IOP Conf. Series: Journal of Physics: Conf. Series **1302** (2019) 022034 doi:10.1088/1742-6596/1302/2/022034

Research on a Novel Improved KMP Fuzzy Query Algorithm

Bowen Ni, Xiaomei Hu, Jianfei Chai*, Hewei Qu and Tao Yu

Shanghai Key Laboratory of Intelligent Manufacturing and Robotics, School of Mechatronic Engineering and Automation, Information Technology office, Shanghai University, Shanghai, China
Email: chaijianfei12@163.com

Abstract. The rapid development of computer technology has led to an increasing demand for database management information systems. Most of the data queries existing in the current system use accurate query methods, which leads to inefficient query and cannot solve the fuzzy matching problem between strings. Based on the Knuth-Morris-Pratt algorithm, this paper introduces the concept of ambiguity and proposes an improved KMP fuzzy query algorithm, which is applied to the disease query system to verify the feasibility of the algorithm. The improved KMP fuzzy query algorithm not only has a high matching speed between strings, but also satisfies the fuzzy matching between strings. Compared with the traditional BF algorithm, KMP algorithm and other algorithms, the improved KMP fuzzy query algorithm has superiority in terms of fuzzy matching.

1. Introduction

With the rapid development of computer technology, a large number of database-based application software for various industries, including government agencies, banks or enterprises, has emerged. They all have the need for a dedicated database management information system [1-3]. In addition to the basic information storage, operation management, maintenance and transmission functions that users pay attention to, such systems are more concerned with data query [4-6]. For data query, we can easily find a problem: In general, because of the lack of relevant professional knowledge, the string we input does not necessarily exist in the database, so we may not be able to query the knowledge we need in the process of querying. In the early query technology, the implementation of the query mostly used the method of precise query, and the input string must be completely matched with the related query record in the database. For example, for a disease query, the medical term for a symptom is called "systemic weakness." When the database is established, “systemic weakness” is stored as a symptom in the disease data. When using the system for self-examination of the disease, due to the limitations of the precise query, when the user inputs “body weakness”, the search cannot be completed. How to successfully complete the search in the case of the above problems is the content of this paper.

2. Improved KMP Fuzzy Query Algorithm

The fuzzy query algorithm can quickly and accurately match the information stored in the database under the condition of limited query information, so that the query results are output in order according to the accuracy. As a string matching algorithm, KMP has been widely used in various fields, but the KMP algorithm is aimed at string exact matching, which cannot meet the actual needs in some applications. Therefore, it is necessary to implement incomplete matching of strings on the basis of improving matching efficiency. Because the query for the disease is actually to achieve the string matching between the query character and the database character, and also to implement the fuzzy

query, this is the incomplete matching of the string. In addition, while the KMP algorithm speeds up character matching, there are also defects in the repeated comparison of text strings and pattern strings. Therefore, based on the KMP algorithm, an improved KMP algorithm is proposed, which eliminates the defects of multiple string repeated comparisons and implements string fuzzy query.

2.1. Defining Ambiguity

Using the improved KMP algorithm to achieve intelligent query of diseases, the following concepts are introduced firstly.

2.1.1. Ambiguity. Ambiguity is used to indicate the degree of matching of strings, and different ambiguities indicate different matching results. In this paper, the ambiguity is set to three levels to indicate three different levels of matching, and the user selects different ambiguity levels according to the actual situation. The details are as follows: (assuming the length of the string two is greater than the string one)

- The ambiguity level 1 indicates a completely inclusive relationship, that is, the content of the string two completely contains the string one, and the order of the included strings is completely identical.
- The ambiguity level 2 indicates a partial inclusion relationship, that is, the string two contains a part of the string one. In other words, the two strings have a certain degree of similarity, and the order of the included strings is also completely identical.
- The ambiguity level 3 indicates that there is a similar relationship, that is, there are only a few identical characters between two strings. It has a certain degree of similarity and does not require these same character sequences to be consistent.

