Code for LCSS

```
package LCSS;

import java.io.File;
import java.util.Scanner;
import java.io.BufferedReader;
import java.io.FileReader;

public class LCSS_main {
        static String[][] docs (String location, int number_of_files_to_use_in_corpus){
                String[][] dummy = new String[0][0];
                try {
                        String string_location = location;

                        File file_locations = new File(string_location);
                        String[] directory = file_locations.list(); //this is an array that holds all of the
names for all of the csv  files
                        File[] files = new File[directory.length];
                        String[][] strings = new String[directory.length][]; //one string for every
comment

                        Scanner reader1 = new Scanner("");
                        String t1;
                        BufferedReader in;
                        for(int i = 0; i< number_of_files_to_use_in_corpus; i++) { //directory.length
this initializes string[][], string[i] holds files, string[][j] holds individual comments
                                                                                                    //I used this
format for possible expansion in case there where too many comments for a computeres
memory the job could be split by strings[i]
                                String string_file_location = string_location + "/" + "/NiceFiles_" +
Integer.toString(i) + ".txt";;
                                files[i] = new File(string_file_location);
                                in = new BufferedReader(new FileReader(files[i]));

                                int file_comment_count = 0;
                                int endl = 0;
                                char tc = ' ';
                                int ti = 0;
                                t1 = "";

                                while(ti != -1) {
                                        ti = in.read();
                                        tc = (char) ti;
```

```
                    if(Character.isAlphabetic(tc)) {
                        t1 = t1 + Character.toLowerCase(tc);;
                    }
                    else if(tc == ';') {
                        endl ++;
                        if(endl == 3) {
                            file_comment_count ++;
                            endl = 0;
                            t1 = t1 + ";;;";
                        }
                    }
                }
                in.close();

                strings[i] = new String[file_comment_count];
                for(int j = 0; j< strings[i].length; j++) {
                    strings[i][j] = ""; //Initializes all of the strings
                }

                reader1 = new Scanner(t1).useDelimiter(";;;");
                int j = 0;
                while(reader1.hasNext()) {
                    j++;
                    strings[i][j-1] = reader1.next();
                }
                reader1.close();

                System.out.println(files[i].toString() +":  "+
strings[i][strings[i].length-1]);
            }
            System.out.println("files done loading");
            return strings;
        }
        catch(Exception e) {
            System.out.println("not working \n" +e);
            return dummy;
        }
    }


    static int LCS(String ppwd, String corp) {
        int [][] lcs = new int[corp.length()+1][ppwd.length()+1];
```

```
		for(int i = 1; i< corp.length()+1; i++) {
			for(int j = 1; j< ppwd.length()+1; j++) {
				char ppwdc = ppwd.charAt(j-1);
				char corpc = corp.charAt(i-1);
				if(ppwdc==corpc) {
					lcs[i][j] = 1 + lcs[i-1][j-1];
				}
				else {
					lcs[i][j] = Math.max(lcs[i-1][j], lcs[i][j-1]);
				}
				/* I just left this here, it was used for error finding, but it is shows
how to work out an example which is kind of cool
				for(int k = 0; k< corp.length()+1; k++) {
					for(int l = 0; l< ppwd.length()+1; l++) {
						System.out.print(lcs[k][l] + "  ");
					}
					System.out.println("\n");
				}
				System.out.println("\n\n");
				*/
			}
		}
		return lcs[corp.length()][ppwd.length()];
	}


	public static void main(String[] args) {
		int number_of_files_to_use_in_corpus = 70; //pick any number less than 72
		String[][] docs = docs("C:/Users/samue/Downloads/NiceFiles",
number_of_files_to_use_in_corpus);
		//*/
		String ppwd = "Great app"; //"Love it";//"Great app and "; //"Great app and worth
paying for the"
		String temparay = "";
		String most_matched;
		ppwd = ppwd.toLowerCase();
		for(int i = 0; i< ppwd.length(); i++) {
			if(Character.isAlphabetic(ppwd.charAt(i))){
				temparay = temparay + ppwd.charAt(i);
			}
		}
		ppwd = temparay;
		double lcs = 0;
		double max_similarities = 0;
```

```
                    double temp = 0;
                    for(int j = 0; j< number_of_files_to_use_in_corpus ; j++) {
                              for(int i = 0; i< docs[j].length; i++) {
                                        lcs = LCS(ppwd, docs[j][i]);
