Natural Language Processing Assignment 2

Intent

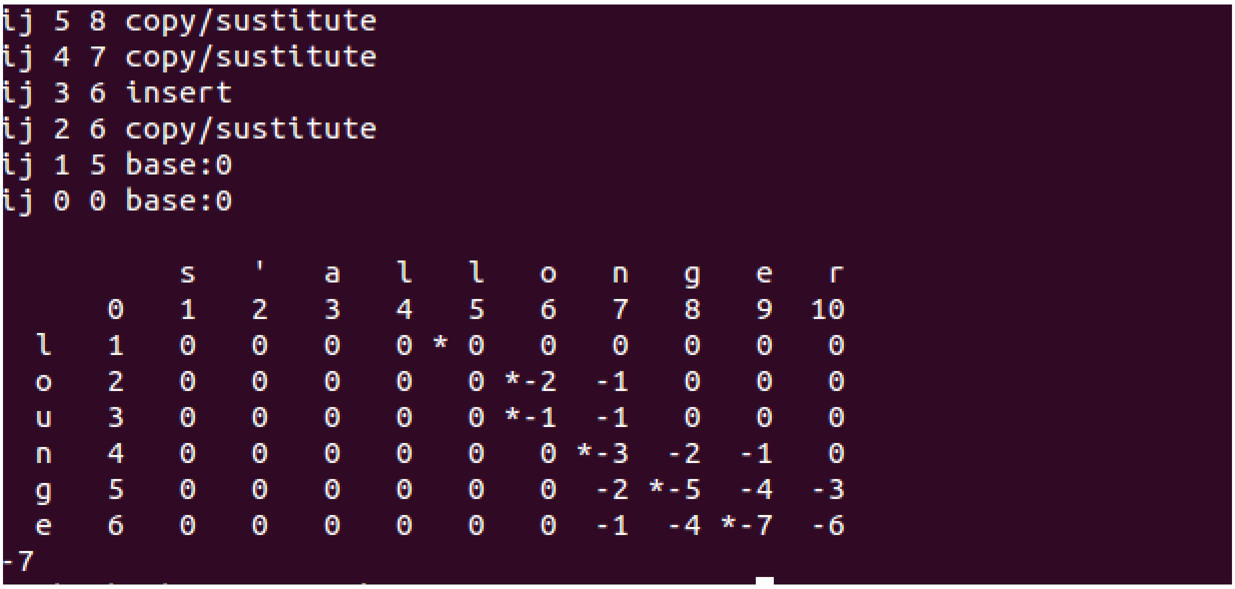
In this Assignment , I have implemented

1. Smith-Waterman Algorithm
2. Affine Gap Algorithm
3. Spell Checker that uses an online dictionary and returns the best match

1.Smith-Waterman Algorithm

I implemented the smith-waterman Algorithm by modifying the stredit.stredit2 function

Output of this Algorithm



```
ij 5 8 copy/sustitute
ij 4 7 copy/sustitute
ij 3 6 insert
ij 2 6 copy/sustitute
ij 1 5 base:0
ij 0 0 base:0

          s   '   a   l   l   o   n   g   e   r
      0   1   2   3   4   5   6   7   8   9  10
  l   1   0   0   0   0 * 0   0   0   0   0   0
  o   2   0   0   0   0   0 *-2  -1   0   0   0
  u   3   0   0   0   0   0 *-1  -1   0   0   0
  n   4   0   0   0   0   0   0 *-3  -2  -1   0
  g   5   0   0   0   0   0   0  -2 *-5  -4  -3
  e   6   0   0   0   0   0   0  -1  -4 *-7  -6
-7
```

