Project Group 28

Project proposal

Problem Description.

The project requires us to implement a plagiarism detector. Using the pattern-matching algorithms KMP, LCSS, and Rabin-Karp, we will design a plagiarism detector that has the best possible time complexity. Given a set of documents as well as an original document as an input, we want to determine whether or not there are some similar patterns to the original in the set of documents. Finding similar subsequences would result in plagiarism found.
Another interesting part of the problem is determining which pattern-matching algorithm would result in the best possible time complexity. Our analysis would consider both worst and average time complexity in order to determine the most suitable algorithm.
Lastly, it is important to define what plagiarism is defined as and what the threshold of resemblance will be set at. Plagiarism will be defined as when a document has a matching percentage (not including matched subsequences that are cited and documented) higher than the defined threshold. We will set a tentative threshold percentage at 7.5% of resemblance for now.

Edge Cases.

Partial matching - if a sentence has been copied but with small changes
Detecting quotations
Differences that should be ignored - capitalization, whitespace, punctuation, ect.

Expected complexities.

(time and space complexity is probably what she's expecting here unless she wants to talk about what difficulties we will encounter)
KMP:

- Time complexity: O(m) for failure function, O(m+n) with m being size of the document to search for and n size of the subsequent pattern.
- Space complexity: O(m)

LCSS:

- Time complexity: O(n*m)
- Space complexity: O(n*m)

Rabin-Karp:

- Time complexity: O(n*m)
- Space complexity: O(1)

Dataset Collection.

We will find a set of articles and convert them all to plain text. We will then set aside a portion of these articles to be the test articles and for some of these we will insert lines from other articles in the collection to act as plagiarism.

Programming Language.

We are going to use Python and a Python Notebook as our programming language as it facilitates data visualization and plotting.

Task Separation and Responsibilities.
We need to divide every task equally, meaning everybody needs to do some part of the design & analysis.

Our group consists of 4 people, as per the TA and professor's approval.

Everyone Involved:
Choice of data structure
Design
Implementations
Analysis
Pseudocode
Test Cases

Will Garbutt:
Proofs of correctness

Evan Jager:
Proofs of correctness

Stu McGorman:
Video

Alrick Vincent:
Video