                                        temp = lcs/Math.max((Math.min(docs[j][i].length(), ppwd.length())),
25);//the 25 here is just to
                                        //prevent a comment that is ~5 letters long from being counted as
plagarism against a longer comment, this is not a perfect system, but will keep misclassificatio
to a minimium
                                        if(temp > max_similarities) {
                                                  max_similarities = temp;
                                                  most_matched = docs[j][i];
                                        }
                              }
                    }

                    max_similarities = max_similarities*100;
                    if(max_similarities > 24) {
                              System.out.println("this document shows signs of potentially being
plagiarized and has a lcs that contains " + max_similarities + "% similarity with at least one
tested document");
                    }
                    else{
                              System.out.println("this document has similarities that are within tolerance
and has not likely been plagiarized (" + max_similarities +"%)");
                    }
                    System.out.println("done");
          }
}
```

Code for KMP

```
package KMP;

import java.io.File;
import java.util.Scanner;
import java.io.BufferedReader;
import java.io.FileReader;

public class KMP_main {
        static String[][] docs (String location, int how_many_document_to_check_against){
                String[][] dummy = new String[0][0];
                try {
                        String string_location = location;

                        File file_locations = new File(string_location);
                        String[] directory = file_locations.list(); //this is an array that holds all of the
names for all of the csv  files
                        File[] files = new File[directory.length];
                        String[][] strings = new String[directory.length][]; //one string for every
comment

                        Scanner reader1 = new Scanner("");
                        String t1;
                        BufferedReader in;
                        for(int i = 0; i< how_many_document_to_check_against; i++) {
//directory.length    this initializes string[][], string[i] holds files, string[][j] holds individual
comments
                                                                                                    //I used this
format for possible expansion in case there where too many comments for a computeres
memory the job could be split by strings[i]
                                String string_file_location = string_location + "/" + "/NiceFiles_" +
Integer.toString(i) + ".txt";;
                                files[i] = new File(string_file_location);

                                in = new BufferedReader(new FileReader(files[i]));

                                int file_comment_count = 0;
                                int endl = 0;
                                char tc = ' ';
                                int ti = 0;
                                t1 = "";
```

```
				while(ti != -1) {
					ti = in.read();
					tc = (char) ti;

					if(Character.isAlphabetic(tc)) {
						t1 = t1 + Character.toLowerCase(tc);;
					}
					else if(Character.isWhitespace(tc)){
						if(t1.length() != 0 &&
!Character.isWhitespace(t1.charAt(t1.length()-1)) && t1.charAt(t1.length()-1) != ';') {
							t1 = t1 + ' ';
						}
					}
					else if(tc == ';') {
						endl ++;
						if(endl == 3) {
							file_comment_count ++;
							endl = 0;
							t1 = t1 + ";;;";
						}
					}
				}
				in.close();

				strings[i] = new String[file_comment_count];
				for(int j = 0; j< strings[i].length; j++) {
					strings[i][j] = ""; //Initializes all of the strings
				}

				reader1 = new Scanner(t1).useDelimiter(";;;");
				int j = 0;
				while(reader1.hasNext()) {
					j++;
					strings[i][j-1] = reader1.next();
				}
				reader1.close();

				System.out.println(files[i].toString() +":  "+
strings[i][strings[i].length-1]);
			}
			return strings;
		}
		///*
		catch(Exception e) {
			System.out.println("not working \n" +e);
			return dummy;
		}
	}

		static int[][] PREFIX_FUNCTION(String ppwd){
		int m = ppwd.length();
		int[][] e = new int[2][m]; //
		int k = 0;
		int spaces = 0;

		for(int q = 1; q < m; q++) {
			while(k > 0 && ppwd.charAt(k) != ppwd.charAt(q)) {
				k = e[0][k];
			}

			//char t1 = ppwd.charAt(k); for testing
			//char t2 = ppwd.charAt(q);

			if(ppwd.charAt(k) == ppwd.charAt(q)) {
				k = k+1;

			}
			if(q+1 == ppwd.length() || ppwd.charAt(q) == ' ') {
				spaces = spaces +1;
				//counts number of spaces in the longest prefix that is also a suffix
			}
			e[0][q] = k;
			e[1][q] = spaces;
		}
		return e;
	}

	static int KMP_MATCHER(String comment, String ppwd, int[][] e) {
		int n = comment.length();
		//e pre-calculated
		int q = 0;

		int temp, prefix_spaces,tc = 0, count = 0;

		for(int i = 0; i< n; i++) {
			char t = ppwd.charAt(q);
