String Matching Techniques for Searching: Algorithms and Applications Term Paper - Spring 2000

KWOK Chi Leong
clkwok@cse.cuhk.edu.hk

Supervised by
Prof. Michael Lyu

Department of Computer Science and Engineering
The Chinese University of Hong Kong
Shatin, N.T., Hong Kong

April 19, 2000

Abstract

Searching of Information is one of the important aspect in the computer science and varies environment, including the fast growing Internet, requires application with effective searching technique. We'll look into several aspect on searching and introduce some string matching techniques and algorithms which will be useful in varies needs of searching.

1 Introduction

Searching is a common and popular problem in the computer science area. Control Instruction and Data are two basic components is the modern computer architecture. We usually need to have difference technique to search something from large amount of data. A lot of different kinds of information in the world, they needs an effective way to be retrieved, searching techniques is playing a main role in technology advancement. Internet is one of the typical and new example in 1990s, information in the Internet is growing fast in exponential rate and a effective search engine for the internet becomes one of the hottest topic.

In general, there are some key-points in doing searching:

-Efficiency, searching from large usually requires a lot of time, we usually want reduce the time in obtaining the search result. Usually efficiency plays a key role in searching.

-Accuracy, in difference searching context, the requirement for the searching result is difference. We want to describe what we're going to search as precise as possible, but there is still limitation of the expression ability of the restricted searching method. In this situation, relevant result and irrelevant result may also exist. The accuracy of searching is to describe how precise the searching method can be for a particular form of searching.

-Precision, in any kind of searching, the first we need to do is to describe what we're going to search. On some occasion, it's difficult to describe in precise way, it's particularly on multimedia information such that everyone may have difference perception and to describe in non-uniform way. Some searching may want to allow some flexibility for the results that may be almost meeting the searching critiria but not exactly.

2 Goal of Searching

2.1 Efficiency

In search of useful information, we need to look from a pool of information (database), which can be in presumed order of sequence or unordered at all.

Searching from ordered information, we may not need to seek and inspect every pieces of information. Shortcut of skipping some irrelevant information due to its ordered nature.

However, it's unlikely to have a information pool with order. Sometimes it's not worth to keep in particular order, and sometimes the information itself does not have concept of order at all. In order to search from unordered pool, we usually need to seek for each elements inside at least (and preferably at most) once. This is the unavoidable cost of searching in the raw database.

2.2 Accuracy

Accuracy of a searching is very depends on the expressive power of the searching criteria. In well-fitted situation, search query can be in exact and the result will be the exact matching information. However, there is not easy to achieve. Therefore, the accuracy may not be high in some application and it is always an open problem in the searching domain.

2.3 Precision

This would be a trade off in designing a search engine depends on particular requirement. We may want a efficient engine that can accurately report the information in exact with our description; we may sometimes need a search engine to gather as much related information as possible with certain amount of error/mismatch. On some occasion, we may discard the consideration of efficiency as main factor, and looking for a engine that can analysis our requirement from the non-precise searching criteria, and to return some accuracy results suiting our need.

We can note that the accuracy and precision of searching is usually on two extremes. One will give you exact information, but it may be irrelevant when searching criteria is not well defined; another will give you more possibilities in a gentle way, but it may result in too much irrelevant information mixed in your required results still.

3 String Matching

String Matching is a special type of one-dimensional data recognition, it is a useful technique to search for information that is in one dimension.

In discussion of string matching, we will assume that the string is composed of sequence of characters in which each character is from a finite set of alphabets. Without loss of generality, we further assume that the alphabet set is A-Z with 26 elements and the sequence of characters should be in random order. With particular application is considered, the corresponding alphabet set can be difference and the string may have a special kind of sequence which help useful for further improvement in particular.

Four typical string matching problem will be reviewed and discussed. They are namely 'Exact matching', 'Substring matching', 'Scattered string matching' and 'Corrective length estimation'.

3.1 Exact matching

Exact matching is the typical example and the most obvious example in string matching. With a given string(S) and the target string(T), we are going to match S with T and to determine if they are exactly the same. Formal speaking, S is match with T if and only if $|S| = |T|$ and for each character in S is equal to the corresponding character in T of the same position.

3.1.1 Analysis

As discussed before, in random environment, we always need to inspect the whole sequence as least once in order to determine for a match. This would be obvious for me to know that computation time for a exact string matching would be $O(|S|)$.

However, if there is any mismatch in between, we need not continue the comparison anymore and can determine that S and T are not matched. In this case, the computation time will be much less the worse case situation.

3.1.2 Algorithm

```
$ExactMatching(S,T)$
1.  if $|S| \neq |T|$ then return(mismatch)
2.  for each $i$ from 1 to $|S|$
3.      if $S_i \neq T_i$ then return(mismatch)
4.  return(match)
```

3.2 Substring matching

Usually, searching of information will usually supply only a part of the requirement string as a sample. The part of string is called substring, which is usually called keyword. Using the substring(S), we're to determine if S is substring of target string(T).

