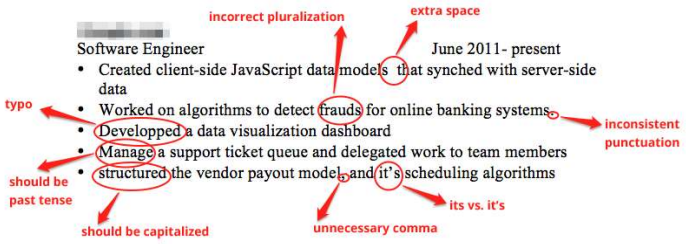
Use of AI tools and downloaded code is PROHIBITED in this assignment

CSCI 4820/5820 (Barbosa)

Project 1: Minimum Edit Distance

Due: See course calendar for due date. May not be turned in late.

Assignment ID: **proj1**
File(s) to be submitted: **proj1.py, myout.log**



Objective(s): 1) Implement minimum edit distance algorithm; 2) enhance the algorithm with backtracing to output the alignment and edit operations; 3) provide required analysis (CSCI 5820 graduate students only).

Project description:

In this assignment you will implement the minimum edit distance algorithm discussed in class. You will write a small **Jupyter notebook program** that constructs the minimum cost table (using dynamic programming), and enhances it with backtracing "pointers" to output the alignment and edit operations.

Requirements:

Grading: Where two weights are shown for an item, the first is for CSCI 4820 and the second for CSCI 5820 credit.

0. All projects in this course will use a **flat directory structure** (the program and input/output files must be at the same level in the directory/folder structure), unless otherwise stated.

1. (**80%/65%**) Your program must provide the functionality listed below, in a single notebook named ***proj1.ipynb***:

- *Input* – The input are sets of words (one set per line in **lowercase**) from a file that must be named ***words.txt***, for which the minimum edit distance is to be calculated. The first word in each line is the **target** word. All other words in the line are **source** words that must be transformed to the target word, using the minimum edit distance algorithm. Two sample input files are provided (both of these are in the same format):
  - ***costs.csv*** – a comma-delimited file containing Levenshtein substitution costs for the lowercase alphabet
  - ***costs2.csv*** – a comma-delimited file containing a confusion matrix edit substitution costs
- *Processing* – The processing requirements include:
  - The cost of insertions and deletions is 1 in all cases. Substitution costs are read from the input files.
  - For each pair of source and target words, calculate the minimum edit distance (using both Levenshtein and confusion matrix costs), and output the cost and backtrace of operations (see below for details).
  - The dynamic programming table must be complete and correct. The backtrace table must capture all possible sources for the minimum cost at each cell.
  - When constructing the backtrace, randomly select any one of the possible cells that provide the minimum cost to the cell being processed. This should be done by importing the *random* module, and ensuring that all possibilities leading to the minimum cost have an equal probability of being selected.
  - Ensure that the submitted code **does not seed** the random number generator (you may seed during development/testing). Your program will be run multiple times so that different alignments are shown.
- *Output* – For each pair of source and target words, the program should display the following output:
  - 4 lines for each of the cost methods (Levenshtein and confusion matrix)
    1. Line 1 will show the **source** word
    2. Line 2 will contain a vertical bar ("|") for each operation (one per character)

Use of AI tools and downloaded code is PROHIBITED in this assignment

3. Line 3 will show the **target** word
4. Line 4 show the operation for each character and the edit cost – The only change from what's in the textbook is that the letter ***k*** (for *keep*), rather than a space, indicates a null substitution.

- A line of 50 hyphens should be used to separate the pair of words from the next pair
- Example output:

```
m i s c h i e * * * * f
| | | | | | | | | | |
m i s c h i e v o u s *
k k k k k k k i i i i d  (5)

m i s c h i e f * * *
| | | | | | | | | | |
m i s c h i e v o u s
k k k k k k k s i i i  (3)
--------------------------------------------------
* * * * * * d e v i o u s
| | | | | | | | | | | | |
m i s c h i * e v * o u s
i i i i i i d k k d k k k  (8)

* d * * e v * i o u s
| | | | | | | | | | |
m i s c h i e v o u s
i s i i s s i s k k k  (4)
--------------------------------------------------
```

2. (**15%**) Test your program – Run it multiple times to ensure that different backtraces are selected. Assess the correctness of each backtrace.

3. (**5%**) Code comments - Add the following comments to your code:

- Place a Markdown cell at the top of the source file(s) with the following identifying information:

  Your Name
  CSCI 4820/5820-001
  Project #X
  Due: mm/dd/yy
- Below your name add comments in each file that gives an overview of the program or class.
- Add a Markdown cell above each code cell explaining the processing carried out by the code in that cell.

4. (**CSCI 5820 students only 15 %**) Analysis – Provide an answer/analysis (in a Markdown cell) for these questions:

a) Compare and contrast the results obtained from using the different cost approaches. Is one "better" than the other? How? Why?
b) While the algorithm you implemented yields the minimum edit distance between a pair of words, it is not clear how it fits into a larger context (e.g., where does it get the words from?). Provide a description of plausible uses in the context of other natural language applications (example source/target sequences, how the best alignment is selected based on costs, handling cases where the correct alignment does not have the minimum cost, etc.)
c) Explain how you might devise a new set of substitution costs: what process would you go through? What data would you use or collect? How would you arrive at final values for the table?

5. Generate a pdf file of the notebook (from the terminal):
   $ *jupyter nbconvert --to html proj1.ipynb*
   $ *wkhtmltopdf proj1.html proj1.pdf*

6. Submit required files via the D2L dropbox for this project