			while(q>0 && ppwd.charAt(q) != comment.charAt(i)) {
```

```
                        temp = q;
                        q = e[0][q];
                        prefix_spaces = e[0][temp]-e[0][q]; //finds the number of spaces
between where q is and where it is set to
                        tc = tc - prefix_spaces; //makes sure not to double count prefixes
                        if(tc >= 3) {
                            count = count + tc;
                            tc = 0;
                        }
                        tc = 0; //this just insures that tc is set back to 0 if a letter does not
match and there are no prefix spaces
                        tc = tc + prefix_spaces;
                    }
                    if(ppwd.charAt(q) == comment.charAt(i) || q == ppwd.length()) {
                        if(i+1 == comment.length() || comment.charAt(i) == ' ') { //word end
                            tc = tc + 1;
                        }
                        else if(q+1 == ppwd.length()) { //if q == ppwd.length() then you
have read to the end of the test comment and so the last word must be counted and you must
break to avoid getting > 100%
                            tc = tc+1;
                            if(tc >= 3) {
                                count = count + tc;
                                tc = 0;
                            }
                            break;
                        }
                        q=q+1;
                    }
                    if(q>0 && ppwd.charAt(q-1) != comment.charAt(i)) {
                        if(tc >= 3) {
                            count = count + tc;
                            tc = 0;
                        }
                    }
                }
                if(tc >= 3) {
                    count = count + tc;
                    tc = 0;
                }
                return count;
            }

            public static void main(String[] args) {
```

```
        String ppwd = "So awesome. Read my";

        String temparary = "";
        ppwd = ppwd.toLowerCase();
        int words_in_ppwd = 0;
        System.out.println(ppwd);
        for(int i = 0; i< ppwd.length(); i++) {
            if(Character.isAlphabetic(ppwd.charAt(i)) ||
Character.isWhitespace(ppwd.charAt(i))){
                if(Character.isWhitespace(ppwd.charAt(i)) && i != 0 &&
!Character.isWhitespace(ppwd.charAt(i-1))) {
                    temparary = temparary + ' ';
                    words_in_ppwd ++;
                }
                else {
                    temparary = temparary + ppwd.charAt(i);
                }
            }
        }


        int [][] e = PREFIX_FUNCTION(ppwd);

        int how_many_document_to_check_against = 40; //must choose <=71 (file 72
does not exist)
        String[][] docs = docs("C:/Users/samue/Downloads/NiceFiles",
how_many_document_to_check_against);

        ppwd = temparary;

        int max = 0, temp;
        double percent;

        String comment_match = "";
        for(int j = 0; j< how_many_document_to_check_against; j++) {
            for(int i = 0; i< docs[j].length; i++) {
                if(j ==37 && i ==1799) {
                    System.out.println("this");
                }
                temp = KMP_MATCHER(docs[j][i], ppwd, e);
                if(temp > max) {
                    max = temp;
                    comment_match = docs[j][i];
```

```
                        }
                    }
                }


                percent = (max/words_in_ppwd)*100;
                if(percent > 24) {
                    System.out.println("there is a " + percent + "% match between the
document and one of the corpus document, it is likely that this document has been
plagarized\n\n");
                    System.out.println("the comment which matches the most with the
presesnted document is\n");
                    System.out.println(comment_match);
                }
                else {
                    System.out.println("there is a " + percent + "% match between the
document and one of the corpus documents, \nit is unlikely thatg the document has been
plagarized against the data set");
                }
            }
}
```

LCSS scatterplot code

```
import java.io.File;
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;
import java.io.BufferedReader;
import java.io.FileReader;

public class LCSS_main {
  static String[][] docs (String location, int number_of_files_to_use_in_corpus){
     String[][] dummy = new String[0][0];
     try {
        String string_location = location;
        File file_locations = new File(string_location);
        String[] directory = file_locations.list(); //this is an array that holds all of the names for all
of the csv  files
        File[] files = new File[directory.length];
        String[][] strings = new String[directory.length][]; //one string for every comment

        Scanner reader1 = new Scanner("");
        String t1;
        BufferedReader in;
        for(int i = 0; i< number_of_files_to_use_in_corpus; i++) { //directory.length    this initializes
