank list. Based on this the top k candidates has developed.

Step 4: Query Reformulation

Query reformulation is the process of rewriting the original query with its similar queries and enhancing the effectiveness of search. It involves evaluating a user's input and expanding the search query to match additional documents. Some of the reformulation methods are add or remove words; word substitution, acronym expansion. The existing methods mainly focus on how to extract useful patterns and rank the candidates with the patterns while the models for candidate generation are simple. This method made the string transformation efficient and accurate by word by word transformation.

Step 5: String Mining

The user's has to download its meanings also he/she can download its substrings and reverse etc. Also check the given string which is present in the bunch of strings, if its present the result will be "String Found" otherwise "String Not Found". System generates k most likely output strings based on the input string.

4. PROPOSED SYSTEM

Query reformulation is the process of rewriting the original query with its similar queries and enhancing the effectiveness of search. It involves evaluating a user's input and expanding the search query to match additional documents. Some of the reformulation methods are add or remove words; word substitution, acronym expansion. The existing methods mainly focus on how to extract useful patterns and rank the candidates with the patterns while the models for candidate generation are simple. This method made the string transformation efficient and accurate by word by word transformation. The goal of query reformulation is to define a new query, starting from the initial one, which is able to lead to improved retrieval results. What exactly "good search results" means can differ according to context in which the search is used, but it usually refers to the relevant documents being as close as possible to the top of the search results list. Figure 1 shows the architecture of the proposed system.

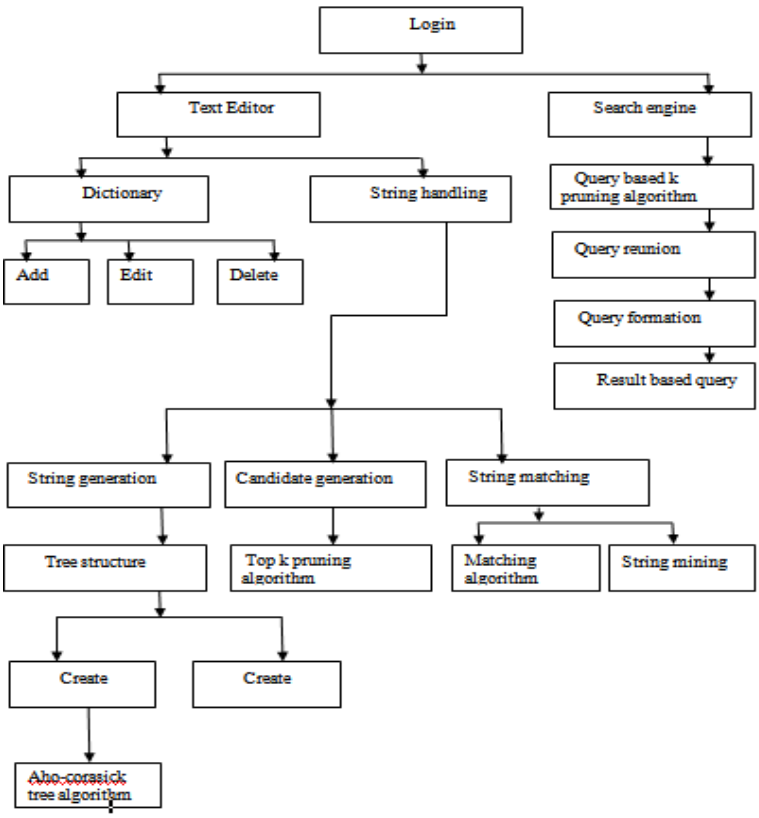

Fig -1: Architecture of the proposed system

4.1 Query transformation

Can be described as follows; if given one input query is 'q' and a set of operators included in the chain, we can transform the input query to the 'n' most probable output queries .Queries can be strings of words, sentence ,characters or any type chips. Each operator is a

transformation rule or semantic definable replace a query with another similar query.

Query based top K pruning algorithm

The query generation problem amounts to that of finding the top *k* output queries given the input query.

