Plagiarism Detector Requirements

Document prepared by:
Manjunath Mattam, Praveen Garimella and Siva Sankar
MSIT Division, International Institute of Information Technology, Hyderabad.

Introduction

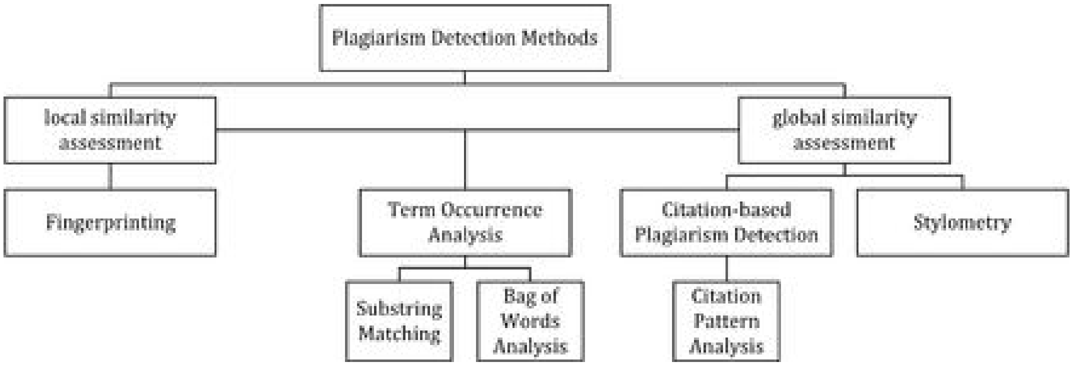
MSIT first year Computer Science Programming and Principles 1 - CSPP1 is a 4 week (4 credit) course work offered after computational thinking course. This course has 3 weeks of IT modules and 1 week of project work. This document has details of week 4 CSSP 1 project. Objective is to create a large enough problem for students to solve such that they can implement / assimilate concepts that were learned in the course. This year students implement basic (beginners) version of plagiarism detector.

Wikipedia Says: Plagiarism detection is the process of locating instances of plagiarism[1] within a work or document. The widespread use of computers and the advent of the Internet has made it easier to plagiarize the work of others. Most cases of plagiarism are found in academia, where documents are typically essays or reports. However, plagiarism can be found in virtually any field, including novels, scientific papers, art designs, and source code.

Detection of plagiarism can be either manual or software-assisted. Manual detection requires substantial effort and excellent memory, and is impractical in cases where too many documents must be compared, or original documents are not available for comparison. Software-assisted detection allows vast collections of documents to be compared to each other, making successful detection much more likely.

Scope of this project is limited to text documents, and focuses on local similarities[2] including fingerprinting, term occurrence analysis. *Global similarity[2] assessment is excluded in this project we do not worry about citations or stylometry.*

1. Plagiarism: According to Wikipedia, plagiarism is the "wrongful appropriation" and "stealing and publication" of another author's "language, thoughts, ideas, or expressions" and the representation of them as one's own original work.
2. The approaches are characterized by the type of similarity assessment they undertake: global or local. Global similarity assessment approaches use the characteristics taken from larger parts of the text or the document as a whole to compute similarity, while local methods only examine pre-selected text segments as input.


Plagiarism Detection Methods
local similarity assessment
global similarity assessment
Fingerprinting
Term Occurrence Analysis
Citation-based Plagiarism Detection
Stylometry
Substring Matching
Bag of Words Analysis
Citation Pattern Analysis

Implementation

In this project input is list (or collection) of text documents that are stored in a directory. Students will need to implement following aspects:

1. Given a document, finding out if there are any similarities with other documents in the directory.
2. List out all the documents with similarities more than *x* percentage. Here *x* is the threshold decided by the user representing % of match among available text documents.

This project is subdivided into following methodologies:

Bag of Words:

In this method you have to compute a distance (an angle) between two given documents or between two strings using the cosine similarity metric. You start with reading in the text and counting frequency of each word. The word frequency distribution for the text **D** is Java's Map from words to their frequency counts, which we'll denote as **freq(D)** . We view **freq(D)** as a vector of nonnegative integers in N-dimensional space. For example, reading the string "To be or not to be" results in the following map
{be=2, not=1, or=1, to=2}

These 4 distinct words make a document representation as a 4-dimensional vector **{2, 1, 1, 2}** in term space.