string[][], string[i] holds files, string[][j] holds individual comments
          //I used this format for possible expansion in case there where too many comments for
a computeres memory the job could be split by strings[i]
          String string_file_location = string_location + "/" + "/NiceFiles_" + Integer.toString(i) +
".txt";;
          files[i] = new File(string_file_location);

          in = new BufferedReader(new FileReader(files[i]));

          int file_comment_count = 0;
          int endl = 0;
          char tc = ' ';
          int ti = 0;
          t1 = "";

          while(ti != -1) {
            ti = in.read();
            tc = (char) ti;

            if(Character.isAlphabetic(tc)) {
              t1 = t1 + Character.toLowerCase(tc);;
            }
            else if(tc == ';') {
              endl ++;
              if(endl == 3) {
                file_comment_count ++;
                endl = 0;
                t1 = t1 + ";;;";
              }
            }
          }
          in.close();
```

```
            strings[i] = new String[file_comment_count];
            for(int j = 0; j< strings[i].length; j++) {
               strings[i][j] = ""; //Initializes all of the strings
            }

            reader1 = new Scanner(t1).useDelimiter(";;;");
            int j = 0;
            while(reader1.hasNext()) {
               j++;
               strings[i][j-1] = reader1.next();
            }
            reader1.close();

            System.out.println(files[i].toString() +":  "+ strings[i][strings[i].length-1]);
         }
         System.out.println("files done loading");
         return strings;
      }
      catch(Exception e) {
         System.out.println("not working \n" +e);
         return dummy;
      }
   }



         static int LCS(String ppwd, String corp) {
                  int [][] lcs = new int[corp.length()+1][ppwd.length()+1];

                  for(int i = 1; i< corp.length()+1; i++) {
                           for(int j = 1; j< ppwd.length()+1; j++) {
                                    char ppwdc = ppwd.charAt(j-1);
                                    char corpc = corp.charAt(i-1);
                                    if(ppwdc==corpc) {
                                             lcs[i][j] = 1 + lcs[i-1][j-1];
                                    }
                                    else {
                                             lcs[i][j] = Math.max(lcs[i-1][j], lcs[i][j-1]);
                                    }
                                    /* I just left this here, it was used for error finding, but it is shows
how to work out an example which is kind of cool
                                    for(int k = 0; k< corp.length()+1; k++) {
                                             for(int l = 0; l< ppwd.length()+1; l++) {
                                                      System.out.print(lcs[k][l] + "  ");
```

```
                                        }
                                        System.out.println("\n");
                                }
                                System.out.println("\n\n");
                                */
                        }
                }
                return lcs[corp.length()][ppwd.length()];
        }




    public static void main(String[] args) {
        int[] nValues = {10000, 100000, 250000, 500000, 750000, 1000000}; // You can choose
different values for n
        List<Integer> nList = new ArrayList<>();
        List<Long> timeList = new ArrayList<>();

        for (int n : nValues) {
            long startTime = System.nanoTime();

            // Run your program with the current n value
            String[][] docs = docs("/Users/sambhavgarg/Downloads/NiceFiles", n);

            String ppwd = "Hello my name is Samuel";
            String temparay = "";
            ppwd = ppwd.toLowerCase();
            for (int i = 0; i < ppwd.length(); i++) {
                if (Character.isAlphabetic(ppwd.charAt(i))) {
                    temparay = temparay + ppwd.charAt(i);
                }
            }
            ppwd = temparay;
            int lcs = 0;
            double max_similarities = 0;
            double temp = 0;
            for (int i = 0; i < docs[0].length; i++) {
                lcs = LCS(ppwd, docs[0][i]);
                temp = lcs / Math.max((Math.min(docs[0][i].length(), ppwd.length())), 15);
                if (temp > max_similarities) {
                    max_similarities = temp;
                }
            }
            max_similarities = max_similarities * 100;
            if (max_similarities > 24) {
                System.out.println("this document shows signs of potentially being plagiarized and has
a lcs that contains " + max_similarities + "% similarity with at least one tested document");
            } else {
                System.out.println("this document has similarities that are within tolerance and has not
likely been plagiarized");
            }
            System.out.println("done");

            long endTime = System.nanoTime();