Input: rule index which will specify the rules and semantics I_r, input string , candidate number 'n'

Output: top 'n' output query

Begin
Step1: Find all rules applicable to *s* from I_r with k purning algorithm
Step 2 *: minscore= 0*
Step 3:While check various possible paths
Step 4: do
Pickup a possible
Step 5: if score <minscore
Step 6: then
Step 7: continue
Step 8: similarly check with all the possible outcomes
Step 9: Let n be the number of string in the query;
Step 10: Let l[0..n − 1] be the lists of ids for the query String;
Step 11: Let g ← 1 be the frequency threshold;
Step 12: Insert the top element on each list to a heap, H;
Step 13: Topk ← φ;
Step 14: while H is not empty do
Step 15: Let T be the top element on H;
Step 16: Pop from H those elements equal to T;
Step 17: Let p be the number of popped elements;
Step 18: if p ≥ g then
Step 19: if Score(T) > Score(k th in Topk) then
Step 20: Insert T into Topk and pop the last one;
Step 21: Recompute threshold g;
Step 22: if g > n
Step 23: then break;
Step 24: end if
Step 25: Push next element (if any) of each popped list to H;
Step 26: else
Step 27: Pop additional g − p − 1 elements from H;
Step 28: Let T′ be the current top element on H;
Step 29: for each of the g − 1 popped lists do
Step 30: Locate its smallest element E ≥ T ′ (if any);
Step 31: Push E to H;
Step 32: end for
Step 33: end if
Step 34: end while
Step 35: Return the elements in Topk
Step 35: if any candidate will have min score
Step 36: then Remove candidate with minimum score
Step 37: Stop

4.2 Dataset query generation

Reformulation of queries in search is aimed at addressing the problem of maturity mismatch. For example, if the query is " apjabdulkalam " and the document only contains " Famous books by Dr.APJ Abdul Kalam ", the query and the document does not fit well and the document does not classified high. Query Reformulation tries to transform " apjabdulkalam " the " Famous books by Dr.APJ Abdul Kalam " and therefore make a better match between the query and the document. In the task, given a query that is needed to generate all similar queries from the original query. Query Reformulation again involves writing the original query string or its consultations or similar words match the dictionary and improve the search efficiency. Most existing methods for managing transformation rules of the mine from pairs of queries in search logs. Given two words "cat", "FAR" to determine whether you can get from the first to the second through simple transformations of valid words ,for example 1 transformation receives CAT to CAR T change R, then another takes you from the car to change C to F FAR all are valid English words.

5. PERFORMANCE ANALYSIS

This work is used to solve two problems, spelling error correction of queries and reformulation of queries in web search. The difference between the two problems is that string transformation is performed at a character level in the former task and at a word level in the latter task. Aho-Corasick algorithm is used to generate words**.** The effect of using the Aho-Corasick algorithm is tested here. The time complexity of Aho- Corasick algorithm is determined by the length of query word plus the number of matches. Also examined how the number of matches on query word changes when the size of the rule set increases. The number of matches is not largely affected when the size of the rule set increases in the rule index. It implies that the time for searching applicable rules is close to a constant and does not change much with different numbers of rules.
This method has been applied to two applications, spelling error correction of queries and reformulation of queries in web search. Experiments have been conducted between this method and the baselines methods. The result shows that this method performs consistently better than the baselines in terms of accuracy. Moreover, the accuracy of this method is constantly better than the baselines in different experiment settings, such as size of the rule set, maximum number of applicable rules, and dictionary size. Experiments on efficiency have been conducted with running time as the measure. The running times of this method are smaller than those of the baselines, which indicate that the pruning strategy in this method is very efficient.

5.1 Spelling error correction

In spelling error correction enters a string and then enters a space. If the input string is not correct then it displayed as red. And combination of characters are generated and displayed as tree structure. Then matching words are listed. The accuracy of the spell check is determined as the ratio of total combinations of characters to the matching candidates.

Accuracy= Total combinations of characters/ matching candidates.

