CSCI 404 Assignment #2

Due: Jan 29 midnight (11:59pm), online submission

Submission

1. All files should be submitted to Canvas. Only one of your team members needs to submit on behalf of the team. Indicate clearly in your submission the team members. You can submit multiple times, but please have the same team member resubmit all required files each time.
2. Implementation in Python.
3. Submit the **assn2.zip** file. **For each question**, submit your programs (if any) and a written report (**report.** txt, .doc, or .pdf) that presents the results you obtained, summarizes the methods you implemented if any, any additional resources you used, and discuss the results if necessary.
4. **DO NOT** include the given data files in your submission!!

Question 1: Minimum Edit Distance (40pts)

The following Python code (distance.py) computes the minimum number of edits (you may have to fix the code for this code was written in python 2.5): insertions, deletions, or substitutions that can convert an input source string to an input target string. Using the cost of 1, 1 and 2 for insertion, deletion and replacement is traditionally called Levenshtein distance.

```
def distance(target, source, insertcost, deletecost, replacecost):
    n = len(target)+1
    m = len(source)+1
    # set up dist and initialize values
    dist = [ [0 for j in range(m)] for i in range(n) ]
    for i in range(1,n):
        dist[i][0] = dist[i-1][0] + insertcost
    for j in range(1,m):
        dist[0][j] = dist[0][j-1] + deletecost
    # align source and target strings
    for j in range(1,m):
        for i in range(1,n):
            inscost = insertcost + dist[i-1][j]
            delcost = deletecost + dist[i][j-1]
            if (source[j-1] == target[i-1]): add = 0
            else: add = replacecost
            substcost = add + dist[i-1][j-1]
            dist[i][j] = min(inscost, delcost, substcost)
    # return min edit distance
    return dist[n-1][m-1]

if __name__=="__main__":
    from sys import argv
    if len(argv) > 2:
        print "levenshtein distance =", distance(argv[1], argv[2], 1, 1, 2)
```

Let's assume we save this program to the file `distance.py`, then:

```
$ python distance.py gamble gumbo
levenshtein distance = 5
```

Your task is to produce the following visual display of the best (minimum distance) alignment:

```
$ python view_distance.py gamble gumbo
levenshtein distance = 5
g a m b l e
|   | |
g u m b _ o

$ python view_distance.py "recognize speech" "wreck a nice beach"
levenshtein distance = 14
_ r e c _ _ o g n i z e   s p e e c h
  | | |       | |   | |     |   | |
w r e c k   a   n i c e   _ b e a c h

$ python view_distance.py execution intention
levenshtein distance = 8
_ e x e c u t i o n
      |     | | | |
i n t e _ n t i o n
```

The 1st line of the visual display shows the target word and the 3rd line shows the source word. An insertion in the target word is represented as an underscore in the 3rd line aligned with the inserted letter in the 1st line. Deletion from the source word is represented as an underscore '_' in the 1st line aligned with the corresponding deleted character in the source on the 3rd line. Finally, if a letter is unchanged between target and source then a vertical bar (the pipe symbol '|') is printed aligned with the letter in the 2nd line.

You can produce this visual alignment using two different methods:

- Memorize which of the different options: insert, delete or substitute was taken as the entries in the table are computed; or
- Trace back your steps in the table starting from the final distance score by comparing the scores from the predecessor of each table entry and picking the minimum each time.

There can be many different alignments that have exactly the same minimum edit distance. Therefore, for the above examples producing a visual display with a different alignment but which has the same edit distance is also correct.

2) Print out the valid alignments with the same minimum edit distance. You should print out the first 50 alignments or N alignments, where N comes from a command line argument -n.

Question 2: Language Model and Noisy Channel (20pts)

Read through the slides “Spell Correction” on Canvas, write an essay describing step by step how the spelling correction is perform by using the example of correcting “acress” on the slides. Make sure to show your understanding to the process. Clarity and correctness is required.