Core String Edits, Alignments, and Dynamic Programming

2015. 03. 10

Lee. Kang bae

Introduction

- **Inexact matching problem**
    - What is the inexact matching problem?
        - Equal to the edit distance problem
            - The most classic inexact matching problem

The edit distance between two strings

- **Edit distance**
  - To measure of the difference or distance between two strings by a series of edit operations
    - So, the edit distance between two strings is defined as the minimum number of edit operations

  - Edit operations are on transforming one string into the other on individual characters such as *I*, *D*, *R*, *M*
    - *I* : the insert operation
    - *D* : the delete operation
    - *R* : the substitute operation (or replace)
    - *M* : the match operation

The edit distance between two strings

- **What is valuable Edit distance?**
  - As the **minimum number** of edit operations except match needed to transform the first string in to the second.

	R	I	M	D	M	D	M	M	I
S_1	v		i	n	t	n	e	r	
S_2	w	r	i		t		e	r	s

	I	D	I	M	D	M	D	M	M	I
S_1		v		i	n	t	n	e	r	
S_2	w		r	i		t		e	r	s

The edit distance between two strings

- **Edit distance problem**
  - To compute the edit distance between two given strings, along with an optimal edit transcript
    - Optimal edit transcript
      - Edit transcript uses the minimum number of edit operations
    - Cooptimal edit transcript
      - there may be more than one optimal transcript

	R	R	R	M	D	M	M	I
S_1	v	i	n	t	n	e	r	
S_2	w	r	i	t		e	r	s

	R	I	M	D	M	D	M	M	I
S_1	v		i	n	t	n	e	r	
S_2	w	r	i		t		e	r	s

The edit distance between two strings

- **Edit transcript**
  - A string over the alphabet I, D, R, M that describes transformation of one string to another

	R	I	M	D	M	D	M	M	I
S_1	v		i	n	t	n	e	r	
S_2	w	r	i		t		e	r	s

String Alignment

- **String Alignment**
  - Display two compared strings by edit distance
  - Place the two resulting strings one above the other
    - So, every character or space in either string is opposite a unique character or a unique space in the other string

q	a	c	-	d	b	d
q	a	w	x	-	b	-

v	-	i	n	t	n	e	r	-
w	r	i	-	t	-	e	r	s

Dynamic programming calculation of edit distance

- **How to compute edit distance of edit transcript or alignment?**
  - Using dynamic programming
  - Dynamic programming
    - Solves problems by combining the solutions to subproblems

Dynamic programming calculation of edit distance

- ***D*(*i*, *j*)**
  - the edit distance of $S_1[1..i]$ and $S_2[1..j]$ For two strings S_1 and S_2
    - $D(1, 1) = 0$

S_1	q	**a**	-	c	d	b	d
S_2	q	a	w	x	-	b	-

    - $D(3, 3) = 1$

S_1	q	a	c	-	d	b	d
S_2	q	a	w	x	-	b	-

    - $D(2, 3) = 1$

S_1	q	a	-	c	d	b	d
S_2	q	a	w	x	-	b	-

Dynamic programming calculation of edit distance

- **The dynamic programming approach has three essential components**

1) The recurrence relation
2) The tabular computation
3) The traceback

Dynamic programming calculation of edit distance

- **The recurrence relation**
  - This establishes
    - A recursive relationship between the value of $D(i,j)$

      Ex. $D(i,j)$ is used to compute $D(i+1,j+1)$
    - For i and j both positive

Dynamic programming calculation of edit distance

- **The recurrence relation**
  - The base conditions
    - $D(i, 0) = i$
      - Because zero characters of S_2 is **deleted** all the i characters of S_1

	1	2	3	4	5	6
S_1	q	a	c	d	b	d
S_2	-	-	-	-	-	-

***D*(6, 0) = 6**

    - $D(0, j) = j$
      - Because zero characters of S_1 is **inserted** all the j characters of S_2

