The Adaptive Spelling Error Checking Algorithm based on Trie Tree

Yongbing Xu[a], Junyi Wang[b]

The computer college of Inner Mongolia University,Hohhot China 010021

[a] fengjiexyb@163.com, [b] Corresponding author wjyi@imu.edu.cnl

Keywords: trie tree;edit distance;dynamic programming;spelling error correction

Abstract. In many applications, the function of spelling error checking and auto-complete according to the user's input is very popular. The spelling error checking algorithm based on the dictionary is very fast to find the word written correctly, but need to recalculate the edit distance of a large number of related words if there is a input error.In this text, the adaptive spelling error checking algorithm based on Trie tree processes both the correct input and the incorrect input synchronously, reduce the amount of calculation of edit distance by the Trie tree and stack, and reduce the time complexity. At the same time, the statistical analysis is carried out on the user’s inputs, and improve the accuracy of the feedback results by a large amount of data. Practice results show that the time complexity of the new algorithm is decreased obviously compared with the other similar algorithms.

Introduction

Spelling error checking is widely used in the search engine,DNA analysis and plagiarism detection,etc.The edit distance ,which is put forward by Damerau[1] and Levenshtein[2] ,is a widely used general spelling error correction model. The most common algorithm used for this model is dynamic programming and automata [3]. Dynamic programming has a variety of improved algorithm[4-5]. Due to calculate an arbitrary value only depends on its three values in the upper left corner, then you can adopt the diagonal way of evaluation[6]. Bit parallel algorithm[7] is suitable for the small pattern string (generally is less than machine word length).

When checking for edit distance to a large number of strings, we can use the B-K tree[8]. The essence of B-K tree is a finite automaton. We will do all the words in turn into a n fork tree. First of all, we just find a word as the root. Then calculate the edit distance from the word to the root first When each time you insert a word. If the distance is appearing for the first time in the node, then establish a new son node, Or continue to be recursive along the corresponding edge.

However, the B-K tree is not convenient to search when no mistakes in spelling. So use the Trie-Node algorithm[9-11]. Trie-Node algorithm adopts the Trie tree to store the string collection, the path from the root node to a leaf node only corresponds to a string. But the repeated calculation of the algorithm is too big, and there is no consideration in how to choose the most suitable one in more than one match.

This paper use the Trie tree to achieve both the correct input and the incorrect input synchronous processing. The use of the stack in the Trie tree reduce the repeated calculation problem of edit distance.Record for every correct lookup,and improve the quality of error correction advice by the frequency of searching the Trie tree.

Relevant concepts

Trie Tree[12],also called the Dictionary Tree, the Prefix Tree, the Word Search Tree or Key Tree, is one of n fork tree. The basic nature of Trie tree are:

a. The root node is not stored characters,each child node is stored one character .

b. The characters on the path from the root node to a node, are connected to form a string which is corresponding to the node.

c. All child nodes of each node contains the character is not the same each other.

Error checking can return the similar results when appearing the user’s input errors. Judge similarity generally has the following several ways:

a. Hamming distance, requires two strings str1 and str2 must be the same length, is to describe the number of different characters on corresponding position between the two equal length of string. Put an exclusive or operation on two strings,and statistics the number of 1 in the result,then this number is hamming distance.

b. Edit distance ,also known as Levenshtein distance, add, remove and replace three ways used to turn a string into another. People have studied a single spelling mistake accounted for 93%-95%[13] of all errors.

c. Damerau-Levenshtein distance, is adding a transform on the basis of the edit distance,that is exchanging the positions of the two characters.

Edit distance has three properties:

a． d(x,y)=0 if and only if x=y(the Levenshtein distance is zero if and only if the two strings are equal)

b． d(x,y)=d(y,x)(the minimum number of steps from x to y is equal to the minimum number of steps from y to x)

c． d(x,y)+d(y,z)>=d(x,z)(the number of steps required from x and z is no more than the sum of the number of steps from x into y first then into z), like a triangle, the sum of both sides must greater than the third side.

The improved algorithm based on Trie tree

As you can see from the previous analysis, when the user inputs a word, the system will first check whether the word is correct. If the input is correct,return the result directly.If not,will find the word whose edit distance is smaller across a large number of words(normally not larger than 2). That is to say, when appearing the input error, there are a lot of calculation.