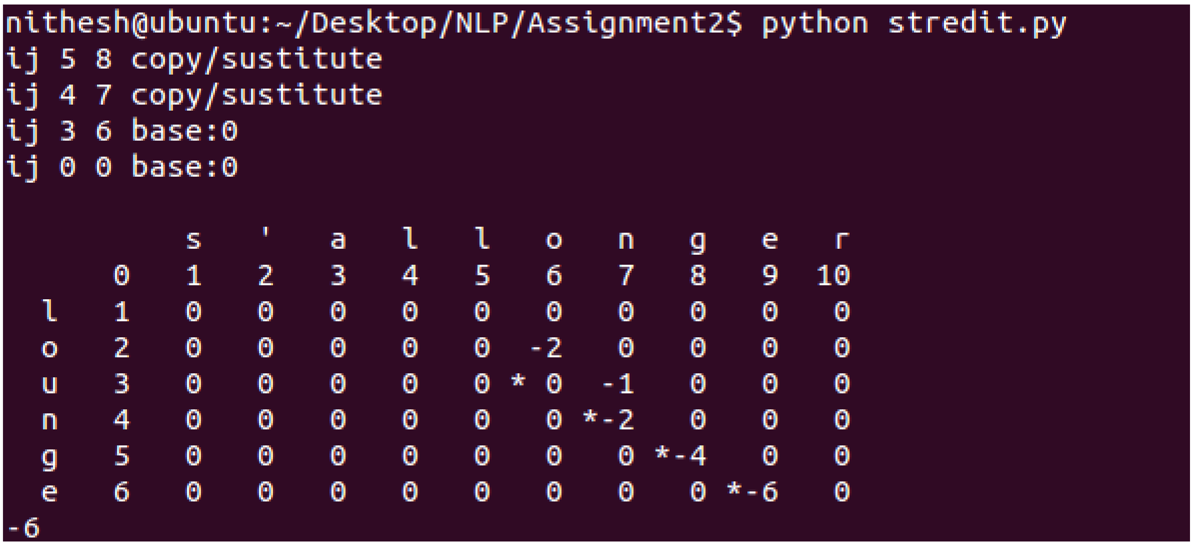
For this case the distances used were : insert/delete : 1 / copy cost : -2 / subst cost : 1 / base : 0

Hence the matched word is longer .

Analysis of this Algorithm

I tried different experiments with the distances for insert/delete/copy and substitute.

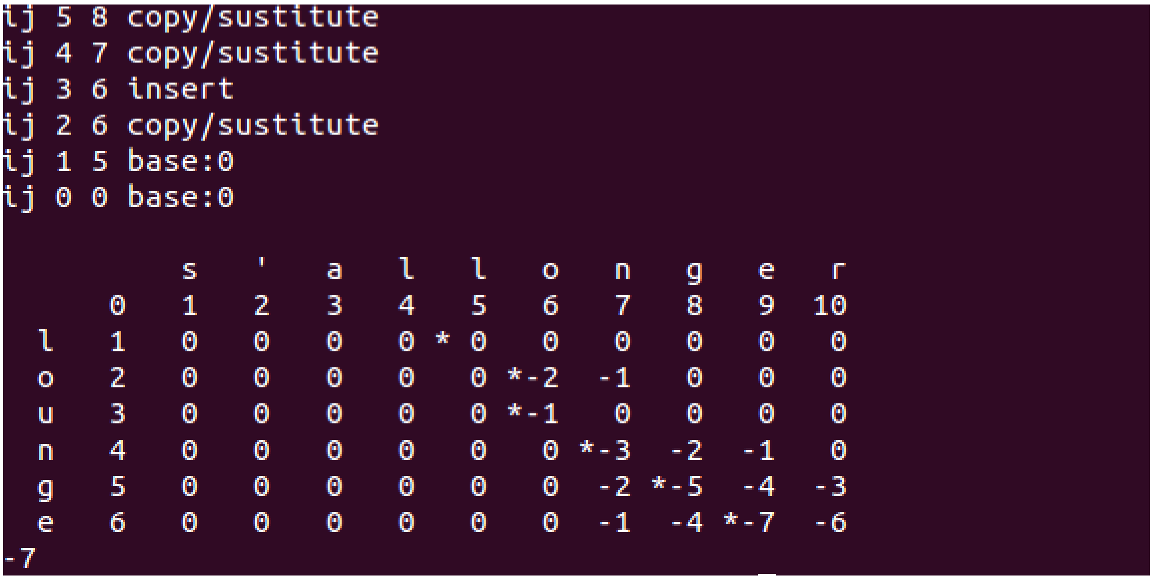
Case 1: For insert/delete : 100 copy : -2 substitute : 1



```
nithesh@ubuntu:~/Desktop/NLP/Assignment2$ python stredit.py
ij 5 8 copy/sustitute
ij 4 7 copy/sustitute
ij 3 6 base:0
ij 0 0 base:0

          s   '   a   l   l   o   n   g   e   r
      0   1   2   3   4   5   6   7   8   9  10
  l   1   0   0   0   0   0   0   0   0   0   0
  o   2   0   0   0   0   0  -2   0   0   0   0
  u   3   0   0   0   0   0 * 0  -1   0   0   0
  n   4   0   0   0   0   0   0 *-2   0   0   0
  g   5   0   0   0   0   0   0   0 *-4   0   0
  e   6   0   0   0   0   0   0   0   0 *-6   0
-6
```

