Theory: Edit distance alignment

16 minutes 0 / 5 problems solved

§1. Introduction

Recall that for two strings s and t, the edit distance $d_E(s, t)$ is the minimum number of insertions, deletions, or substitutions required to transform s into t. For now, we learned the algorithm that allows finding only the number of such operations. But sometimes, we want to know the exact sequence of transformations corresponding to the edit distance.

For example, suppose we want to compare a DNA sequence corresponding to some gene with a reference DNA to find mutations that happened in the gene. In this case, it is essential to know the exact sequence of transformations since different mutation scenarios may lead to different outcomes.

Or, assume we are interested in the history of some language and want to understand how one word originated from another. To do this, we can try to compare the words using the edit distance metric. In this case, the exact sequence of transformations is crucial as well.

So, having the exact order of edit operations is essential in some cases. Let's consider an algorithm that allows finding these operations for two arbitrary strings.

Current topic:

Edit distance alignment

Topic depends on:

× Edit distance

Topic is required for:

Edit distance alignment in Java

§2. Reconstructing edit operations

For two arbitrary strings, we can reconstruct the sequence of edit operations using a table of intermediate answers. Recall that for $s = \text{bone}$ and $t = \text{brown}$ the table looks like this:

		b	r	o	w	n
	0	1	2	3	4	5
b	1	0	1	2	3	4
o	2	1	1	1	2	3
n	3	2	2	2	2	2
e	4	3	3	3	3	3

To reconstruct the edit operations, we can start from the lower right cell of the table (that contains $d_E(s, t)$) and move to the adjacent (upper, left, or diagonal) cell corresponding to the optimal transformation. We consider the string in the first column as the initial string and the string in the first row as the final result. Therefore, a move to the upper cell corresponds to deletion, a move to the left cell corresponds to insertion, and a diagonal movement corresponds to substitution. We need to continue moving through the table until we reach the upper-left cell. Then, considering all the operations in the reverse order, we will get the sequence of transformations corresponding to the edit distance.

§3. An example

Now let's see how we can reconstruct the edit operations for the example above. We begin from the lower right cell of the table:

		b	r	o	w	n
	0	1	2	3	4	5
b	1	0	1	2	3	4
o	2	1	1	1	2	3
n	3	2	2	2	2	2
e	4	3	3	3	3	3

In this case, we might get the final value either from the upper or from the diagonal cell. So, let's move to the diagonal cell (this movement corresponds to the substitution of "e" by "n"):

		b	r	o	w	n
	0	1	2	3	4	5
b	1	0	1	2	3	4
o	2	1	1	1	2	3
n	3	2	2	2	2	2
e	4	3	3	3	3	3

Now we can get the current value only from the cell on the diagonal (this movement corresponds to the substitution of "n" by "w"):

		b	r	o	w	n
	0	1	2	3	4	5
b	1	0	1	2	3	4
o	2	1	1	1	2	3
n	3	2	2	2	2	2
e	4	3	3	3	3	3

Again, we can get the current value only from the cell on the diagonal. Note that in this case, the corresponding symbols are the same, so the cost of this substitution is 0.

		b	r	o	w	n
	0	1	2	3	4	5
b	1	0	1	2	3	4
o	2	1	1	1	2	3
n	3	2	2	2	2	2
e	4	3	3	3	3	3

At this step, we can get the current value only from the cell on the left. This movement corresponds to the insertion of "r" after "b":

		b	r	o	w	n
	0	1	2	3	4	5
b	1	0	1	2	3	4
o	2	1	1	1	2	3
n	3	2	2	2	2	2
e	4	3	3	3	3	3

Finally, we need to perform a substitution (again with zero cost, since the corresponding symbols are the same):

		b	r	o	w	n
	0	1	2	3	4	5
b	1	0	1	2	3	4
o	2	1	1	1	2	3
n	3	2	2	2	2	2
e	4	3	3	3	3	3

So, to transform the word **bone** into the word **brown**, we need to perform a substitution first, then insert "r" after "b" and finally do three substitutions for the symbols "o", "n", and "e". Skipping the substitutions of the equal characters, we can represent the edit operations in the following way:

```
bone -> [insert "r" after "b"] ->
brone -> [substitute "n" by "w"] ->
browe -> [substitute "e" by "n"] ->
brown
```

At some steps, there are several optimal ways to choose an edit operation, meaning there may exist several optimal ways to transform one string into another.

§4. Alignment

A convenient way to represent a sequence of edit operations is by using a so-called **alignment**. The following alignment corresponds to the transformation sequence above:

b	-	o	n	e
b	r	o	w	n

The first row in this table corresponds to the initial word (**bone**), and the second row corresponds to the final word (**brown**). A column with a gap symbol in the upper cell and a letter in the lower corresponds to insertion. The opposite case corresponds to deletion. The columns having no gaps correspond to a substitution.

For the example above, the second column corresponds to insertion, and the last two columns are substitutions. The first and the third columns are substitutions having zero cost.

Since there may exist several ways to transform one string into another, the strings may have several alignments as well. An alignment corresponding to the optimal sequence of edit operations we will further call an **optimal alignment**.

§5. Complexity analysis

Assuming that for two strings s and t a table of intermediate answers is calculated, the sequence of edit operations can be reconstructed in $O(|s| + |t|)$, since on each step we process at least one symbol from one of the strings, and the total number of symbols is $|s| + |t|$.

In this case, we need to store the whole table of intermediate answers to reconstruct the operations, so the algorithm requires $O(|s| \cdot |t|)$ additional memory. However, it can be reduced to $O(min(|s|, |t|))$. Since the algorithm is rather complicated to be fully explained here, those who are interested may take a look at a corresponding Wikipedia article.

Report a typo

19 users liked this theory. 0 didn't like it. What about you?

Start practicing

This content was created almost 2 years ago and updated 2 days ago. Share your feedback below in comments to help us improve it!

Comments (1) Hints (0) Useful links (0) Show discussion