	1	2	3	4	5
S_1	-	-	-	-	-
S_2	q	a	w	x	b

***D*(0, 5) = 5**

Dynamic programming calculation of edit distance

- **The recurrence relation**
  - $D(i, j) = \min[D(i-1, j) + 1, D(i, j-1) + 1, D(i-1, j-1) + t(i, j)]$

  - Where t(i,j) is defiend
    - $t(i, j) = 1$, if $S_1(i) \neq S_2(j)$
    - $t(i, j) = 0$, if $S_1(i) = S_2(j)$

Dynamic programming calculation of edit distance

- **The recurrence relation**
  - Two string is given
    - *vintner*
    - *writers*

1	2	3	4	5	6	7
v	i	n	t	n	e	r
w	r	i	t	e	r	s

  - $D(2, 3) = 4$

v	**-**	**-**	**i**
w	**r**	**i**	**-**

  - $D(1, 3) = 3$

v	**-**	**-**
w	**r**	**i**

$$\boldsymbol{D(2, 3) = D(1, 3) + 1}$$

Dynamic programming calculation of edit distance

- **The recurrence relation**
  - Two string is given
    - *vintner*
    - *writers*

1	2	3	4	5	6	7
v	i	n	t	n	e	r
w	r	i	t	e	r	s

  - $D(2, 3) = 4$

v	**-**	**i**	**-**
w	**r**	**-**	**i**

  - $D(2, 2) = 3$

v	**-**	**i**
w	**r**	**-**

$$\boldsymbol{D(2, 3) = D(2, 2) + 1}$$

Dynamic programming calculation of edit distance

- **The recurrence relation**
  - Two string is given
    - *vintner*
    - *writers*

1	2	3	4	5	6	7
v	i	n	t	n	e	r
w	r	i	t	e	r	s

  - $D(2, 3) = 2$

v	**-**	**i**
w	**r**	**i**

  - $D(1, 2) = 2$
    - $S_1(2) = S_2(3)$

v	**-**
w	**r**

***D*(2, 3) = *D*(1, 2)**

Dynamic programming calculation of edit distance

- **Correctness of the general recurrence**
  - Establish correctness in the next two lemmas using the concept of an edit transcript
    - Lemma 11.3.1
      - The value of $D(i, j)$ must be $D(i, j-1) + 1$, $D(i-1, j) + 1$, or $D(i, j) + t(i, j)$. There are no other possibilities.
    - Lemma 11.3.2
      - $D(i, j) \leq \min[D(i-1, j) + 1, D(i, j-1) + 1, D(i-1, j-1) + t(i, j)]$

Dynamic programming calculation of edit distance

- **Theorem 11.3.1**
  - When both i and j are strictly positive,
    $D(i, j) = \min[D(i - 1, j) + 1, D(i, j - 1) + 1, D(i - 1, j - 1) + t(i, j)]$

Dynamic programming calculation of edit distance

- **Tabular computation of edit distance**
  - It is to use the **bottom-up**
    - First, compute $D(i, j)$ for the smallest possible values for i and j
    - And then compute values of $D(i, j)$ for increasing values of i and j

Dynamic programming calculation of edit distance

- **Tabular computation of edit distance**
  - Bottom-up computation used a size $(i+1)*(j+1)$ of dynamic programming table
    - Vertical axis is string S_1
    - Horizontal axis is string S_2
    - Ex) $S_1[1..3]$ and $S_2[1..3]$,
      Size of table = 4*4

$D(i, j)$		0	1	2	3
			a	b	c
0		0	1	2	3
1	a	1	0	1	2
2	b	2	1	0	1
3	d	3	2	1	1

Dynamic programming calculation of edit distance

- **Tabular computation of edit distance**
  - $S_1 = w\ r\ i\ t\ e\ r\ s$
  - $S_2 = v\ i\ n\ t\ n\ e\ r$