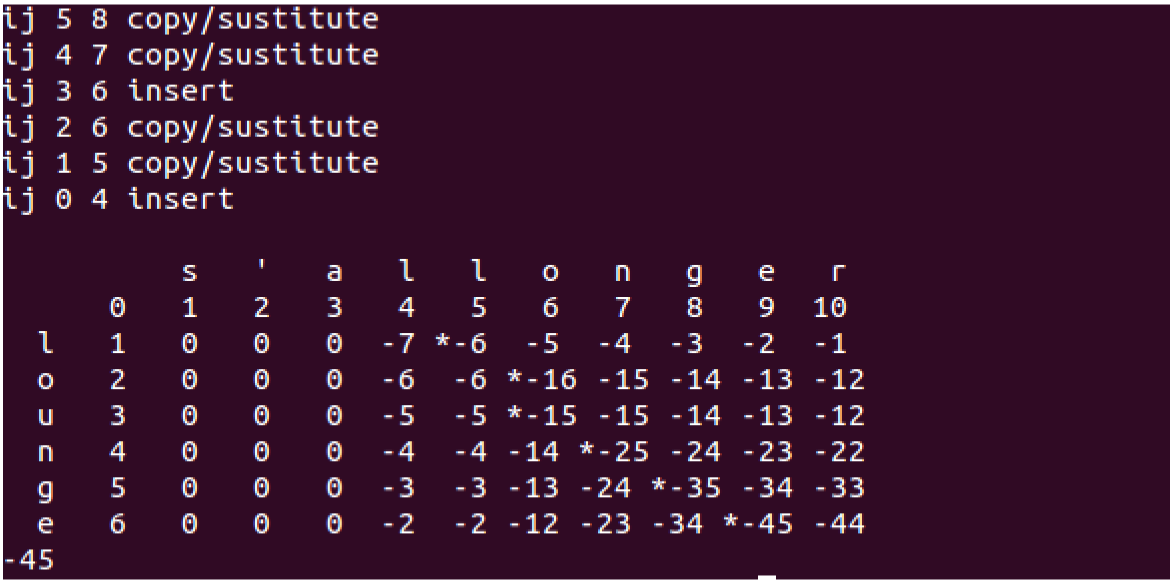
Case 2: For insert/delete : 1 copy : -2 substitute : 100



```
ij 5 8 copy/sustitute
ij 4 7 copy/sustitute
ij 3 6 insert
ij 2 6 copy/sustitute
ij 1 5 base:0
ij 0 0 base:0

          s   '   a   l   l   o   n   g   e   r
      0   1   2   3   4   5   6   7   8   9  10
  l   1   0   0   0   0 * 0   0   0   0   0   0
  o   2   0   0   0   0   0 *-2  -1   0   0   0
  u   3   0   0   0   0   0 *-1   0   0   0   0
  n   4   0   0   0   0   0   0 *-3  -2  -1   0
  g   5   0   0   0   0   0   0  -2 *-5  -4  -3
  e   6   0   0   0   0   0   0  -1  -4 *-7  -6
-7
```