            long timeTaken = endTime - startTime;

            nList.add(n);
            timeList.add(timeTaken);
        }
    }
}
```

New KMP

```
import java.io.File;

import java.io.FileNotFoundException;

import java.util.Scanner;

import java.io.BufferedReader;

import java.io.FileReader;

public class KMP {

                static int[][] PREFIX_FUNCTION(String ppwd){

                int m = ppwd.length();

                int[][] e = new int[2][m]; //

                int k = 0;

                int spaces = 0;


                for(int q = 1; q < m; q++) {
```

```
                while(k > 0 && ppwd.charAt(k) != ppwd.charAt(q)) {

                        k = e[0][k];

                }


                //char t1 = ppwd.charAt(k); for testing

                //char t2 = ppwd.charAt(q);


                if(ppwd.charAt(k) == ppwd.charAt(q)) {

                        k = k+1;


                }

                if(q+1 == ppwd.length() || ppwd.charAt(q) == ' ') {

                        spaces = spaces +1;

                        //counts number of spaces in the longest prefix that is also a suffix

                }

                e[0][q] = k;

                e[1][q] = spaces;

        }

        return e;

}


static int KMP_MATCHER(String comment, String ppwd, int[][] e) {

        int n = comment.length();

        //e pre-calculated

        int q = 0;


        int temp, prefix_spaces,tc = 0, count = 0;
        for(int i = 0; i< n; i++) {
                char t = ppwd.charAt(q);
                while(q>0 && ppwd.charAt(q) != comment.charAt(i)) {
                        temp = q;
                        q = e[0][q];
                        prefix_spaces = e[0][temp]-e[0][q]; //finds the number of spaces between
where q is and where it is set to
                        tc = tc - prefix_spaces; //makes sure not to double count prefixes
                        if(tc >= 3) {
                            count = count + tc;
                            tc = 0;
                        }
                        tc = 0; //this just insures that tc is set back to 0 if a letter does not match
and there are no prefix spaces
                        tc = tc + prefix_spaces;
                }
                if(ppwd.charAt(q) == comment.charAt(i) || q == ppwd.length()) {
                        if(i+1 == comment.length() || comment.charAt(i) == ' ') { //word end
                            tc = tc + 1;
                        }
                        else if(q+1 == ppwd.length()) { //if q == ppwd.length() then you have read
to the end of the test comment and so the last word must be counted and you must break to avoid getting
> 100%
                            tc = tc+1;
                            if(tc >= 3) {
                                count = count + tc;
                                tc = 0;
                            }
                            break;
                        }
```

```
                        q=q+1;
                    }
                    if(q>0 && ppwd.charAt(q-1) != comment.charAt(i)) {
                        if(tc >= 3) {
                            count = count + tc;
                            tc = 0;
                        }
                    }
                }
                if(tc >= 3) {
                    count = count + tc;
                    tc = 0;
                }
                return count;
            }
            public static void main(String[] args) {
                long start = System.currentTimeMillis();
                String ppwd = "This is a great app";


                String temparary = "";
                ppwd = ppwd.toLowerCase();
                int words_in_ppwd = 0;
                System.out.println(ppwd);
                for(int i = 0; i< ppwd.length(); i++) {
                    if(Character.isAlphabetic(ppwd.charAt(i)) ||
        Character.isWhitespace(ppwd.charAt(i))){
                        if(Character.isWhitespace(ppwd.charAt(i)) && i != 0 &&
        !Character.isWhitespace(ppwd.charAt(i-1))) {
                            temparary = temparary + ' ';
                            words_in_ppwd ++;
                        }
                        else {
                            temparary = temparary + ppwd.charAt(i);
                        }
                    }
                }


                int [][] e = PREFIX_FUNCTION(ppwd);

                int how_many_document_to_check_against = 100000;
                String string_location = "C:\\Users\\jiyih\\Downloads\\NiceFiles";

                File file_locations = new File(string_location);
                String[] directory = file_locations.list(); //this is an array that holds all of the names for all
        of the csv files
                String[][] strings = new String[1][]; //one string for every comment

                ppwd = temparary;

                int max = 0, temp;
                double percent;
                int fileCounter = 0;

                String comment_match = "";
                for(int i = 0; i< how_many_document_to_check_against; i++) {