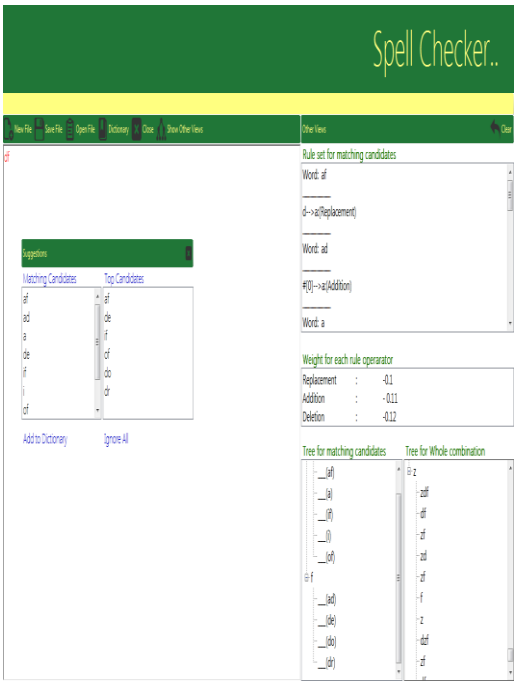

Fig -2: Spelling error correction

For example consider this image. Here after enter the space combinations of letters are generated that is shown. Here first count the total combinations of letters and then count the matching candidates. Take the ratio of total combinations to matching candidates. That is the accuracy. And the time required to show the matching candidates after enter space is the time delay.

Table -1: Examples of spell check contain two letter input string

Data set input	Total combinations	Matching candidates	Time delay in seconds	Accuracy
Df	413	8	.11	51.625
lm	173	3	.23	57.66
cd	296	4	.36	74
Bv	238	4	.16	59.5

The table 1 shows the accuracy of spell check. Here input is two letter words. These words are not correct. So combinations of characters are generated. That is the total combinations. The generation is done by using Aho-Corasick algorithm. The time delay for the generation algorithm is shown in the table. That is the time required to generate combinations of characters after enter the space. Then top k pruning algorithm is performed to select matching candidates. Count number of matching candidates and total combinations of characters. Then calculate the accuracy. From the above examples average accuracy of the system when enter input with two letters.
Average accuracy = total accuracy/ 4 = 60.696%
Time delay is the time required to display the matching candidates after enter input string and space. If time delay is less then the system has high efficiency.
The average time delay= total time delay/4 = 0.215 second
The system only needs minimum time. So it is efficient.

Table -2: Examples of spell check contain three letter input string

Data set input	Total combinations	Matching candidates	Time delay in seconds	Accuracy
Asd	938	13	.58	74.46
Crn	976	17	1.5	57.41
Far	848	14	.91	60.57
Ter	1075	20	.87	53.75

In this table input is three letter strings. So combinations of letters are generated.
Average accuracy = 61.547
Average time delay = 0.965 second

Table -3: Examples of spell check contain four letter input string

Data set input	Total combinations	Matching candidates	Time delay in seconds	Accuracy
Gare	1345	26	1.35	51.73
Comn	1237	15	.97	82.42
Fint	876	13	1.27	67.38
Worg	1149	13	.85	88.38

Average accuracy =76.35%
Average time delay =1.11 second

From these tables as letters in input string increases combinations of characters are also increases. Thus

accuracy also increases. But in the case of time delay as number of letters in the input string increases delay also increases due to large number of combinations. That means as efficiency slightly decreases as number of letters in the input string increases.
Based on these examples, average accuracy and efficiency of the system can be calculated.
Average accuracy = 66.197%
Average time delay = 0.763second

5.2 Query reformulation

Query reformulation is the process of rewriting the original query with its similar queries and enhancing the accuracy of search. It involves evaluating a user's input and expanding the search query to match additional documents. When given a query the system generate all similar queries from the original query.

In query reformulation enter a query. The system shows matching queries. From list of matching queries select the top query which user wants. When select that query the system shows the details in a tabular format. Here accuracy can be calculated by the information which is retrieved. For that given some example queries and then takes the matching query. When the query is selected information is obtained in a table form. These queries are related to books and authors. If the result obtained is correct then it is 99% accurate.