Case 3: For insert/delete : 1 copy : -10 substitute : 1



```
ij 5 8 copy/sustitute
ij 4 7 copy/sustitute
ij 3 6 insert
ij 2 6 copy/sustitute
ij 1 5 copy/sustitute
ij 0 4 insert

          s   '   a   l   l   o   n   g   e   r
      0   1   2   3   4   5   6   7   8   9  10
  l   1   0   0   0  -7 *-6  -5  -4  -3  -2  -1
  o   2   0   0   0  -6  -6*-16 -15 -14 -13 -12
  u   3   0   0   0  -5  -5*-15 -15 -14 -13 -12
  n   4   0   0   0  -4  -4 -14*-25 -24 -23 -22
  g   5   0   0   0  -3  -3 -13 -24*-35 -34 -33
  e   6   0   0   0  -2  -2 -12 -23 -34*-45 -44
-45
```

Result

For the case 1 , due to high insert and delete cost , the system preferred copy and substitute and hence the words matched were more or less of the same size

For the second case , since the substitution was more expensive , the system preferred copy or insert/delete and hence the output could delete/add more characters but did not substitute any characters

For the third case , the system would always prefer copy which could happen only if the same character occurs in both the strings . Hence the increase in this size did not have much effect . Independent of the magnitude , copy was always chosen