First of all,a 26 fork Trie tree is constructed, and its branches are corresponding to the 26 letters one to one respectively. Each the ith layer of the tree corresponding to the ith character of the input string(the root node of the Trie tree is on 0th layer). If the input word is right, as long as get down along the Trie tree branches,the Trie tree path is the entire string when have finished traversal.

For the list of a lot of word, generally use the method of hash table to lookup. Calculating the hash value generally need to shift and multiplication operation on the each letter in the word. So when the input words are correct, the time complexity of the Trie tree and hash table is consistent.

If appearing user input errors, we can't find the words in the Trie tree. We only need to traverse the tree for all the correct words whose length vary within 2 compared the wrong words,and check their edit distance in turn, then it will filter out a lot of irrelevant words.

Assumes that the first letter didn't write wrong[14](this is less likely),it can also only check the branch which is determined by the first letter, this reduces the inspection of the other 25 major branches.

If the length of input string is len, find all the word whose edit distance is less than n, we only check len±n layers of a branch of the Trie tree.

The process of search Trie tree is equivalent to traversing the Trie tree locally. The traversal result is concentrated, and the difference between the adjacent two results is the smallest. The latter results is always operating at the end of the previous results(to add or remove a few letters). So we can use the stack to save operation steps, and don't have to start to calculate the next word.

In most cases, the size of the returned result set will be greater than 1. So which one is most likely would be the result of the user? It have to modify the traditional Trie tree. In addition to the letter, each node save a frequency value, which is used to represent the match frequency between the letter and its previous letter. In each time you find the right, the frequency value of each node, on the branch in the trie tree, plus one. At this time when a user input error, the returned result sets, would be ordered according to the wrong match frequency.

At the same time, when user's input is not complete, it can be calculated according to the last letter collocation frequency,and several of the maximum frequency will be recommended to the user.

Contrasting and analyzing the experimental results

The data set of this paper is divided into two parts, as shown in table 1.The correctly spelled part uses more than 30 books of the Project Gutenberg[15], total 1915046 words, and 109572 words of them don't repeat.The wrong spelled part is generated according to Kemighan's easy wrong frequency table[16]. At the same time it avoids the artificial markers of work.

Table 1 the statistical results of data set

correct spelling	wrong spelling	total
109572（89.6%）	12704（10.4%）	122276

Evaluation is divided into two parts: correct input and wrong input. When inputting correctly , we compare the Trie tree algorithm, the Hash algorithm and B-K tree algorithm. When inputting wrong,we compare the Trie tree algorithm, the DP algorithm and Trie Node algorithm. At the same time using two indicators to evaluate the algorithm performance of this text:

R@N[17]: For each query, the algorithm give the first N best advices. If the firs N best advice includes the right error correction advice, we think that the system has given the right suggestions to the specified query.

Search time: the time the algorithm run from the beginning of the input to calculate of the result set.

Figure 1 shows the performance of the three algorithms in the correct input cases. Through the tests of different amount of data, we can see that the algorithm is superior to other algorithms in addition to HASH algorithm.

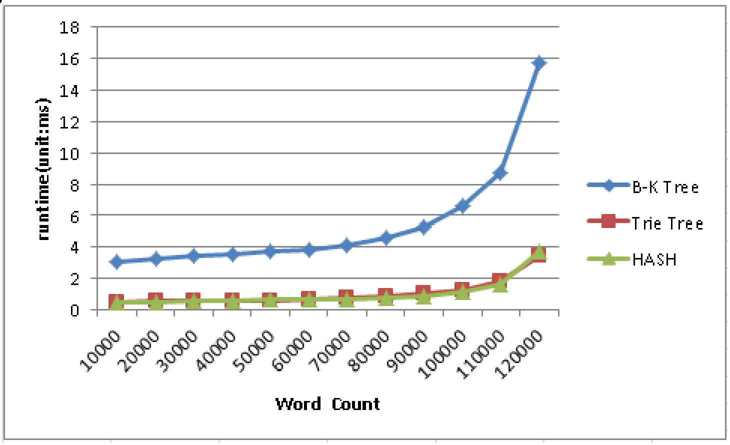

Figure 1 comparison among the three algorithms in the correct input cases

In the case of wrong input, the algorithm shows great advantages. As shown in figure 2, the correcting accuracy is up to 73.65% in the case of result set size 1,and the average time is only 848 μ s.