Table -4: Examples of query reformulation

Query	Match query	Result	Accuracy (%)
Apjabdulkalam	Famous books by Dr.APJ Abdul Kalam	Books and its details are shown in a table format	99
Muhammadbasheer	Famous books by Vaikom Muhammad Basheer	Books and its details are shown in a table format	98
m t vasudevan	Famous books by M T Vasudavan Nair	Books and its details are shown in a table format	99

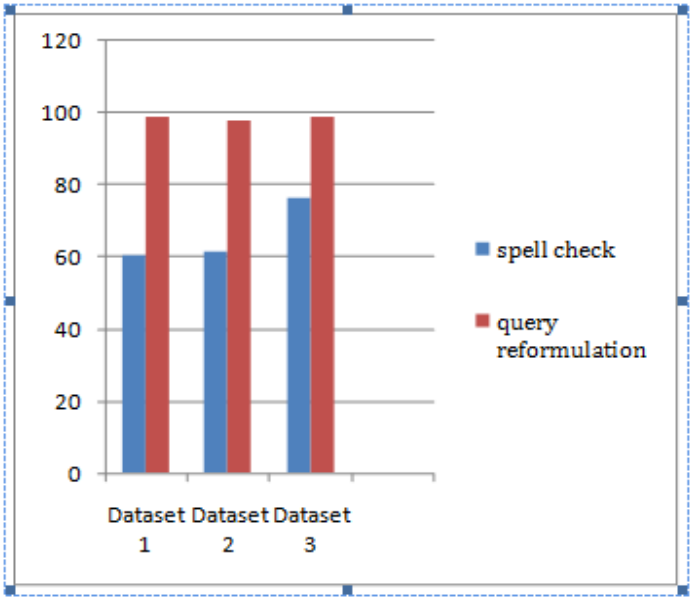

Chart -1: Accuracy comparison between spell check and query reformulation

3. CONCLUSIONS

String transformation, which is an essential problem, in many applications. In natural language processing, pronunciation generation, spelling error correction, word transliteration, and word stemming can all be formalized as string transformation. String transformation can also be used in query reformulation and query suggestion in search. The proposed systems that we focus on generate new queries by substituting or adding words or phrases to the original query. Two queries that are different from each other by only one phrase are selected and the corresponding pairs of phrases are recorded as substitution candidates, which are used to generate substitutions for new queries. Query reformulation is the process of rewriting the original query with its similar queries and enhancing the accuracy of search. It involves evaluating a user's input and expanding the search query to match additional documents.

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to Asst.Professor Mrs. Kavitha karun who guided me throughout my thesis work, without whose knowledge and assistance, I would not have completed my thesis work.

REFERENCES

[1] Ziqi Wang , Gu Xu , Hang Li , and Ming Zhang,"A Probabilistic Approach to String Transformation",IEEE Transaction on knowledge and data engineering,2013,pp.1-14.

[2] M. Li, Y. Zhang, M. Zhu, and M. Zhou, "Exploring distributional similarity based models for query spelling correction," in *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, ser. ACL '06. Morristown, NJ, USA: Association for Computational Linguistics, 2006, pp. 1025–1032.

[3] A. R. Golding and D. Roth, "A winnow-based approach to context-sensitive spelling correction," *Mach. Learn.*, vol. 34, pp. 107–130, February 1999.

[4] J. Guo, G. Xu, H. Li, and X. Cheng, "A unified and discriminative model for query refinement," in *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, ser. SIGIR '08. New York, NY, USA: ACM, 2008, pp. 379–386.

[5] A. Behm, S. Ji, C. Li, and J. Lu, "Space-constrained gram-based indexing for efficient approximate string search," in *Proceedings of the 2009 IEEE International Conference on Data Engineering*, ser. ICDE '09. Washington, DC, USA: IEEE Computer Society, 2009, pp. 604–615.

[6] E. Brill and R. C. Moore, "An improved error model for noisy channel spelling correction," in *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, ser. ACL '00. Morristown, NJ, USA: Association for Computational Linguistics, 2000, pp. 286–293.

[7] N. Okazaki, Y. Tsuruoka, S. Ananiadou, and J. Tsujii, "A discriminative candidate generator for string transformations," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, ser. EMNLP '08. Morristown, NJ, USA: Association for Computational Linguistics, 2008, pp. 447–456.

[8] M. Dreyer, J. R. Smith, and J. Eisner, "Latent-variable modeling of string transductions with finite-state methods," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, ser. EMNLP '08. Stroudsburg, PA, USA: Association for Computational Linguistics, 2008, pp. 1080–1089.

[9] A. Arasu, S. Chaudhuri, and R. Kaushik, "Learning string transformations from examples," *Proc. VLDB Endow.*, vol. 2, pp. 514– 525, August 2009.