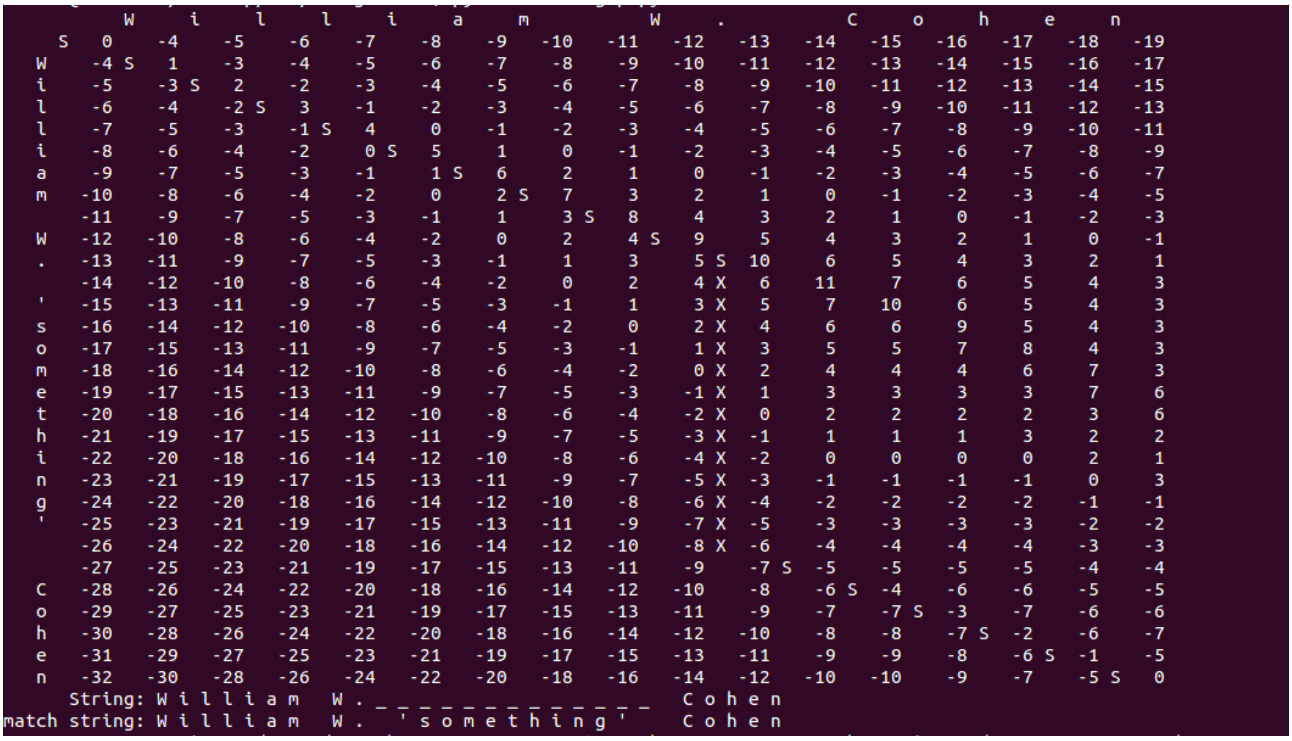
2.Affine Gap Algorithm

I implemented the Affine gap algorithm after referring to the material available at

http://pages.cs.wisc.edu/~bsettles/ibs08/lectures/02-alignment.pdf

In this algorithm , we compute the max function and positive distances for copy and negative distances for inserting gaps.

Output of this Algorithm



```
         W      i      l      l      i      a      m             W      .             C      o      h      e      n
   S    0     -4     -5     -6     -7     -8     -9    -10    -11    -12    -13    -14    -15    -16    -17    -18    -19
  W    -4 S    1     -3     -4     -5     -6     -7     -8     -9    -10    -11    -12    -13    -14    -15    -16    -17
  i    -5     -3 S    2     -2     -3     -4     -5     -6     -7     -8     -9    -10    -11    -12    -13    -14    -15
  l    -6     -4     -2 S    3     -1     -2     -3     -4     -5     -6     -7     -8     -9    -10    -11    -12    -13
  l    -7     -5     -3     -1 S    4      0     -1     -2     -3     -4     -5     -6     -7     -8     -9    -10    -11
  i    -8     -6     -4     -2      0 S    5      1      0     -1     -2     -3     -4     -5     -6     -7     -8     -9
  a    -9     -7     -5     -3     -1      1 S    6      2      1      0     -1     -2     -3     -4     -5     -6     -7
  m   -10     -8     -6     -4     -2      0      2 S    7      3      2      1      0     -1     -2     -3     -4     -5
      -11     -9     -7     -5     -3     -1      1      3 S    8      4      3      2      1      0     -1     -2     -3
  W   -12    -10     -8     -6     -4     -2      0      2      4 S    9      5      4      3      2      1      0     -1
  .   -13    -11     -9     -7     -5     -3     -1      1      3      5 S   10      6      5      4      3      2      1
      -14    -12    -10     -8     -6     -4     -2      0      2      4 X    6     11      7      6      5      4      3
  '   -15    -13    -11     -9     -7     -5     -3     -1      1      3 X    5      7     10      6      5      4      3
  s   -16    -14    -12    -10     -8     -6     -4     -2      0      2 X    4      6      6      9      5      4      3