  - Compute edit distance of S_1 and S_2
    - Hence, Compute $D(m, n)$

Dynamic programming calculation of edit distance

- **Tabular computation of edit distance**

$D(i, j)$		0	1	2	3	4	5	6	7
			w	r	i	t	e	r	s
0		0	1	2	3	4	5	6	7
1	v	1							
2	i	2							
3	n	3							
4	t	4							
5	n	5							
6	e	6							
7	r	7							

Dynamic programming calculation of edit distance

- **Tabular computation of edit distance**

$D(i, j)$		0	1	2	3	4	5	6	7
			w	r	i	t	e	r	s
0		0	1	2	3	4	5	6	7
1	v	1	1	2	3	4	5	6	7
2	i	2	2	2	2	3	4	5	6
3	n	3	3	3	3	3	4	5	6
4	t	4	4	4	4				
5	n	5							
6	e	6							
7	r	7							

Dynamic programming calculation of edit distance

- **Tabular computation of edit distance**

D(*i*, *j*)		0	1	2	3	4	5	6	7
			w	r	i	t	e	r	s
0		0	1	2	3	4	5	6	7
1	v	1	1	2	3	4	5	6	7
2	i	2	2	2	2	3	4	5	6
3	n	3	3	3	3	3	4	5	6
4	t	4	4	4	4				
5	n	5							
6	e	6							
7	r	7							

3
5
4

Dynamic programming calculation of edit distance

- **Tabular computation of edit distance**

D(*i*, *j*)		0	1	2	3	4	5	6	7
			w	r	i	t	e	r	s
0		0	1	2	3	4	5	6	7
1	v	1	1	2	3	4	5	6	7
2	i	2	2	2	2	3	4	5	6
3	n	3	3	3	3	3	4	5	6
4	t	4	4	4	4	3			
5	n	5							
6	e	6							
7	r	7							

Dynamic programming calculation of edit distance

- **Tabular computation of edit distance**

$D(i, j)$		0	1	2	3	4	5	6	7
			w	r	i	t	e	r	s
0		0	1	2	3	4	5	6	7
1	v	1	1	2	3	4	5	6	7
2	i	2	2	2	2	3	4	5	6
3	n	3	3	3	3	3	4	5	6
4	t	4	4	4	4	3	4	5	6
5	n	5	5	5	5	4	4	5	6
6	e	6	6	6	6	5	4	5	6
7	r	7	7	6	7	6	5	4	5

Dynamic programming calculation of edit distance

- **Tabular computation of edit distance**
  - Theorem 11.3.2 (= Time complexity)
    - Using dynamic programming, edit distance $D(n, m)$ can be computed in $O(nm)$ time
    - Because the dynamic programming table for computing the edit distance between a string of length n and a string of length m can be filled in with $O(nm)$ work.

$D(i, j)$		0	1	2	…	n
			w	r	…	t
0		0	← 1	← 2	…	← 4
1	v	↑ 1	↖ 1	↖←2	…	↖←4
2	i	↑ 2	↖ ↑2	↖ 2	…	← 3
…	…	…	…	…	…	↖ 3
m	t	↑ 4	↖ ↑4	↖ ↑4	↖ ↑4	↖ 3

m

n

Dynamic programming calculation of edit distance

- **The traceback**
  - How is the associated optimal edit transcript extracted?
    - The easiest way is to establish pointers in the table when the table value are computed
    - The pointers allow easy recovery of an optimal edit transcript

Dynamic programming calculation of edit distance

- **The traceback**

$D(i, j)$		0	1	2	3	4	5	6	7
			w	r	i	t	e	r	s
0		0	← 1	← 2	← 3	← 4	← 5	← 6	← 7
1	v	↑ 1							
2	i	↑ 2							
3	n	↑ 3							
4	t	↑ 4							
5	n	↑ 5							
6	e	↑ 6							
7	r	↑ 7							

