Report - Homework 7

How system works and justification on Algorithm used:

The program starts by reading the files given as input in a line-by-line manner. For each line read, we separate out the information such that the line number at the beginning of the string is extracted separately and the string after “---” is extracted separately. For this I make use of substring property of a string by specifying the IndexOf the substring to extract.

Now, the algorithm proceeds and checks for each string (each line leaving out the initial line number) in file1 with all strings in file2 and calculates similarity score for each of the case. I do it for 3 algorithms (i.e Needleman-Wunch, Smith-Waterman and Jaro-Winkler). Then the max similarity score for the string is calculated. If the max score is greater than 0.88 then the match is considered and displayed in outfile.

Looking at each line in the input files, the first thing to notice is that many strings which match have the common prefix, so a bonus score has to be given. Also, better score has to be given for having characters in common if they are “close by”. So, **Jaro-Winkler** is used.

Ex: arnie morton's of chicago 435 s. la cienega blv. los angeles American

arnie morton's of chicago 435 s. la cienega blvd. los angeles steakhouses

In the example like above, Jaro-Winkler works very well because of the prefix match. Hence for such matches I make use of Jaro-Winkler.

For some other strings **Smith-Waterman** works well, because there are many subsequences which align well. Since this algorithm works on local-alignment to find best subsequences to align.

Ex: locanda veneta 8638 w. third st. los angeles italian

locanda veneta 3rd st. los angeles Italian

Here we see that the subsequences are matched inorder to get a good similarity score by Smith-Waterman algorithm.

For some strings **Needleman-Wunch** works well. Since it works on global alignment to fully align sequences, openings gaps as needed.

Ex: doidge's 2217 union st. san francisco American

perry's 1944 union st. san francisco American

In examples like above, this algorithm gives good scores when the characters match (score of 2), penalizes for mismatch of characters (score of -1) and lower penalty for gap mismatch (score of -0.5)

So, since we have different cases where different algorithms work well, the best approach was to use a combination of all 3 algorithms and get the best match.

Additional Jar file used: Simmetrics jar file for java, downloaded from

http://sourceforge.net/projects/simmetrics