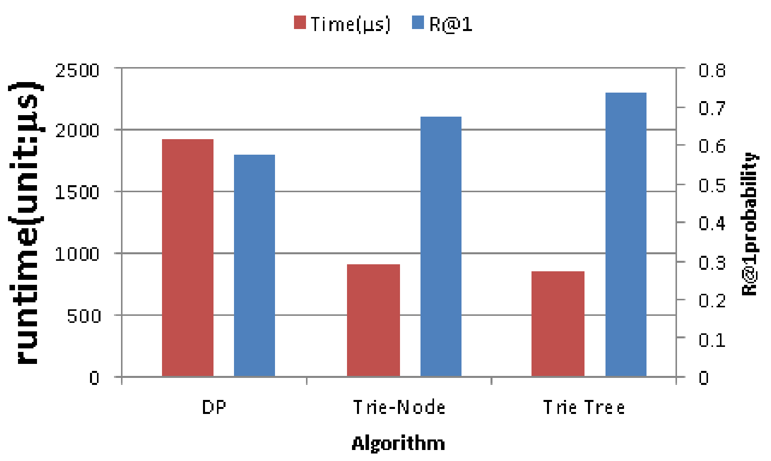

Figure 2 comparison among the three algorithms in the incorrect input cases

Conclusion

This text has studied the string spelling error checking algorithm by limitting spelling error conditions, has put forward an improved algorithm based on Trie tree,which both improves the validity of the corrected result and reduces the calculation time consumption of edit distance at the same time. The superiority of the algorithm is verified by the experiments. However, this algorithm does not take into account space consumption.

References

[1] Damerau F J. Communications of the ACM, 1964, 7(3): 171-176.

[2] Levenshtein V I. Binary codes capable of correcting deletions, insertions and reversals//Soviet physics doklady. 1966, 10: 707.

[3] Ukkonen E. Journal of algorithms, 1985, 6(1): 132-137.

[4] Chang W I, Lampe J. Theoretical and empirical comparisons of approximate string matching algorithms//Annual Symposium on Combinatorial Pattern Matching. Springer Berlin Heidelberg, 1992: 175-184.

[5] Baeza-Yates R, Navarro G. Algorithmica, 1999, 23(2): 127-158.

[6] Zhong C, Chen GL. Journal of Software, 2004,15(2):159~169.

[7] Hyyrö H. Explaining and extending the bit-parallel approximate string matching algorithm of Myers. Technical Report A-2001-10, Dept. of Computer and Information Sciences, University of Tampere, Tampere, Finland, 2001.

[8] Burkhard W A, Keller R M. Communications of the ACM, 1973, 16(4): 230-236.

[9] Feng J, Wang J, Li G. The VLDB Journal—The International Journal on Very Large Data Bases, 2012, 21(4): 437-461.

[10] Arasu A, Ganti V, Kaushik R. Efficient exact set-similarity joins//Proceedings of the 32nd international conference on Very large data bases. VLDB Endowment, 2006: 918-929.

[11] Bayardo R J, Ma Y, Srikant R. Scaling up all pairs similarity search//Proceedings of the 16th international conference on World Wide Web. ACM, 2007: 131-140.

[12] Fredkin E. Communications of the ACM, 1960, 3(9): 490-499.

[13] Peterson J L. Communications of the ACM, 1986, 29(7): 633-637.

[14] Peterson J L. Communications of the ACM, 1980, 23(12): 676-687.

[15] Hart M. Project gutenberg. Project Gutenberg, 1971.

[16] Kernighan M D, Church K W, Gale W A. A spelling correction program based on a noisy channel model//Proceedings of the 13th conference on Computational linguistics-Volume 2. Association for Computational Linguistics, 1990: 205-210.

[17] WANG Xiuzhen, CONG Rui, WANG Fei. Computer Engineering and Applications, 2015, 51（14）： 113-119.