A word is a sequence of letters [a..zA..Z] that might include digits [0..9] and the underscore character. All delimiters are thrown away and not kept as part of the word. Here are examples of words:
abcd
abcd12
abc_
a12cd
15111

We'll treat all upper-case letters as if they are lower-case, so that "CMU" and "cmu" are the same word.
The **Euclidean norm** of the frequency vector is defined by
||x|| = sqrt(x1^2+x2^2+....+xn^2)
where **x_k** denote frequencies for each word in the text. For the above example, the norm is
||x|| = sqrt(2^2+1^2+1^2+2^2) = 3.16228
The **Dot product** (or the inner product) of two frequency vectors X and Y is defined by
X = {x1,x2,x3,...,xn}; Y = {y1,y2,y3,...,yn}
X . Y = x1.y1+x2.y2+...+xn.yn
Here we multiply frequencies **x_k** and **y_k** of the same word in both text documents.
Finally, we define a distance between two documents D_1 and D_2 by using cosine similarity measurement:

Observe, the distance between two identical documents is 0, and the distance is Pi/2 = 1.57... if two documents have no common words.

dist(D1,D2) = arccos(freq(D1) . freq(D2) / ||freq(D1)|| * ||freq(D2)||)

Please read from the following URL.
https://www.andrew.cmu.edu/course/15-121/labs/HW-4%20Document%20Distance/lab.html

Expected output: If there is plagiarism, most words that are used in text will match. There will be significant similarities with frequencies as well.

Example:
File - 1 Content: To be or not to be
File - 2 Content: Doubt truth to be a liar

The words from both the files are as follows.

Note: Freq(X) is : Frequency of words from file x

Words	Freq(file 1)	Freq(file 2)	Dot product of (Freq(File1) and Freq(File 2))

To	2	1	2
Be	2	1	2
Or	1	0	0
Not	1	0	0
Doubt	0	1	0
Truth	0	1	0
A	0	1	0
Liar	0	1	0

The Euclidean norm of the frequency vector is as follows:

|| File 1 || is: 10
|| File 2 || is: 6

So, the similarity between the documents are

Similarity Function = cos θ = Dot Product(Frequency of File 1, Frequency of File 2) / ||File 1|| * || File 2||

So, you will get, cos θ = 4 / 60 = 0.066 which is 6.6% matching

Limitations: This method is inefficient for long documents, because order in which these terms appear is lost. Given any topic there are certain words that are bound of appear which make this method informal. This method results in both false positive and false negative which is not representing actual copied document and not identifying a copied document. This method will not identify cases where central idea is copied with use of different words.

String matching:

In this method every text document is checked for common sub strings with the verifying document. Although this method is computationally expensive finding longest common substring will tell us what % or how many sentences are copied from the verifying document. This method relies on patterns, and text overlaps.

Example:
File - 1 Content: what is your name
File - 2 Content: my name is xyz

In Computer Science, brute-force search or exhaustive search, also known as generate and test, is a very general problem-solving technique that consists of systematically enumerating all possible candidates for the solution and checking whether each candidate satisfies the problem's statement.

Step1: **w**hat is your name
my name is xyz
w and m does not match so we shift the string to next position(word), until we find a match.

Step2: what **is** your name
my name **is** xyz
Now, we can see is and is matches. So, LCS = 2 and we continue the iteration.
.
.
StepN: what is your **name**
my **name** is xyz
Now, LCS changes to 4
Length of LCS is 4
Total length of file1 and file2 is 31
%match = ((4 * 2) / 31) * 100 = 25%

Expected output: If there is plagiarism, longest common substring will have large portions of verifying documents. It is possible that we may find other sentences as well that were copied.
Limitations: This method is computationally expensive in terms of time and space. Most string matching problems use effective algorithms and data structures that were not part of CSPP1 as a result you will end up using brute force (naive) method which is costly and ineffective.

Fingerprinting:

In this method every text document is checked for signatures with verifying document. Signature is set of words in contiguous sequence (set of words written together) equivalent to sentences in a paragraph (document). Students are expected to compute all possible signatures i.e set of words in sequence upto end of sentence or upto non space special character like , : etc. or set of words within quotations. Signatures can exclude stop words like a, an, the, by, on, to etc. which are common occurring in any 2 documents. After computing signatures of text document and verifying document this method needs to search for common signatures that are approximately matching that is finding signatures that are approximately matching rather than exactly matching.

Expected Output: If there is plagiarism, there will be common signatures (approximate matching) in both documents. Implementers have liberty to choose which signatures are approximate matching (and which are not) based on order of words in a given signature.

Limitations: Signatures may vary even if there are 2 exact same documents,like if punctuations or alignments are changed. Approximate matching is computationally expensive and implementer specific so it is hard to find exact signature matches. Generally fingerprinting problems use effective algorithms and data structures that were not part of CSPP1 as a result you will end up using brute force method which is costly and ineffective.

Putting it all together - Results / Plagiarism verdict

Each of the methods that were used in the plagiarism detector will provide approximate / possible % of similarities. Based on these results a final plagiarism verdict (or conclusion) has to be presented for given text documents. Students are free to assign metric (weight) for each of the methods and create f(x) to assert plagiarism (boolean - has occurred or not). Remember this project not only computes plagiarism detection between 2 documents, but also to find all copies from list of documents in a directory. Therefore a log file is needed to store evidences if f(x) asserts plagiarism.