Through the above division of the ambiguity level, it can be clearly seen that as the ambiguity level is continuously reduced, the query requirements become stricter, and the fuzzy query becomes more accurate in the fuzzy requirement. When performing fuzzy query, the ambiguity can be adjusted appropriately, and the disease fuzzy query can be completed according to the demand.

2.1.2. Fuzzy similarity. Fuzzy similarity refers to the ratio of the number of identical characters contained in two strings to the number of characters in the shortest string, and the ratio is used to represent the fuzzy similarity.

2.2. Research on Improved KMP Fuzzy Query Algorithm

2.2.1. Definition of data structure. The data structure of the improved KMP fuzzy query algorithm is defined as follows: Define a record to store the result and similarity of the string matching, the record name is $KMPPRO$. The record includes two variables: a boolean variable $find$ and an integer variable $number$, where $find$ is used to determine whether the KMP algorithm is used to complete the string matching, and $number$ is used to store the similarity generated during the KMP algorithm. The specific definition is shown in Figure 1.

```
KMPPRO = record;
number : integer;
find : boolean;
end;
```

Figure 1. Definition of data structure.

2.2.2. Improved KMP algorithm. Define a function to represent the improved KMP algorithm. The function header contains three parameters S, T, M. S and T represent the type of the string, and M is the type of $KMPPRO$, which is used to determine whether the two strings match exactly and the similarity of the matching.

It is assumed that the pointers of the characters to be compared of the main string S and the pattern string T are i and j respectively, the initial values of both are 1. In the process of matching, if $s[i] = t[j]$, the values of i and j are increased by 1 respectively. Otherwise, the values of $m.number$ and j are compared. If $m.number > j$, then $m.number$ is assigned to j, otherwise the value of $m.number$ remains unchanged, i retreats to $i - j + 2$, and j is reassigned to 1, and so on, until the following two situations occur:

- The value of i is greater than the length of the main string S.
- The value of j exceeds the value of the pattern string T.

In this way, the value of j can be used to judge whether the two strings match, and if the two strings do not match, the maximum similarity of the matches is recorded by $m.number$. The improved KMP algorithm function is shown in Figure 2.

```
procedure indexf(s, t:string; var m: KMPPRO);
begin
 m.number =1; i =1; j =1;
 length 1= length(s);
 length 2= length(t);
 while (i <=length 1 && j <=length 2 ) do
    if (s[i] = t[j])
            i =i + 1; j =j + 1;
    else
            if (m.number < j)
               m.number = j − 1;
               i = i − j + 2;j = 1;
 if(j >length2)
  m.find =ture;
 else
  m.find =false;
end;
```

Figure 2. Improved KMP algorithm function.

2.2.3. Ambiguity judgment. The ambiguity judgment function is expressed as $mh()$, and the function implementation principle is as follows: the length of the character string S is greater than the length of the character string T. The length of T is m, and there are n characters of the same, then the relationship between m and n can be used to judge whether the meanings of the two strings are similar.

According to relevant research and experience, if m is 4 times or even more than 4 times of n, then the meaning of the two strings is different. If it is less than 4 times, the relationship between m and n is judged according to the value of m, which is divided into the following two cases:

- When $m < 5$, if $m - n < n$ holds, the meaning of the two string expressions is considered to be the same. Otherwise, the meanings of the two expressions are different.
- When $m > 5$, if $\frac{m-n}{n} < (\frac{2+\frac{m}{10}}{m})$ is established, the meaning of the two string expressions is considered to be the same. Otherwise, the meaning of the two expressions is different.

The code implementation is shown in Figure 3.

```
function mh(m, n: word):boolean; begin
if(m / n > 4)then begin mh:= false; exit end
else begin
if(n < 5)then begin if(m − n) < n then mh:= true else mh:= false
end else
if(m − n)/ n <((2+ m /10)/ m)then mh:= true else mh:= false
end;
end;
```

Figure 3. Code implementation.

