Text Analytics HWS 24/25
Jörg Schlötterer

Exercise 03: Minimum Edit Distance

Pen & Paper Tasks

1 String Alignment

In this task we want to compute an alignment with minimum edit distance between the source string **"sitting"** and the target string **"kitten"**.

Fill in the edit distance table below by using the algorithm from the lecture, where deletions, insertions and substitutions all have a cost of 1. Remember to add pointers to indicate from which cell you came from. Specify one possible alignment of the strings with minimum edit distance and indicate their distance.

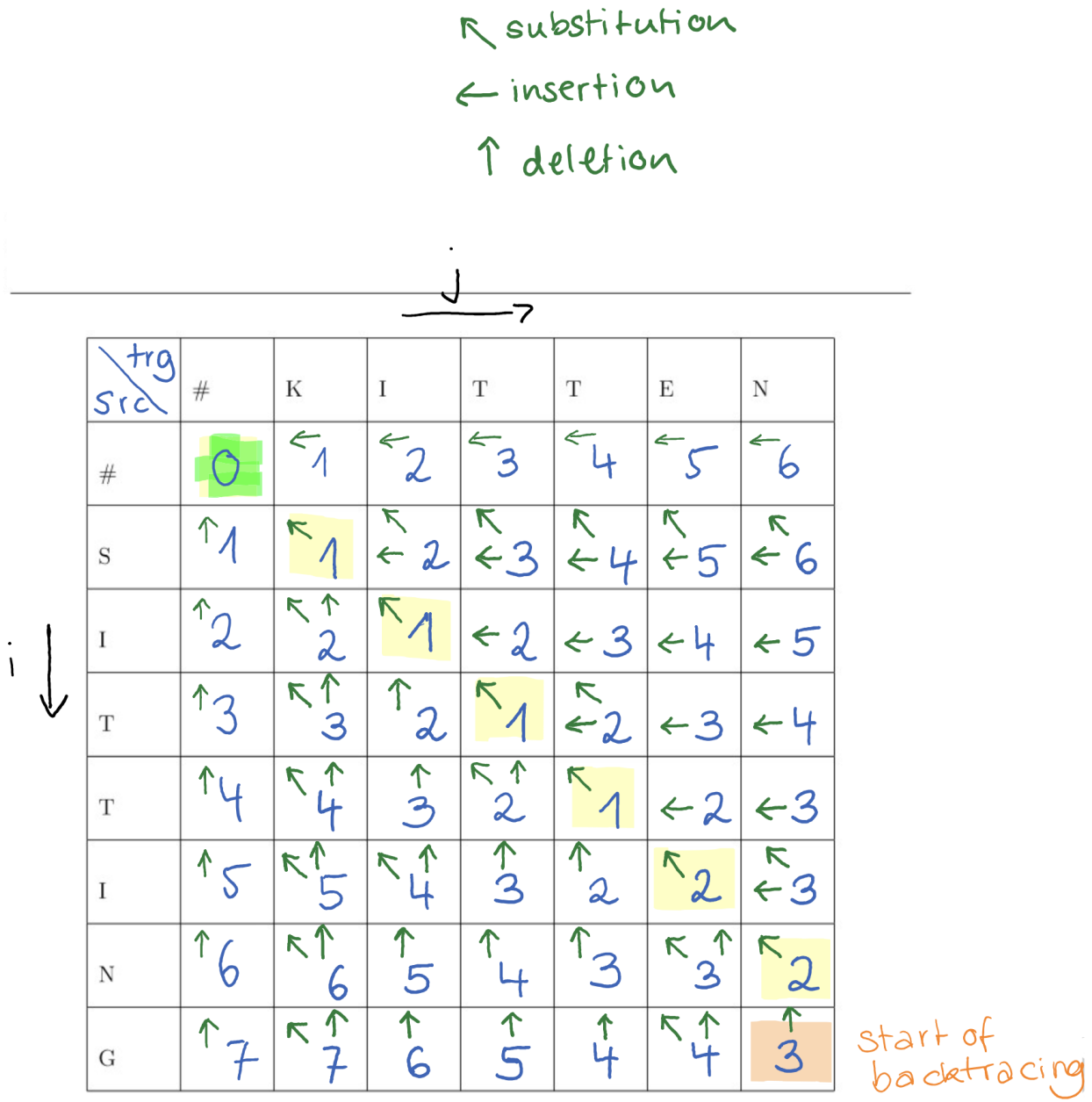
Initialization

$D(i, 0) = i$
$D(0, j) = j$

Recurrence Relation
For each $i = 1 \dots m$
For each $j = 1 \dots n$

$$D(i,j) = min \begin{cases} D(i-1,j) + 1 \\ D(i,j-1) + 1 \\ D(i-1,j-1) \begin{cases} +1 & \text{if } X(i) \neq Y(j) \\ +0 & \text{if } X(i) = Y(j) \end{cases} \end{cases}$$

Termination
$D(n, m)$ is distance



src \ trg	#	K	I	T	T	E	N
#	0	←1	←2	←3	←4	←5	←6
S	↑1	↖1	↖←2	↖←3	↖←4	↖←5	↖←6
I	↑2	↖↑2	↖1	←2	←3	←4	←5
T	↑3	↖↑3	↑2	↖1	↖←2	←3	←4
T	↑4	↖↑4	↑3	↖↑2	↖1	←2	←3
I	↑5	↖↑5	↖↑4	↑3	↑2	↖2	↖←3
N	↑6	↖↑6	↑5	↑4	↑3	↖↑3	↖2
G	↑7	↖↑7	↑6	↑5	↑4	↖↑4	↑3

Min edit distance: 3

Alignment:

```
source  S I T T I N G
target  K I T T E N *
cost    1       1   1
op      S       S   D
```

2 Subtleties in Regular Expressions

What are the differences between these regular expressions? You may use examples to support your explanations.

a) i. `r"a\.?b"`
   ii. `r"a.?b"`
   iii. `r"a\.b"`

b) i. `r"^[a-z]"`
   ii. `r"[^a-z]"`

c) i. `r"<\w+>"`
   ii. `r"<(\w+)>"`
   iii. `r"<(:?\w+)>"`
   iv. `r"<(?:\w+)>"`

Solution

a) i. `a` followed potentially by `.` then `b`
   ii. `a` followed potentially by any character then `b`
   iii. the exact sequence `a.b`

b) i. A lowercase letter at the beginning of the string/line
   ii. A character that is not a lowercase letter

c) Consider the example sequence `<test> <:test>` and the application of `re.findall()`
   i. The result is `<test>` (`<` followed by one or more alphanumeric characters then `>`)
   ii. The result is `test` (Parentheses indicates a capture group and if a single capturing group is present, `re.findall()` returns this group)
   iii. The result is `test` and `:test` (same situation with capture group, `:?` allows an optional `:` at the beginning of the group, before the alphanumeric characters)
   iv. The result is `<test>` (essentially like i) $\Rightarrow$ `?:` makes the parentheses non-capturing