Ling 473 Project 6 (Extra Credit)
Due 11:45pm on Thursday, September 6, 2011

This project is optional, for students who need additional programming practice, or who missed a project or wish to improve their grade in the class. There are two parts described in this document, (a.) and (b.). Part (b.) builds on the previous part, so you can choose to do either only part (a.), or both parts.

For your last project you will implement a program to calculate edit distance, and then use your function to report the difference between two short texts. We will use a variation on the Levenshtien method which is explained in the Jurafsky and Martin textbook. You will measure the edit distance between two revisions of the Wikipedia article on British Petroleum plc, before and after the Gulf Oil Spill.

Abstractly, edit distance treats each input as a sequence of elements. In this project, each text will be treated as a sequence of lines. But furthermore, as we do this calculation, the cost of substituting one line for another will be based on the edit distance between those two lines—this time treated as a sequence of characters.

a.) Calculate the numeric (floating-point) edit distance between the two texts, according to the parameters given below.

b.) Print the best alignment between the two texts.

Parameters

Edit-distance is a problem best solved with dynamic programming. Set up your code with a rectangular array, according to the Levenshtein edit distance example in section 3.11 of Jurafsky and Martin. Walking through the grid to find the edit distance is explained in the lecture.

Implement a normalized edit distance function

$$distance = G(source_{1\ldots m}, target_{1\ldots n}, f).$$

The m-row by n-column distance matrix will contain floating point values rather than integers. The insertion and deletion costs are both 1.0, and the substitution cost (using subscripts as in J&M Figure 3.25) will be

$$f(source_{j-1}, target_{i-1}).$$

The particular substitution cost function f varies, as will be discussed below. The edit distance d returned by your function will be normalized to 1.0 by dividing $distance[n, m]$ by $m + n$, unless the lengths of the source and target are zero, in which case the normalized edit distance is zero.

a.) Return the floating point edit distance between the two *texts*, treating each *line* (string) as a single element. As mentioned, the **insertion** and **deletion** costs are always 1.0.
   i. For this outer calculation, the **substitution cost function** f is defined as:
      - 0.0, if the two *lines* (strings) are identical, or
      - (0.5 + the floating point edit distance between the two lines), otherwise.

ii. When computing the edit distance between two *lines* (strings) in step (i), t**he substitution cost function** f will have the value 0.0 if the *characters* are the same, and 2.0 if the *characters* are different.

In other words, you will have nested calls to your edit distance function G. Overall, you are finding the distance between two *texts*, and each *line* is an element. The cost function used during this inner calculation requires calculating the edit distance between two *lines*, and for this, each *character* is an element.

b.) Add to your code from part (a.) to implement backtrace and report a best alignment.

As explained in J&M, when backtracing, you sometimes have up to three backtrace cells to choose from. For this project, always choose the backtrace cell with the *lowest* floating-point distance value. Because of this, you don't actually have to store the directional arrows. Just start from the ending cell, and follow the path of the backtrace cell (of the three) that has the lowest value.

Coding Recommendations

It is best to start with the basics. Implement the basic Levenshtein edit distance function for strings with insertion, deletion, and substitution costs { 1, 1, 2 } respectively, and no normalization. Using the words *intention* and *execution*. make sure you can repeat the results of the Jurafsky and Martin example and calculate an edit distance of 8. This should not take too long.

Be very careful about indexing, which is not clearly used in J&M. If your **arrays** are zero-based, then row [0, *j*] and column [*i*, 0] will contain distances from the empty string. If your **sequences** are also zero-based, then cell [1,1] of your array will correspond to the first element—at index zero—in the source and target sequences. Thus, when you call the substitution cost function

$$dist[i-1, j-1] + f(source_{j-1}, target_{i-1}),$$

—which is correct for this type of setup—you are referencing the row and column for ***previous*** elements in the matrix $dist$ and the cost function for ***current*** elements.

If you cannot get your implementation of the J&M pseudo-code to work, please do not continue on; instead just focus on getting this part working for your submission, and explain the problems you ran into in your write-up.

When this is working, test your code with strings of different lengths, to make sure that you didn't mix up the row and column indexes. The distance between *abcde* and *bd* should be 3. Only when this is working should you generalize the function to handle:

- distance between whole texts, of strings (in addition to distance between strings, of characters);
- normalization (the normalized distance between *abcde* and *bd* should be 0.4286); and
- the adjustable substitution cost function.

Reminder: in your edit distance function, when substituting strings, the substitution cost is either 0.0 (if the strings are identical) or (0.5 + *the edit distance between the strings*), and when substituting characters, the substitution cost is either 0.0 (if the characters are identical) or 2.0 (if they are different).

Parts (a.) and (b.) both require calculating the edit distance between lines of text (strings) as well as between characters. For weakly-typed languages such as Python, it is not a problem to call the same function with heterogeneous types at different times. If you are using a strongly-typed language such as Java or C#, the easiest approach would be to have two separate edit distance functions. This would also let you hard-code the substitution cost function for each one. A more elegant approach using function or class templates was shown in the lecture.