[10] S. Tejada, C. A. Knoblock, and S. Minton, "Learning domainindependent string transformation weights for high accuracy object identification," in *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, ser. KDD '02. New York, NY, USA: ACM, 2002, pp. 350–359.

[11] M. Hadjieleftheriou and C. Li, "Efficient approximate search on string collections," *Proc. VLDB Endow.*, vol. 2, pp. 1660–1661, August 2009.

[12] C. Li, B. Wang, and X. Yang, "Vgram: improving performance of approximate queries on string collections using variable-length grams," in *Proceedings of the 33rd international conference on Very large data bases*, ser. VLDB '07. VLDB Endowment, 2007, pp. 303–314.

[13] X. Yang, B. Wang, and C. Li, "Cost-based variable-length-gram selection for string collections to support approximate queries efficiently," in *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, ser. SIGMOD '08. New York, NY, USA: ACM, 2008, pp. 353–364.

[14] C. Li, J. Lu, and Y. Lu, "Efficient merging and filtering algorithms for approximate string searches," in *Proceedings of the 2008 IEEE 24th International Conference on Data Engineering*, ser. ICDE '08. Washington, DC, USA: IEEE Computer Society, 2008, pp. 257–266.

[15] S. Ji, G. Li, C. Li, and J. Feng, "Efficient interactive fuzzy keyword search," in *Proceedings of the 18th international conference on World wide web*, ser. WWW '09. New York, NY, USA: ACM, 2009, pp. 371–380.

[16] R. Vernica and C. Li, "Efficient top-k algorithms for fuzzy search in string collections," in *Proceedings of the First International Workshop on Keyword Search on Structured Data*, ser. KEYS '09. New York, NY, USA: ACM, 2009, pp. 9–14.

[17] Z. Yang, J. Yu, and M. Kitsuregawa, "Fast algorithms for top-k approximate string matching," in *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence*, ser. AAAI '10, 2010, pp. 1467–1473.

[18] C. Whitelaw, B. Hutchinson, G. Y. Chung, and G. Ellis, "Using the web for language independent spellchecking and autocorrection," in *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, ser. EMNLP '09. Morristown, NJ, USA: Association for Computational Linguistics, 2009, pp. 890–899.

[19] E. S. Ristad and P. N. Yianilos, "Learning string-edit distance," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 20, pp. 522–532, May 1998.

[20] J. Oncina and M. Sebban, "Learning unbiased stochastic edit distance in the form of a memoryless finite-state transducer," in *In Workshop on Grammatical Inference Applications: Successes and Future Challenges*, 2005.

[21] A. McCallum, K. Bellare, and F. Pereira, "A conditional random field for discriminatively-trained finite-state string edit distance," in *Proceedings of the 21st Conference on Uncertainty in Artifical Intelligence*, ser. UAI '05, 2005, pp. 388–395.

[22] F. Ahmad and G. Kondrak, "Learning a spelling error model from search query logs," in *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, ser. HLT '05. Morristown, NJ, USA: Association for Computational Linguistics, 2005, pp. 955–962.

[23] K. Toutanova and R. C. Moore, "Pronunciation modeling for improved spelling correction," in *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ser. ACL '02. Morristown, NJ, USA: Association for Computational Linguistics, 2002, pp. 144–151.

[24] H. Duan and B.-J. P. Hsu, "Online spelling correction for query completion," in *Proceedings of the 20th international conference on World wide web*, ser. WWW '11. New York, NY, USA: ACM, 2011, pp. 117–126.

[25] Q. Chen, M. Li, and M. Zhou, "Improving query spelling correction using web search results," in Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, ser. EMNLP '07, 2007, pp. 181–189.

[26] A. Islam and D. Inkpen, "Real-word spelling correction using google web it 3-grams," in *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, ser. EMNLP '09. Morristown, NJ, USA: Association for Computational Linguistics, 2009, pp. 1241–1249.

[27] R. Jones, B. Rey, O. Madani, and W. Greiner, "Generating query substitutions," in *Proceedings of the 15th international conference on World Wide Web*, ser. WWW '06. New York, NY, USA: ACM, 2006, pp. 387–396.