  o   -17    -15    -13    -11     -9     -7     -5     -3     -1      1 X    3      5      5      7      8      4      3
  m   -18    -16    -14    -12    -10     -8     -6     -4     -2      0 X    2      4      4      4      6      7      3
  e   -19    -17    -15    -13    -11     -9     -7     -5     -3     -1 X    1      3      3      3      3      7      6
  t   -20    -18    -16    -14    -12    -10     -8     -6     -4     -2 X    0      2      2      2      2      3      6
  h   -21    -19    -17    -15    -13    -11     -9     -7     -5     -3 X   -1      1      1      1      3      2      2
  i   -22    -20    -18    -16    -14    -12    -10     -8     -6     -4 X   -2      0      0      0      0      2      1
  n   -23    -21    -19    -17    -15    -13    -11     -9     -7     -5 X   -3     -1     -1     -1     -1      0      3
  g   -24    -22    -20    -18    -16    -14    -12    -10     -8     -6 X   -4     -2     -2     -2     -2     -1     -1
  '   -25    -23    -21    -19    -17    -15    -13    -11     -9     -7 X   -5     -3     -3     -3     -3     -2     -2
      -26    -24    -22    -20    -18    -16    -14    -12    -10     -8 X   -6     -4     -4     -4     -4     -3     -3
      -27    -25    -23    -21    -19    -17    -15    -13    -11     -9     -7 S   -5     -5     -5     -5     -4     -4
  C   -28    -26    -24    -22    -20    -18    -16    -14    -12    -10     -8     -6 S   -4     -6     -6     -5     -5
  o   -29    -27    -25    -23    -21    -19    -17    -15    -13    -11     -9     -7     -7 S   -3     -7     -6     -6
  h   -30    -28    -26    -24    -22    -20    -18    -16    -14    -12    -10     -8     -8     -7 S   -2     -6     -7
  e   -31    -29    -27    -25    -23    -21    -19    -17    -15    -13    -11     -9     -9     -8     -6 S   -1     -5
  n   -32    -30    -28    -26    -24    -22    -20    -18    -16    -14    -12    -10    -10     -9     -7     -5 S    0
     String: W i l l i a m   W . _ _ _ _ _ _ _ _ _ _ _ _ _ _   C o h e n
match string: W i l l i a m   W .  ' s o m e t h i n g '      C o h e n
```