Dynamic programming calculation of edit distance

- **The traceback**

$D(i, j)$		0	1	2	3	4	5	6	7
			w	r	i	t	e	r	s
0		0	← 1	← 2	← 3	← 4	← 5	← 6	← 7
1	v	↑ 1	↖ 1	↖←2	↖←3	↖←4	↖←5	↖←6	↖←7
2	i	↑ 2	↖↑2	↖ 2	↖ 2	← 3	← 4	← 5	← 6
3	n	↑ 3	↖↑3	↖↑3	↖↑3	↖ 3	↖←4	↖←5	↖←6
4	t	↑ 4	↖↑4	↖↑4	↖↑4	↖ 3	↖←4	↖←5	↖←6
5	n	↑ 5	↖↑5	↖↑5	↖↑5	↑ 4	↖ 4	↖←5	↖←6
6	e	↑ 6	↖↑6	↖↑6	↖↑6	↑ 5	↖ 4	↖←5	↖←6
7	r	↑ 7	↖↑7	↖ 6	↑↖←7	↑ 6	↑ 5	↖ 4	← 5

Dynamic programming calculation of edit distance

- **The traceback**

D(*i*, *j*)		0	1	2	3	4	5	6	7
			w	r	i	t	e	r	s
0		0	← 1	← 2	← 3	← 4	← 5	← 6	← 7
1	v	↑ 1	↖ 1	↖←2	↖←3	↖←4	↖←5	↖←6	↖←7
2	i	↑ 2	↖↑2	↖ 2	↖ 2	← 3	← 4	← 5	← 6
3	n	↑ 3	↖↑3	↖↑3	↖↑3	↖ 3	↖←4	↖←5	↖←6
4	t	↑ 4	↖↑4	↖↑4	↖↑4	↖ 3	↖←4	↖←5	↖←6
5	n	↑ 5	↖↑5	↖↑5	↖↑5	↑ 4	↖ 4	↖←5	↖←6
6	e	↑ 6	↖↑6	↖↑6	↖↑6	↑ 5	↖ 4	↖←5	↖←6
7	r	↑ 7	↖↑7	↖ 6	↑↖←7	↑ 6	↑ 5	↖ 4	← 5

Dynamic programming calculation of edit distance

- **The traceback**

D(i, j)		0	1	2	3	4	5	6	7
			w	r	i	t	e	r	s
0		0	← 1	← 2	← 3	← 4	← 5	← 6	← 7
1	v	↑ 1	↖ 1	↖←2	↖←3	↖←4	↖←5	↖←6	↖←7
2	i	↑ 2	↖↑2	↖ 2	↖ 2	← 3	← 4	← 5	← 6
3	n	↑ 3	↖↑3	↖↑3	↖↑3	↖ 3	↖←4	↖←5	↖←6
4	t	↑ 4	↖↑4	↖↑4	↖↑4	↖ 3	↖←4	↖←5	↖←6
5	n	↑ 5	↖↑5	↖↑5	↖↑5	↑ 4	↖ 4	↖←5	↖←6
6	e	↑ 6	↖↑6	↖↑6	↖↑6	↑ 5	↖ 4	↖←5	↖←6
7	r	↑ 7	↖↑7	↖ 6	↑↖←7	↑ 6	↑ 5	↖ 4	← 5

Dynamic programming calculation of edit distance

- **The traceback**

$D(i, j)$		0	1	2	3	4	5	6	7
			w	r	i	t	e	r	s
0		0	← 1	← 2	← 3	← 4	← 5	← 6	← 7
1	v	↑ 1	↖ 1	↖←2	↖←3	↖←4	↖←5	↖←6	↖←7
2	i	↑ 2	↖↑2	↖ 2	↖ 2	← 3	← 4	← 5	← 6
3	n	↑ 3	↖↑3	↖↑3	↖↑3	↖ 3	↖←4	↖←5	↖←6
4	t	↑ 4	↖↑4	↖↑4	↖↑4	↖ 3	↖←4	↖←5	↖←6
5	n	↑ 5	↖↑5	↖↑5	↖↑5	↑ 4	↖ 4	↖←5	↖←6
6	e	↑ 6	↖↑6	↖↑6	↖↑6	↑ 5	↖ 4	↖←5	↖←6
7	r	↑ 7	↖↑7	↖ 6	↑↖←7	↑ 6	↑ 5	↖ 4	← 5