When sharing the same code between the string-based and character-based edit distance functions, it is also natural to use a lambda function to allow the caller to specify the substitution cost function f. There will be no deduction for the less elegant approaches, but I do encourage you to explore the advanced facilities of your programming language if you can.

Input

In the following location you will find two text files containing the first few lines of the Wikipedia article on BP. One is from mid-April and the other is from late-August 2010.

`/opt/dropbox/11-12/473/project6/`

For testing purposes, there is also a pair of Lady Gaga comparison files which should give an edit distance of 0.4576.

Output

a.) Using your edit distance function, calculate a single floating point number, which is the edit

```
                                    Lady GaGa 0.00 Lady GaGa
Joanne Stefani Germanotta (born March 20, 1986), 0.32 Stefani Joanne Angelina Germanotta (born March
         best known by her stage name Lady GaGa, 0.19 28, 1986), known by her stage name Lady Gaga,
    is an Italian-American singer-songwriter and 0.40 is an American recording artist. She began
                                                1.00 performing in the rock music scene of New York
                                                1.00 City's Lower East Side in 2003 and enrolled at
                                                1.00 New York University's Tisch School of the
        dance-pop/electronica musician signed to 0.67 Arts. She soon signed with Streamline Records,
                             Interscope Records. 0.36 an imprint of Interscope Records. During
                                                1.00 her early time at Interscope, she worked as a
                                                1.00 songwriter for fellow label artists and
                                                1.00 captured the attention of Akon, who recognized
                                                1.00 her vocal abilities, and signed her to
                                                1.00 his own label, Kon Live Distribution.
                                                0.00
  Lady Gaga at Sommarkrysset, Stockholm, Sweden. 0.70 Gaga performing at The Monster Ball Tour in April 2010
                          Background information 0.00 Background information
              Birth name Stefani Germanotta (lol) 0.28 Birth name Stefani Joanne Angelina Germanotta
       Born March 20, 1986 (1986-03-20) (age 24) 0.05 Born March 28, 1986 (1986-03-28) (age 24)
                                                1.00 New York City, New York,
                                                1.00 United States[1]
         Origin Yonkers, New York, United States 0.65 Genres Pop, electronic, dance
                  Occupations Singer, songwriter 0.05 Occupations Singer-songwriter
                                                1.00 Instruments Vocals, piano, synthesizer, keytar
                     Years active 2007 - present 0.08 Years active 2006-present
        Labels Streamline, Kon Live, Interscope 0.21 Labels Def Jam, Streamline, Kon Live, Cherrytree, Interscope
                       Website www.ladygaga.com 0.00 Website www.ladygaga.com
                                                0.00
                                 Contents [hide] 0.00 Contents [hide]
                                     1 Biography 0.71 1 Life and career
                                   1.1 Early life 0.28 1.1 1986-2004: Early life
                                                1.00 1.2 2005-07: Career beginnings
                                1.2 Music career 0.71 1.3 2008-present: The Fame and The Fame Monster
                             2 Critical reception 0.60 2 Musical style and influences
                  3 TV appearances and performances 0.76 2.1 Public image
                                   4 Discography 0.08 3 Discography
                                4.1 Studio albums 0.58 4 Tours
                                   5 Music Videos 0.63 5 Awards and nominations
                         6 Notable writing credits 0.57 6 References
                                     7 References 0.59 7 Further reading
                                 8 External links 0.00 8 External links
```

distance between the two British Petroleum text files, according to the instructions above. Include the result in your readme file write-up.

b.) For this part, you can output your alignment as shown above, or in a similar format. Here, lines from the two files are aligned with each other on the left and right, and the edit distance between the two lines is shown in the center. Note that this sample shows the correct output for the Lady Gaga files, where the overall edit distance is 0.45760480772406.

Submission

Include the following files in your submission. For part (a.), you do not have to provide an output file. Instead, just include the edit distance value that your program calculates for the British Petroleum file pair in your readme file.

`compile.sh`	Contains command(s) that compile your program. If you are using python, shell scripts, or any other interpreted language that does not require compiling, then this file will be empty, or contain just the single line: `#!/bin/sh`
`run.sh`	The command(s) that run your program. Be sure to include compiled binaries in your submission so that this script will execute without first running compile.sh
`part-b-output`	The output from part (b.)
`readme.{pdf, txt}`	Your write-up of the project. Describe your approach, any problems or special features, or anything else you'd like me to review. If you could not complete some or all of the project's goals, please explain what you were able to complete.
`(source code and binary files)`	All source code and binary files (jar, a.out, etc., if any) required to run and compile your program

Gather together all the required files, making sure that, for example, any PDF or other binary files are transferred from your local machine using a binary transmission format. Then, from within the directory containing your files, issue the following command to package your files for submission. Replace the bracketed portion with your UW NetID.

```
tar -czf [your-uw-netid].tar.gz .
```

Notice that this command packages all files in the current directory; do not include any top-level directories. Upload the file to CollectIt.