In any other words, with given string(S), we are going to determine whether exist a substring of text, say T', which is a part of consecutive sequences in T, such that S and T' are exactly matching.

3.2.1 Analysis

Using the exact string matching comparison method, we can match S with every substring of T with length $|S|$. This method is directly derived from exact string matching. The computation time of this approach is $O(|S||T|)$.

However, substring matching is rather common and it has a lot of method to doing this with better computation time.

3.2.2 Algorithm

In [8], it has discussed quite a lot of algorithms. The Rabin-Karp algorithm [3] is a generalized method come with computation time $O((|T|-|S|+1)|S|)$. This method is to make use of a hash function to compute the digest of string S. Using the hash function that is suitable for fast computation in windows sliding of text T, each comparison of substring digest will be in atomic time $O(1)$. It will generally linear to $|T|$, and for each matched digest, it will perform a exact string matching with time $|S|$ until an exact match is found. Therefore, it is still in $O((|T|+|S|+1)|S|)$ but in good average running time in practical.

In [5], it also has a topic discussing the substring matching problem. A fast algorithm, KMP algorithm [4] is provided and discussed there. It is to make use of a self recurring reference memory to avoid duplication of comparison and get reach to computation time of $O(|T|)$.

3.3 Scattered string matching

The substring matching problem can also have a scattered version. This may be very useful in an environment that error may be injected on target string and so it will have some noisy characters inside.

In this matching, with a string(S), we are going to determine whether S contains in the target string(T). That is, we are going to see whether we can have a way to inject new characters inside S to form new string(S') such that S' and T are exactly matched.

In another words, if S is in sequence of $S_1, S_2, ...S_{|S|}$ and T is in sequence of $T_1, T_2, ...T_{|T|}$, we can going to found a sequence $T_{t_1}, T_{t_2}, ..., T_{t_{|S|}} for t_i < t_{i+1} such that S_i = T_{t_i}$.

3.3.1 Analysis

In scattered environment, we can construct a maximum of $C_{|S|}^{|T|}$ difference string from T with length $|S|$ and then to perform an exact matching with each of them. However, the computation time for this approach will be growing very fast $|S|$ and $|T|$.

For this problem, automata approach will be useful for small $|S|$. We can construct an automata for the sequence of S and to process the sequence of T in the automata to determine for a match.

3.3.2 Algorithm

```
$ScatteredMatching(S, T)$
1.    for each $i$ from 1 to $|S|$
2.        A[i] = FALSE
3.    A[0] = TRUE
4.    for each $j$ from 1 to $|T|$
5.
6.        tmp = A
7.        for each $i$ from 1 to $|S|$
8.            if (($S_i = T_j$) and (tmp[i-1] = TRUE)) then
9.                A[i] = TRUE
10.           else
11.               A[i] = FALSE
12.
13.    return(A[$|S|$])
```

3.4 Corrective length estimation

There is a given string(S) and a target string(T), S and T may look similar but we need to have some measurement on how close are they look alike. To make it more precise, we will estimation the minimum morphing operation of S to T. And each insertion, deletion and replacement of a single character will count as one single operation.

Insertion:
With original String S, if a character c is insert before position i. We will obtain a new String S' with corrective length 1 to String S.
$S' = insert(S, c, i)$ while $|S'| = |S| + 1$

Deletion:

With original String S, if a character is deleted at position i. We will obtain a new String S' with corrective length 1 to String S.
$S' = delete(S, i)$ while $|S'| = |S| - 1$

Replacement:
With original String S, if a character at position i is replace with character c. We will obtain a new String S' with corrective length 1 to String S.
$S' = replace(S, c, i)$ while $|S'| = |S|$

3.4.1 Analysis

In the error corrective problem, we are looking for the minimum number of changes to transform given string(S) to the expected target(T). One of the obvious way is to try all possible in S so as to produce a set of string S_k with all possible string with corrective length k to original string S. Corrective length would be k if there exist a exact matching with S_k and T and no exact matching with S_v and T for all $v < k$.

$$S_0 = S$$

$$S_{k+1} = \{i < |S_k|, c \epsilon \mathcal{Z}, j < |S_k| + 1 \mid insert(S_k, j, c), delete(S_k, i), replace(S_k, i, c)\}$$

```
    Algorithm
1.  oS = $S$
2.  length = 0
3.  while $T \notin$ oS do
4.
4.      nS = $insert(oS, i, c), delete(oS, i), replace(oS, i, c)$
5.      oS = nS
6.      length = length + 1
7.
8.  return(length)
```

This is only a simple idea and it's not difficult to observe that the set oS is growth in a very fast way and thus it is impractical to hold all the generated strings.

In fact, even for a pair of total irrelevant strings, their error length is no more than $|S| + |T|$, that is to first insert all characters in T in order and then remove all characters in original S.

The replacement operation, in fact, is a kind of shortcut that can combine a insert and delete pair in same position.

We may construct a walk map from S to T with insertion operation as horizontal and deletion operation as vertical. Diagonal path for insertion/deletion will be there with length 1 if insert and delete character are difference and length 0 if insert and delete characters are the same.

