**Capstone Research Project**

**Fall 2015**

**Introduction:**

There are several techniques able to take two texts and calculate the number of modifications required to transform the first to the second [1, 2, 3]. These techniques are supported by string editing metrics, which are used in computational and mathematical sciences. These metrics measure the similarities or dissimilarities (distance) between two texts. One well-known area where string editing metrics is used is DNA sequencing analysis. It is also used in areas of fraud detection, fingerprint analysis, plagiarism detection, ontology merging, image analysis, evidence based machine learning, database data duplication, and data mining. This research examines some of those different methods and how they can be evaluated in terms of effectiveness and performance. As effectiveness, the methods must be analyzed to verify that they are able to measure the differences or similarities between two texts and such measurement could be applied to estimate an evolutionary sequence (possible sequence of changes applied to an original text). With respect to performance, the methods need to be implemented and their processing time measured in order to compare their computational time. A theoretical analysis of their time complexity may accompany their performance results.

**Source for different editing metrics links:**

<https://en.m.wikipedia.org/wiki/Edit_distance>

<https://en.m.wikipedia.org/wiki/Levenshtein_distance>

**Sample texts to be used in tests:**

From <http://www.goodreads.com/quotes/show/44777> designated poem 1:

“Faith is a fine invention

When gentlemen can see,

But microscopes are prudent

In an emergency.”

From http://www.online-literature.com/dickinson/poems-series-2/32/ designated poem 2:

Faith is a fine invention

For gentlemen who see;

But microscopes are prudent

In an emergency!

From <http://www.emilydickinsonmuseum.org/church> designated poem 3:

"Faith" is a fine invention

For Gentlemen who see!

But Microscopes are prudent

In an Emergency!

**Project:**

Each student will select two editing metric methods from the sources provided or any other source, where one of them must be the Levenshtein method. Since you are all working as a team you may discuss which additional method to Levenshtein each of you will work with.

Each student will then code an implementation of the algorithms and execute them in order to get results in the comparison of the three sample texts and their computational time. The time must be measured as the average of multiple runs (at least three) done on a computer without any other significant activity in order to eliminate possible interference from the execution environment. A group report must be submitted with the following sections:

1. Introduction (motivation behind the study and methods chosen and discarded)
2. Editing metric methods (description of the target methods, what they measure)
3. Complexity analysis (for all three algorithms according to the group’s implementation)
4. Test results and comparisons (time comparison and evolution estimation based on the numerical results, assuming small differences or larger similarities as an indication of versioning)
5. Conclusion (which method would be recommended for a more general application and a reason for such recommendation)

**Proposed timeline:**

Sprint 1 – Fully grasp the project, select methods and fully understand them (steps 1 & 2)

Sprint 2 – Have all methods implemented and results from test runs (step 3)

Sprint 3 – Complete the comparison of the three methods and include the comparisons with Levenshtein’s algorithm. (steps 4 & 5)

**Goal**:

Based on the report that will be submitted we will be trying to publish a research in a peer reviewed conference.

In the case that the submission/publication takes place after student graduation you will be informed of the fact so that you are able to add the publication to your resume.

**References:**

[1]        Hamming, Richard W. (1950), “Error detecting and error correcting codes”, *Bell System Technical Journal* 29 (2): 147–160.

[2]        Levenshtein, V. I. (1966). Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 10(8):707–710.

[3]        Woon, W.L., Wong, K.D. “String alignment for automated document versioning,” Knowledge Information Systems, 2009, pp. 293-309.