Dynamic programming calculation of edit distance

- **The traceback**

D(*i*, *j*)		0	1	2	3	4	5	6	7
			w	r	i	t	e	r	s
0		0	← 1	← 2	← 3	← 4	← 5	← 6	← 7
1	v	↑ 1	↖ 1	↖←2	↖←3	↖←4	↖←5	↖←6	↖←7
2	i	↑ 2	↖↑2	↖ 2	↖ 2	← 3	← 4	← 5	← 6
3	n	↑ 3	↖↑3	↖↑3	↖↑3	↖ 3	↖←4	↖←5	↖←6
4	t	↑ 4	↖↑4	↖↑4	↖↑4	↖ 3	↖←4	↖←5	↖←6
5	n	↑ 5	↖↑5	↖↑5	↖↑5	↑ 4	↖ 4	↖←5	↖←6
6	e	↑ 6	↖↑6	↖↑6	↖↑6	↑ 5	↖ 4	↖←5	↖←6
7	r	↑ 7	↖↑7	↖ 6	↑↖←7	↑ 6	↑ 5	↖ 4	← 5

Dynamic programming calculation of edit distance

- **The traceback**

w	r	i	t	-	e	r	s
v	i	n	t	n	e	r	-

w	r	i	-	t	-	e	r	s
v	-	i	n	t	n	e	r	-

w	r	i	-	t	-	e	r	s
-	v	i	n	t	n	e	r	-

D(*i*, *j*)		0	1	2	3	4	5	6	7
			w	r	i	t	e	r	s
0		0	← 1	← 2	← 3	← 4	← 5	← 6	← 7
1	v	↑ 1	↖ 1	↖←2	↖←3	↖←4	↖←5	↖←6	↖←7
2	i	↑ 2	↖↑2	↖ 2	↖ 2	← 3	← 4	← 5	← 6
3	n	↑ 3	↖↑3	↖↑3	↖↑3	↖ 3	↖←4	↖←5	↖←6
4	t	↑ 4	↖↑4	↖↑4	↖↑4	↖ 3	↖←4	↖←5	↖←6
5	n	↑ 5	↖↑5	↖↑5	↖↑5	↑ 4	↖ 4	↖←5	↖←6
6	e	↑ 6	↖↑6	↖↑6	↖↑6	↑ 5	↖ 4	↖←5	↖←6
7	r	↑ 7	↖↑7	↖ 6	↑↖←7	↑ 6	↑ 5	↖ 4	← 5

Dynamic programming calculation of edit distance

- **The traceback**

w	r	i	t	-	e	r	s
v	i	n	t	n	e	r	-

w	r	i	-	t	-	e	r	s
v	-	i	n	t	n	e	r	-

w	r	i	-	t	-	e	r	s
-	v	i	n	t	n	e	r	-

D(*i*, *j*)		0	1	2	3	4	5	6	7
			w	r	i	t	e	r	s
0		0	← 1	← 2	← 3	← 4	← 5	← 6	← 7
1	v	↑ 1	↖ 1	↖←2	↖←3	↖←4	↖←5	↖←6	↖←7
2	i	↑ 2	↖↑2	↖ 2	↖ 2	← 3	← 4	← 5	← 6
3	n	↑ 3	↖↑3	↖↑3	↖↑3	↖ 3	↖←4	↖←5	↖←6
4	t	↑ 4	↖↑4	↖↑4	↖↑4	↖ 3	↖←4	↖←5	↖←6
5	n	↑ 5	↖↑5	↖↑5	↖↑5	↑ 4	↖ 4	↖←5	↖←6
6	e	↑ 6	↖↑6	↖↑6	↖↑6	↑ 5	↖ 4	↖←5	↖←6
7	r	↑ 7	↖↑7	↖ 6	↑↖←7	↑ 6	↑ 5	↖ 4	← 5