3.4.2 Algorithms

Details of a efficient approach in calculating corrective length is explained in [10] as well as [1]. The general idea is using dynamic programming approach to constraint a fixed number of possible correction paths, each of the path is heading to the path with direction and we're going to find a minimum path inside.

$PossibleAlgorithm$
1. Construct the walk path map(WPM), which is a graph with $|S| * |T|$ vertices and about $3 * |S| * |T|$ edges.
2. From WPM, we have S as source and T as destination and we can calculate the shortest path from S to T in WPM using *Single Source Shortest Path Algorithm* [8].
3. The correction length of the string S and T is the length of the shortest path.

3.4.3 Application

Spelling checker of text processor can make use of this to compare the word with spelling mistake and suggest some correct words with least error correct length.

In field of molecular biology, the mutated DNA sequences maybe similar but with some difference, this technique may also be used there.

Besides, although it is a one-dimension problem, but multi-dimension environment may also be mapped into a one-dimension environment with similar kind of properties.

4 Searching Application with string matching

String matching problems itself are discussed based on text searching on alphabets.

However, many complicated searching problems in difference areas may be simplified into a string matching problems model. Under process of suitable refinement and mapping, string matching may have significant contribution to them. Details of some research applications example is in [7] for character recognition and in [9] give an idea for text image detection. [2] illustract an example with sound and music and [6] provides more ideas about string matching.

4.1 Music Recognition

Computer music are usually in MIDI format. MIDI music is file with /it .mid as extension storing information of each single music note. For example, the pitch and the duration of a note, and/or the loudness and addition special effect. Each channel of the music will have a dedicated midi instrument will play the role of notes inside that channel.

To recognize a music, we will usually hear its melody part(or harmony part for some people), those are mainly pitch and duration information of music notes. Therefore, music recognition is also a kind of text based searching of a particular piece of music pattern over a music database.

4.2 Internet Search Engine

Information of the Internet is of varies format, but they are basically text file plus some other multimedia files such as video, music, and pictures. The Internet itself is a huge database and each hypertext file will usually consist of a couple of keywords. The author of the homepage provide this keywords in help of searching. Therefore, Internet searching, in this form, is sometimes a text base (one-dimensional) searching. The techniques of one-dimensional searching will usually be able to apply to the Internet searching directly on difference requirement of searching goal and accuracy.

4.3 Extract Features of Image Database

High dimensional data such as images or pictures are usually difficult to search in raw structure. Abstract or features of images are extracted and to be used for searching instead. With a good understanding of the requirement and extracting key feature, searching for image can be done in a linear way with help of string matching techniques. This topics includes the representation and structure of the abstraction and a suitable matching criteria that mostly suitable for the realistic requirement.

5 Conclusions

String matching problem is a fundamental problem in identifying the similarity of two strings, usually word in text. With difference scenario and realistic situation, the matching can have different requirements and the difficulty to achieve the goal will be difference. But they almost being studied and discussed during the history of time. Many common searching and recognition problem are dealing with high dimensional data and it may sometimes be flattened into a string-liked one-dimensional abstract. With a good understanding of the problem, the flattened abstract can almost represent the original data. Those problems will then be able to be solved with effective and efficient string matching techniques.

6 Future Work

1. Continue the study of difference kinds of matching techniques
2. Investigate on some matching problems with better algorithms
3. Applying the techniques on realistic application such as multimedia database or Internet information

References

[1] Mikhail J. Atallah. *Algorithms and Theory of Computation Handbook.* CRC, 1999.

[2] Jia-Lien Hsu Chih-Chin Liu and Arbee L. P. Chen. An approximate string matching algorithm for content-based music data retrieval. 1999.

[3] Richard M. Karp and Michael O. Rabin. Efficient randomized pattern-matching algorithms. *Technical Report TR-31-81, Aiken Computation Laboratory, Harvard University*, 1981.

[4] J. H. Morris Knuth D. E. and V. R. Pratt. Fast pattern matching in strings. *SIAM Journal on Computing*, 6, 1977.

[5] UDI Manber. *Introduction to algorithms.* Assison-Wesley, 1989.

[6] Jin Hwan Park and K. M. George. Parallel string matching algorithms based on dataflow. In *Proceedings of the 32rd Hawaii International Conference on System Sciences*, 1999.

[7] Sargur N. Srihari Sung-Hyuk Cha, Yong-Chul Shin. Approximate stroke sequence string matching algorithm for character recognition and analysis. 1999.

[8] Charles E. Leiserson Thomas H. Cormen and Ronald L. Rivest. *Introduction to algorithms.* McGraw Hill, 1989.

[9] Victor Wu and Edward M. Riseman. Textfinder: An automatic system to detect and recognize text in images. *IEEE Transactions on pattern analysis and machine intelligence*, 21(11), 1999.

[10] Wen-Yen Wu and Mao-Jiun J. Wang. Two-dimensional object recognition through two-stage string matching. *IEEE Transactions on image processing*, 8(7), 1999.