2.2.4. Methods of fuzzy query and how to implement fuzzy query level. Write the indexf() function to implement the matching of characters and record the effect of matching similarity. The $mh()$ function discriminates whether two strings can be considered as fuzzy matching. Through these contents, you can design a procedure to call these two functions and realize the matching judgment of fuzzy query. The variable design of the process function is shown in Figure 4.

```
function mhcheck (S, T: string, mhtype: boolean; mhdeep: 1,2,3):boolean;
```

Figure 4. Variable design of process function.

Among them, the boolean $mhtype$ is used to distinguish the content of the string; $mhdeep$ is the fuzzy precision.

The principle of the function is: Firstly, the two strings are matched and calculated by calling the indexf() function to find whether the match and the similarity of the match, if it matches, it returns True, otherwise the ambiguity is further determined. If the ambiguity is 1, it returns False. If the ambiguity is 2, the $mh()$ function is called to judge the ambiguity ($m.number$) and the length of the short character. If the return value of $mh()$ is true, it returns True, otherwise it returns False. If the ambiguity is 3, the $mhtype$ value is judged. If the $mhtype$ is true, the judged string is a Chinese character, otherwise it is English, and then the short string and the long string character unit are cyclically compared to see whether there are equal bytes, if there is an equal byte, the third-order similarity identifier $mn3$ is increased by 1. After the above process, the value of $mn3$ is calculated, and the $mh()$ function is used to judge the return result of $mn3$ and the length of the short string, and it returns to the mhcheck process. The mhcheck() function procedure code is shown in Figure 5.

```
function mhcheck(S, T:string, mhtype:boolean, mhdeep:1,2,3):boolean;
var m, n, mn1:word; i, j:word; mn: KMPPRO;
begin
m:= length(S); n:= length(T);
mn3:= 0;
if m < n then indexf(T, S, mn)
else indexf(S, T, mn);
if mn.find= true then result:= true else
if mhdeep = 2 then begin
if mn.number = 0 then result:= false
else begin
if(mh(mn.number, m))or(mh(mn.number, n))then result:= true else
if mhdeep>= 3 then
if mhtype then begin
for i:= 1 to div(m,2) do
for j:= 1 to div(n,2) do
if((S[2* i − 1]= T[2* j − 1])and(S[2* i]= T[2* j]))
then mn3 := mn3 + 1;
if(mh(mn3, n))or(mh(mn3, m))then result:= true else result:= false
end else begin
for i:= 1 to m do
for j:= 1 to n do
if S[i]= T[j] then mn3 := mn3 + 1;
if(mh(mn3, n))or(mh(mn3, m))then result:= true
else result:= false; end
else result:= false; end end
else result:= false;
end;
```

Figure 5. Mhcheck() function procedure code.

3. Case Analysis and Algorithm Comparison

In the field of disease query of traditional Chinese medicine, because the amount of traditional Chinese medicine data is extremely large and there is ambiguity and uncertainty, the traditional query algorithm can complete the query only when the input disease characters and the characters existing in the database are identical, and the efficiency is low. The improved KMP fuzzy query algorithm introduces the concept of "ambiguity", which not only realizes the fuzzy matching between strings, but also improves the matching speed, and applies it to the disease query system of Chinese medicine, which improves the query efficiency and accuracy. The improved KMP fuzzy query algorithm is further illustrated by specific example analysis and algorithm comparison.

3.1. Case Analysis

Through the improvement of the data structure of the KMP algorithm and the addition of related functions, the incomplete matching between the two strings is realized on the basis of ensuring the fast matching of the strings. In order to better illustrate the different ambiguity of the improved KMP algorithm to generate different query results, three different pattern strings are listed to match the main string respectively. The analysis process and the results are as follows:

- Main string I: low voice, short and weak breathing.
- Patterns string II: short and weak breathing.
- Patterns string III: short and weak breathing, mental fatigue and limb fatigue.
- Mode string IV: small voice, short breathing.

