**Capstone Project – Digital Humanities**

INTRODUCTION

Over the years students have found various ways of transforming text from the original authors into something that they could possibly use in their research. Many times it becomes very hard for a student to transform this published text into a usable, trustworthy solution that they might use in a formal paper. It is often the challenge of the student to know which publication is the original transcript of the text. Calculating the number of modifications between similar text is one way of determining sequence analysis for identification of the most likely original text. This comparison of texts is known as string editing metrics.

The technique of computing the distances between texts is very complicated and can be achieved using many different methods. The testing base mark for this Fall, 2015, Capstone Project will use Levenshtein’s Distance method. This distance algorithm will be put in place to determine a distance measurement between two text inputs. This text input is to be convert to a string during the input process. The coding for this method has been done through the input of group members Greg Dawkins, Marlene Williams, and Kevin Woods.

Once the Levenstein Distance method is analyzed by the group the input string will then be analyzed by three different distance algorithm methods. The methods that have been chosen are Jaro-Winkler method, Needleman-Wunsch method, and the Hunt-McIlroy method.

LEVENSHTEIN’S DISTANCE METHOD

Levenschteins distance algorithm is a method that was developed by Vladimir Levenshtein in 1965. In the *Dictionary of Algorithms and Data Structures,* Levenshtein’s method is described as an algorithm that uses the smallest number of insertions, deletions, and substitutions that would be needed to change one string into a similar string sequence. He published this as a tool to measure the distance between two sequences. This tool is still used widely today in various computational sequences like DNA analysis, spell checker software, or even fuzzy string matching.

JARO-WINKLER ALGORITHM

NEEDLEMAN-WUNSCH ALGORITHM

Another string comparison method was developed by Saul B Needleman and Christian D Wunsch around 1970. This method is long been used in biometrics to find the best alignment of string sequences. In the article written by Authur M. Lesk for Encyclopeadia Britannica, he describes the Needleman-Wunsch as an algorithm that divides the larger string into smaller subsets which in turn constructs the solution to larger problem.

Their algorithm is implemented by creating a grid based on the length of the strings. This grid of strings is then sequenced through each letter or symbol while making comparisons of the ASCII representation. These computations are the identified in this grid as match or mismatch. Also, there is a function to allow identification of gaps in sequences known as a gap penalty. Finally, after the grid is computed, the alignment is traced back through the grid by way of recursion for the max separation distance of the set of strings.

These algorithms are widely used for DNA sequencing and even voice/face recognition. The complexities of the algorithm on small scale strings make this algorithm one of the widely used in the industry for many of these type sequencing analysis tasks. However, because of the grid allotment of the string comparisons this algorithm is not optimal for larger data sets. This would not be suitable when making larger string comparisons.

The Levenstein’s Distance method and the Needleman-Wunsch are similar in several ways. Both traverse through the string to identify like characters. There are a few dissimilarities. One major difference is the global and local alignment sequences. Levenshtein’s is more efficient but lacks the reliability of the Needleman-Wunsch when measuring smaller data sets. However, on the larger strings, the efficiency of Levenstein’s Distance method could play a much greater role in determining these larger strings.

HUNT-MCILROY

The Hunt-McIlroy algorithm was discovered by J. W. Hunt from the Department of Electrical Engineering at Stanford University in Stanford, California and M. D. McIlroy from Bell Laboratories in Murray Hill, New Jersey. The algorithm is also called the diff program. Hunt and McIlroy published their findings in the document, An Algorithm for Differential File Comparison, which was first published in Bell Laboratories Computing Science Technical Report #41 in 1976. Diff is utilized as the central procedure and used to solve the longest common subsequence problem. The main difference between the Hunt-McIlroy algorithm and Levenshtein Distance is that Hunt-McIlroy determines the string subsequences in which two files are similar. It uses that information and attempts to construct a new file using the similarities. However, Levenshtein Distance returns the amount of changes between the two files. Another difference is that Hunt-McIlroy analyzes the files line-by-line while Levenshtein compares the entire file.

To use the algorithm, one must first read in the files line-by-line. The *i*th line of file one would then be compared to the *j*th line of file two. An integer will be returned which tells how many characters are common in the two lines, *Pij*. This integer represents the longest common subsequence between the two lines of the two files. For large files, meaning thousands of lines, one would create a hash value. For instance, one possible hash value might be to remove all white spaces within each linein order to create one long computer word, which would then be compared. For our purposes, the files were read-in as one line since the poem used was four lines long.

After the comparison is made, the two files would merge into a third file using only the longest common subsequences within the two files. The Hunt-McIlroy algorithm has been used to determine plagiarism, as well as create patches for older programs. While the algorithm works well for these purposes, Levenshtein’s Distance is best for the purpose of determining the order in which files have been created.

CONCLUSION

In conclusion, the Levenstein’s Distance method may be an optimal choice for larger data. It would be determined by the size of the data, storage of the string, and capacity of the hardware.

REFERENCES

Algorithms and Theory of Computation Handbook, CRC Press LLC, 1999, "Levenshtein distance", in [Dictionary of Algorithms and Data Structures](http://www.nist.gov/dads/) [online], Vreda Pieterse and Paul E. Black, eds. 22 June 2015. (accessed TODAY) Available from: <http://www.nist.gov/dads/HTML/Levenshtein.html>

"bioinformatics". Encyclopædia Britannica. Encyclopædia Britannica Online.  
Encyclopædia Britannica Inc., 2015. Web. 08 Nov. 2015  
<<http://www.britannica.com/science/bioinformatics>>.

Hunt, J. W., McIlroy, M. D. (1976) *An Algorithm for Differential File Comparison*, retrieved from http://www.cs.dartmouth.edu/~doug/diff.pdf