CS307P-SYSTEM PRACTICUM CPYNOT.

B13107 – Amit Kumar
B13141 – Vinod Kumar
B13218 – Paawan Mukker

What is CPYNOT ?

- It’s a plagiarism checker, designed to avoid copying of written text assignments for students studying in universities.

- **Plagiarism detection** is the process of locating instances of plagiarism within a work or document.*

Type -

- Documents’ Plagiarism checker –
  - Match two documents and give a measure of their closeness.
  - Use of algorithms such as Edit distance/String matching .

Functionality -

- Proposed plan –
  - Type -> Documents' Plagiarism checker
  - Take two documents and generate a measure of similarity.
  - Generate these measures for every pairs of documents and conclude the verdict for similar copies
  - Linking it to Moodle/Other Online File repositories.
  - Additional Feature - Application of different heuristics based on the type of document (e.g. an essay or program code in a specific language).

Comparison -

- A two tier comparison of documents –
  - 1) Generate dissimilarity measures by – Document distance approach.
  - 2) Generate dissimilarity measures by – comparing the strings/subsequences (Edit distance/Longest common subsequence).

Document distance -

- Document similarities are measured based on the content overlap between documents.

- Document as -
  - **Word** = sequence of alphanumeric characters
  - **Document** = sequence of words
    - – Ignore punctuation & formatting

Document distance -

- Distance ??
- Idea: focus on shared words
  - **Word frequencies:**
  – D(w) = number of occurrences of word w in doc D.

- Treat each document as a vector of its words
  – One coordinate for every possible word w.

Document distance -

- Similarity measure – Dot product between vectors.

$$D_1 \circ D_2 \equiv \sum_w D_1(w) \cdot D_2(w)$$

Document distance -

- Normalize by magnitude -

$$\frac{D_1 \circ D_2}{||D_1|| \cdot ||D_2||}$$

Compute angle between them-

$$\theta(D_1, D_2) = \text{acos}\left(\frac{D_1 \circ D_2}{||D_1|| \cdot ||D_2||}\right)$$

Problem with Document distance

- Does not takes into consideration the order of occurrence of words.

- Solution –
  - Weighted score with edit distance.
  - And Longest Common Subsequence

Edit distance

- Given two strings str1 and str2 and below operations that can performed on str1. Find minimum number of edits (operations) required to convert ‘str1′ into ‘str2′.
  - Insert
  - Remove
  - Replace
- Dynamic programming

Longest Common Subsequence

- The longest common subsequence (LCS) problem is the problem of finding the longest subsequence common to two sequences.

- Subsequences are not required to occupy consecutive positions within the original sequences.

Dynamic programming -

- The Edit-Distance, Longest Common Subsequence algorithm is implemented using dynamic programming paradigm with time complexity of O(n^2) and space complexity of O(n^2) as well.

Modification in Edit-Distance and LCS –

- Treating the document in P paragraphs each of size 'N' and then applying LCS between every two paragraphs of the two submissions.
- This ensures to catch the plagiarism even if the relative ordering of two paragraphs are changed in the two submissions to trick/by-pass plagiarism checker.
- A final score is generated as the sum of scores by application of LCS paragraph wise.

Automated Script

- The user finally gets to run an Automated Script and gives the directory as input where all files(.txt) are present
- The output(all scores) is generated in log files which is separate for each student.
- Final verdict is given based upon whether the final score is greater than the set threshold.

Thank You