The main string I stores a symptom of "deficiency of vital energy ". When the user queries it, there may be symptom input of mode string II, III and IV. For different ambiguities, inputting the string in three modes produces the following different results:

- When the ambiguity level is 1, only mode string II can match the main string I, and III and IV do not match.
- When the ambiguity level is 2, the pattern string II and the pattern string III can match the main string I, but the pattern string IV cannot be matched with the main string I.
- When the ambiguity level is 3, the pattern string II, the pattern string III and the pattern string IV can all match the main string I.

According to the above analysis, it is easy to find that as the degree of ambiguity increases, the range of characters available for the user to input is getting larger and larger, thereby achieving the query requirement. However, the increase of the ambiguity increases the search range of the database, which leads to the cumbersome search results. Therefore, when making the query, the appropriate ambiguity must be selected according to the actual situation to achieve the desired expectation.

3.2. Comparison of Algorithms

The traditional Chinese medicine disease database contains 489 kinds of common diseases. According to the books such as traditional Chinese medicine science, the data are authentic and accurate. The accuracy and matching efficiency of the string matching algorithm BF algorithm [7], KMP algorithm [8] and the improved KMP fuzzy matching algorithm proposed in this paper are compared as shown in Table 1 below.

Table 1. Algorithm comparison result.

	Accuracy	*Matching efficiency*
BF algorithm	67%	0.5
KMP algorithm	67%	0.7
Improved KMP fuzzy matching algorithm	88%	0.9

Through the comparison of accuracy and matching efficiency, it can be clearly seen that the improved KMP fuzzy matching algorithm has a significant improvement over the BF algorithm and the KMP algorithm.

4. Conclusion

The traditional string matching algorithm cannot solve the fuzzy matching problem between strings. Based on the KMP algorithm, this paper integrates the concept of "ambiguity" and proposes an improved KMP fuzzy query algorithm. The change of data structure, fuzzy matching judgment and fuzzy search method are implemented. The feasibility of the algorithm is proved by an example. The improved KMP fuzzy query algorithm not only has a high matching speed between strings, but also satisfies the fuzzy matching between strings. Compared with the traditional BF algorithm, KMP algorithm and other algorithms, it has superiority in terms of fuzzy matching.

5. References

[1] Li Li, Jiang Yuxi, Lin Wei, Jiang Binghua. An improved algorithm based on KMP algorithm KMPP[J]. Computer Engineering and Applications, 2016, 52(08): 33-37.
[2] Cai Ting, Yang Weishuai. An Improved String Pattern Matching Algorithm[J]. Internet of Things Technology, 2017, 7(07): 89-91+95.

[3] Al-Ssulami, Abdulrakeeb M. Hybrid string matching algorithm with a pivot [J]. Journal ofInformation Science, 2015, 41 (1): 82-88.
[4] Rao, Chinta Someswara, Viswanadha Raju, S. A novel multi pattern string matching algorithmwith while shift [C]. ICTCS 2016, 2016, 04.
[5] Hlayel, Abdallah A , Hnaif, Adnan. An algorithm to improve the performance of stringmatching [J]. Journal of Information Science, 2014, 40(3):357-362.
[6] Kalita, Nitashi, Chitra, Sharma, Radhika, Borah, Samarjeet. EKMP: A proposed enhancement of KMP algorithm [J]. Smart Innovation, Systems and Technologies, 2015, 33: 479-487.
[7] Yan Fawen, Huang Min, Wang Zhongfei. Detection of Network Abnormal Traffic Behavior Based on BF Algorithm[J]. Computer Engineering, 2013, 39(07): 165-168+172
[8] Yang Zhanhai. Research and Analysis of KMP Pattern Matching Algorithm[J].Computer and Digital Engineering,2010,38(05):38-41.