An Improvement and an Extension on the Hybrid Index for Approximate String Matching

Heikki Hyyrö[1,2] ⋆

[1] PRESTO, Japan Science and Technology Agency
[2] Department of Computer Sciences, University of Tampere, Finland.
Heikki.Hyyro@cs.uta.fi

Abstract. In [2] Navarro and Baeza-Yates found their so-called hybrid index to be the best alternative for indexed approximate search in English text. The original hybrid index is based on Levenshtein edit distance. We propose two modifications to the hybrid index. The first is a way to accelerate the search. The second modification is to make the index permit also the error of transposing two adjacent characters ("Damerau distance"). A full discussion is presented in Section 11 of [1].

Let $ed(A, B)$ denote the edit distance between strings A and B, $|A|$ denote the length of A, A_i denote the ith character of A, and $A_{i..j}$ denote the substring of A that begins from its ith and ends at it jth character. Given a length-m pattern string P, a length-n text string T, and an error limit k, the task of approximate string matching is to find such text positions j where $ed(P, T_{h..j}) \leq k$ and $h \leq j$. Levenshtein edit distance $ed_L(A, B)$ is the minimum number of single-character insertions, deletions and substitutions needed in transforming A into B or vice versa. Damerau edit distance $ed_D(A, B)$ is otherwise similar but permits also the operation of transposing two permanently adjacent characters.

Using an index structure during the search can accelerate approximate string matching. One such index is the hybrid index of Navarro & Baeza-Yates [2] for Levenshtein edit distance, which they found to be the best choice for searching English text. It uses *intermediate partitioning*, where the pattern is partitioned into j pieces $P^1, .., P^j$, and then each piece P^i is searched for with $d^i = \lfloor k/j \rfloor$ errors. If $j > 1$ and a hit $T_{j-h..j}$ is found so that $ed_L(P^i, T_{j-h..j}) \leq d^i$, the text area $T_{j-m-k..j+m+k}$ will be included in a check for a complete match of P with k errors. The hits for each piece P^i are found by a depth-first search (DFS) over a suffix tree[1] built for the text. This involves filling a dynamic programming table D, where $D[r, l] = ed(P^i_{1..r}, T_{j+1..j+l})$, during the DFS. When the DFS arrives at a node that corresponds to the text substring $T_{j+1..j+l}$, the distances $ed_L(P^i_{1..r}, T_{j+1..j+l})$ are computed for $r = 1 \ldots m^i$, where $m^i = |P^i|$.

Our main proposal for accelerating the DFS is as follows. When the DFS reaches a depth-l node that corresponds to the text substring $T_{j+1..j+l}$ and where $D[r, l] \geq d^i$ for $r = 1..m^i$, the only strings that have $T_{j+1..j+l}$ as a prefix and match P^i with d^i errors are of form $T_{j+1..j+l} \circ P^i_{h+1..m^i}$, where $\circ$ denotes

⋆ Supported by Tampere Graduate School in Information Science and Engineering.

[1] A trie of all suffixes of the text in which each suffix has its own leaf node and the position of each suffix is recorded into the corresponding leaf.

a)

		t	h	e	r	e			
	0	1	2	3	4	5	6	7	8
t	1	0	1	2	3	4			
h	2	1	0	1	2	3			
e	3	2	1	0	1	**2**			
s	4	3	2	1	1	**2**	2		
i	5	4	3	2	2	**2**	2	2	
s	6	5	4	3	3	3	2	2	2

b)

$m^i = m$	$d^i = k$	OURS/NBY (WSJ)	OURS/NBY (yeast)
5	1	0,23	0,66
5	2	0,33	0,74
10	1	0,19	0,20
10	2	0,33	0,31
10	3	0,41	0,39
15	1	0,19	0,19
15	2	0,34	0,27
15	3	0,43	0,36
15	4	0,50	0,49

Fig. 1. Figure a): Matrix D for computing $ed_L(P^i_{1..r}, T_{j+1..})$, where $P^i_{1..m^i}$ = "thesis", $T_{j+1..}$ = "there..", and $d^i = 2$. Now $D[r, 5] \geq d^i = 2$ for $r = 1 \ldots m^i$, and the only way to reach a cell value $D[m^i, x] \leq 2$, where $x > 5$, is to have only matches at the remaining parts of the top-left-to-bottom-right diagonals with the value $D[h, 5] = 2$. The cells in these diagonal extensions have the value $d^i = 2$ underlined, and the pattern suffixes corresponding to the cell values $D[3, 5]$, $D[4, 5]$ and $D[5, 5]$ (shown in bold) are $P^i_{4..6}$ = "sis", $P^i_{5..6}$ = "is" and P^i_6 = "s", respectively. Figure b): The ratio between the running time of our improved DFS (OURS) and the runtime of the original DFS of Navarro and Baeza-Yates (NBY). We tested with two $\approx$ 10 MB texts: Wall Street Journal articles (WSJ) and the DNA of baker's yeast (yeast). The computer was a 600 Mhz Pentium 3 with 256 MB RAM, Linux OS and GCC 3.2.1 compiler.

concatenation and h fulfills the condition $D[h, l] = d^i$. In this situation we check directly for the presence of any of these concatenated substrings, and then let the DFS backtrack. Fig. 1a illustrates, and Fig. 1b shows experimental results from a comparison against the original DFS of [2] when $P^i = P$ and $d^i = k$.

In addition we propose the following lemma for partitioning P under Damerau distance. It uses *classes of characters*, which refers to permitting a pattern position to match with any character enumerated inside square brackets. For example P = "thes[ei]s" matches with the strings "theses" and "thesis".

Lemma 1. *Let P^i, $i = 1..j$, be j non-overlapping substrings of the pattern P that are ordered so that P^{i+1} occurs on the right side of P^i in P. Also let B be some string for which $ed_D(P, B) \leq k$, let each P^i be associated with the corresponding number of errors d^i, and let strings $\bar{P}^i$, $i = 1..j$, be defined as follows:*

$\bar{P}^i = P^i$, if $i = j$ or P^i and P^{i+1} do not occur consecutively in P.

$\bar{P}^i = P^i_{1..m^i-1} \circ [P^i_{m^i} P^{i+1}_1]$, otherwise.

If $\sum_{i=1}^{j} d^i \geq k - j + 1$, then one of the strings $\bar{P}^i$ matches inside B with at most d^i errors.

References

1. H. Hyyrö. *Practical Methods for Approximate String Matching.* PhD thesis, Department of Computer Sciences, University of Tampere, Finland, December 2003.
2. G. Navarro and R. Baeza-Yates. A hybrid indexing method for approximate string matching. *Journal of Discrete Algorithms (JDA)*, 1(1):205–239, 2000.