Dynamic programming calculation of edit distance

- **The traceback**

w	r	i	t	-	e	r	s
v	i	n	t	n	e	r	-

w	r	i	-	t	-	e	r	s
v	-	i	n	t	n	e	r	-

w	r	i	-	t	-	e	r	s
-	v	i	n	t	n	e	r	-

D(*i*, *j*)		0	1	2	3	4	5	6	7
			w	r	i	t	e	r	s
0		0	← 1	← 2	← 3	← 4	← 5	← 6	← 7
1	v	↑ 1	↖ 1	↖←2	↖←3	↖←4	↖←5	↖←6	↖←7
2	i	↑ 2	↖↑2	↖ 2	↖ 2	← 3	← 4	← 5	← 6
3	n	↑ 3	↖↑3	↖↑3	↖↑3	↖ 3	↖←4	↖←5	↖←6
4	t	↑ 4	↖↑4	↖↑4	↖↑4	↖ 3	↖←4	↖←5	↖←6
5	n	↑ 5	↖↑5	↖↑5	↖↑5	↑ 4	↖ 4	↖←5	↖←6
6	e	↑ 6	↖↑6	↖↑6	↖↑6	↑ 5	↖ 4	↖←5	↖←6
7	r	↑ 7	↖↑7	↖ 6	↑↖←7	↑ 6	↑ 5	↖ 4	← 5

Dynamic programming calculation of edit distance

- **The traceback**

w	r	i	t	-	e	r	s
v	i	n	t	n	e	r	-

w	r	i	-	t	-	e	r	s
v	-	i	n	t	n	e	r	-

w	r	i	-	t	-	e	r	s
-	v	i	n	t	n	e	r	-

D(*i*, *j*)		0	1	2	3	4	5	6	7
			w	r	i	t	e	r	s
0		0	← 1	← 2	← 3	← 4	← 5	← 6	← 7
1	v	↑ 1	↖ 1	↖←2	↖←3	↖←4	↖←5	↖←6	↖←7
2	i	↑ 2	↖↑2	↖ 2	↖ 2	← 3	← 4	← 5	← 6
3	n	↑ 3	↖↑3	↖↑3	↖↑3	↖ 3	↖←4	↖←5	↖←6
4	t	↑ 4	↖↑4	↖↑4	↖↑4	↖ 3	↖←4	↖←5	↖←6
5	n	↑ 5	↖↑5	↖↑5	↖↑5	↑ 4	↖ 4	↖←5	↖←6
6	e	↑ 6	↖↑6	↖↑6	↖↑6	↑ 5	↖ 4	↖←5	↖←6
7	r	↑ 7	↖↑7	↖ 6	↑↖←7	↑ 6	↑ 5	↖ 4	← 5

Dynamic programming calculation of edit distance

- **The traceback**
  - Theorem 11.3.3. (= Time complexity)
    - Once the dynamic programming table with pointers has been computed, an optimal edit transcript can be found in $O(n + m)$ time.
      - This case is ***worst case***

$D(i, j)$		0	1	2	…	n
			w	r	…	t
0		0	← 1	← 2	…	← 4
1	v	↑ 1	↖ 1	↖ ←2	…	↖ ←4
2	i	↑ 2	↖ ↑ 2	↖ 2	…	← 3
…	…	…	…	…	…	↖ 3
m	t	↑ 4	↖ ↑ 4	↖ ↑ 4	↖ ↑ 4	↖ 3

m

n

Dynamic programming calculation of edit distance

- **The traceback**
  - Theorem 11.3.4.
    - Any path from (n, m) to (0, 0) specifies an edit transcript with the minimum number of edit operations
    - Conversely, any optimal edit transcript is specified by such a path.
    - Moreover, since a path describes only one transcript, the correspondence between paths and optimal transcripts is **one to one**
      - **The number of path from (n, m) to (0, 0) = the number of optimal edit transcript**