I have attached the snapshot of the output .

Here this table is a abstracted version of the original table .

A 3d table is produced by the algorithm and we choose just the most optimal path and print it

Here the trace is represented using

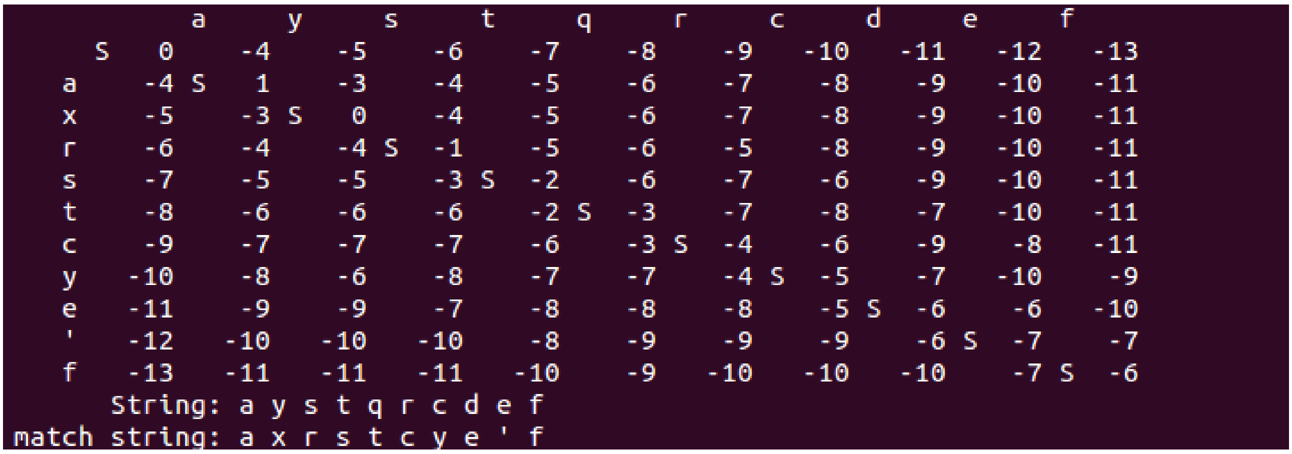
S – substitution

X – Insert a gap in the X string

Y – Insert a gap in the Y string

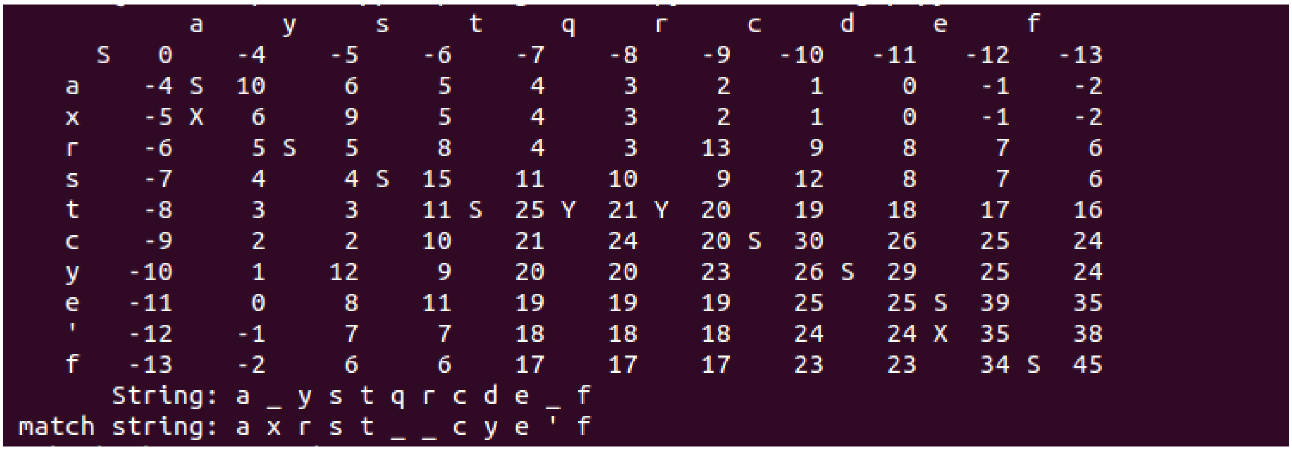
Analysis of this Algorithm

Case 1 : Copy cost 1(Only if the characters match) / Insertion Cost -1(X/Y)



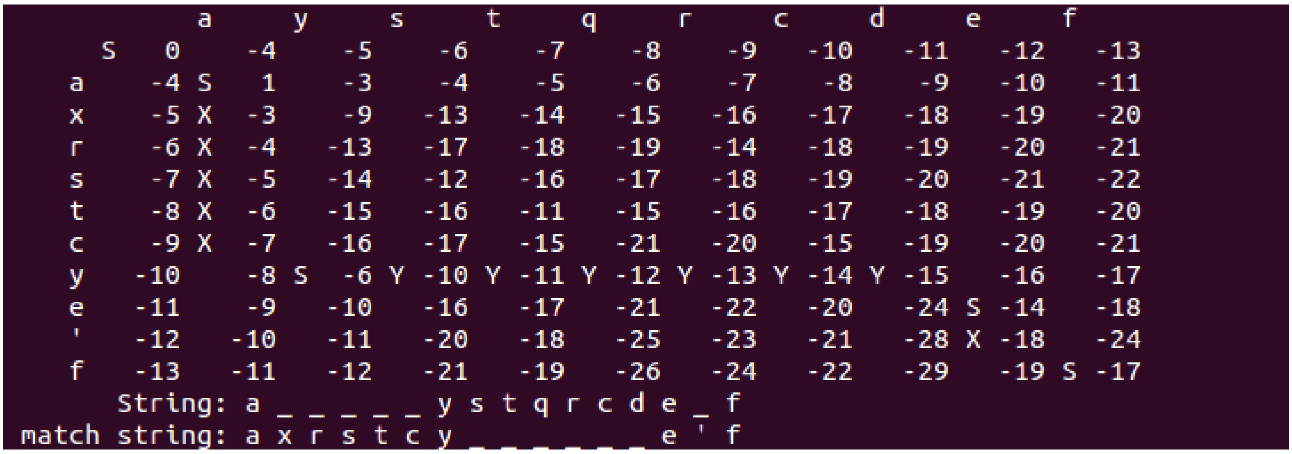
```
            a     y     s     t     q     r     c     d     e     f
    S   0    -4    -5    -6    -7    -8    -9   -10   -11   -12   -13
   a    -4 S   1    -3    -4    -5    -6    -7    -8    -9   -10   -11
   x    -5    -3 S   0    -4    -5    -6    -7    -8    -9   -10   -11
   r    -6    -4    -4 S  -1    -5    -6    -5    -8    -9   -10   -11
   s    -7    -5    -5    -3 S  -2    -6    -7    -6    -9   -10   -11
   t    -8    -6    -6    -6    -2 S  -3    -7    -8    -7   -10   -11
   c    -9    -7    -7    -7    -6    -3 S  -4    -6    -9    -8   -11
   y   -10    -8    -6    -8    -7    -7    -4 S  -5    -7   -10    -9
   e   -11    -9    -9    -7    -8    -8    -8    -5 S  -6    -6   -10
   '   -12   -10   -10   -10    -8    -9    -9    -9    -6 S  -7    -7
   f   -13   -11   -11   -11   -10    -9   -10   -10   -10    -7 S  -6
      String: a y s t q r c d e f
match string: a x r s t c y e ' f
```

Case 2 : Copy cost 10(Only if the characters match) / Insertion Cost -1(X/Y)



```
            a     y     s     t     q     r     c     d     e     f
    S   0    -4    -5    -6    -7    -8    -9   -10   -11   -12   -13
   a    -4 S  10     6     5     4     3     2     1     0    -1    -2
   x    -5 X   6     9     5     4     3     2     1     0    -1    -2
   r    -6     5 S   5     8     4     3    13     9     8     7     6
   s    -7     4     4 S  15    11    10     9    12     8     7     6
   t    -8     3     3    11 S  25 Y  21 Y  20    19    18    17    16
   c    -9     2     2    10    21    24    20 S  30    26    25    24
   y   -10     1    12     9    20    20    23    26 S  29    25    24
   e   -11     0     8    11    19    19    19    25    25 S  39    35
   '   -12    -1     7     7    18    18    18    24    24 X  35    38
   f   -13    -2     6     6    17    17    17    23    23    34 S  45
      String: a _ y s t q r c d e _ f
match string: a x r s t _ _ c y e ' f
```

Case 3: Copy cost 1 (only if characters match) / Insertion Cost -10(X/Y)



```
            a     y     s     t     q     r     c     d     e     f
    S   0    -4    -5    -6    -7    -8    -9   -10   -11   -12   -13
   a    -4 S   1    -3    -4    -5    -6    -7    -8    -9   -10   -11
   x    -5 X  -3    -9   -13   -14   -15   -16   -17   -18   -19   -20
   r    -6 X  -4   -13   -17   -18   -19   -14   -18   -19   -20   -21
   s    -7 X  -5   -14   -12   -16   -17   -18   -19   -20   -21   -22
   t    -8 X  -6   -15   -16   -11   -15   -16   -17   -18   -19   -20
   c    -9 X  -7   -16   -17   -15   -21   -20   -15   -19   -20   -21
   y   -10    -8 S  -6 Y -10 Y -11 Y -12 Y -13 Y -14 Y -15   -16   -17
   e   -11    -9   -10   -16   -17   -21   -22   -20   -24 S -14   -18
   '   -12   -10   -11   -20   -18   -25   -23   -21   -28 X -18   -24
   f   -13   -11   -12   -21   -19   -26   -24   -22   -29   -19 S -17
      String: a _ _ _ _ _ y s t q r c d e _ f
match string: a x r s t c y _ _ _ _ _ _ e ' f
```

Result

In the first case both the penalty of other operations and the gain in copying the same characters are the same and hence the system decides to substitute (one of the other operations) .

However in the second case when the gain in copying increases 10 times , the system tries to perform copying as many times as possible . Since the penalty of performing other operations isn't much , the system does perform other operations just to match the same characters in both the strings

In the third case , the system can perform copying only if both the strings have the same character . For the other cases it will have to perform other operations and there is nothing it can do about it . Hence it decides to perform insertions . Note that this is just optimal path , there are many and the code just prints out one of them

Spell Checker

I also implemented a spell checker by using a word dictionary from the Internet . I apply the smith-waterman algorithm to find the closest match and return that as output . The user has to call the python file with the word as the first argument

Sample Output of this Algorithm

```
python spellchecker.py wprd

ward

python spellchecker.py wordasard

wordage
```