                    //System.out.println(fileCounter + " " + directory.length);
```

```
                        if(fileCounter >= directory.length) {

                                break;

                        }

                        String string_file_location = string_location + "/" + "/NiceFiles_" +
Integer.toString(i) + ".txt";


                        try {

                                File file = new File(string_file_location);

                                BufferedReader reader = new BufferedReader(new FileReader(file));

                                int file_comment_count = 0;

                                while (reader.readLine() != null) {

                                        file_comment_count++;

                                }

                                reader.close();




                                strings[0] = new String[file_comment_count+1];

                                for(int j = 0; j< strings[0].length; j++) {

                                        strings[0][j] = ""; //Initializes all of the strings

                                }


                                Scanner reader1 = new Scanner(file).useDelimiter(";;;");

                                int j = 0;

                                while(reader1.hasNext()) {

                                        j++;

                                        strings[0][j-1] = reader1.next().replaceAll("[^a-zA-Z0-9 ]",
"").toLowerCase();

                                }

                                reader1.close();
```

```
                //System.out.println(file.toString() +": "+ strings[0][strings[0].length-1]);

            }

            catch(Exception e1) {

                System.out.println("Can't read file \n" +e1);

                continue;

            }

            fileCounter++;

            for(int j = 0; j< strings[0].length; j++) {

                temp = KMP_MATCHER(strings[0][j], ppwd, e);

                if(temp > max) {

                    max = temp;

                    comment_match = strings[0][j];

                }

            }

            if(fileCounter % 500 == 0) {

                System.out.println("Files Checked: " + fileCounter);

                System.out.println("Time Taken: " + (System.currentTimeMillis() - start));

            }

        }




        percent = (max/words_in_ppwd)*100;

        if(percent > 24) {

            System.out.println("there is a " + percent + "% match between the document and
one of the corpus document, it is likely that this document has been plagarized\n\n");

            System.out.println("the comment which matches the most with the presesnted
document is\n");

            System.out.println(comment_match);
```

```
                    }
                    else {
                              System.out.println("there is a " + percent + "% match between the document and
one of the corpus documents, \nit is unlikely thatg the document has been plagarized against the data
set");
                    }
                    System.out.println("Files Checked: " + fileCounter);
                    System.out.println("Time Taken: " + (System.currentTimeMillis() - start));
          }
}
```

New LCSS

```
import java.io.File;
import java.io.FileNotFoundException;
import java.util.Scanner;
import java.io.BufferedReader;
import java.io.FileReader;
public class LCSS {
          static String[][] docs (String location, int number_of_files_to_use_in_corpus, int i) throws
Exception{
                    File file_locations = new File(location);
                    String[] directory = file_locations.list(); //this is an array that holds all of the names for all
of the csv files
                    String[][] strings = new String[1][]; //one string for every comment


                    int fileCounter = 0;


                    //System.out.println(fileCounter + " " + directory.length);
                    String string_file_location = location + "/" + "/NiceFiles_" + Integer.toString(i) + ".txt";
```

```
        File file = new File(string_file_location);

        BufferedReader reader = new BufferedReader(new FileReader(file));

        int file_comment_count = 0;

        while (reader.readLine() != null) {

                file_comment_count++;

        }

        reader.close();



        strings[0] = new String[file_comment_count+1];

        for(int j = 0; j< strings[0].length; j++) {

                strings[0][j] = ""; //Initializes all of the strings

        }


        Scanner reader1 = new Scanner(file).useDelimiter(";;;");

        int j = 0;

        while(reader1.hasNext()) {

                j++;

                strings[0][j-1] = reader1.next().replaceAll("[^a-zA-Z0-9 ]", "").toLowerCase();

        }

        reader1.close();


        //System.out.println(file.toString() +": "+ strings[0][strings[0].length-1]);

        return strings;

}
```

```
	static int LCS(String ppwd, String corp) {

		int [][] lcs = new int[corp.length()+1][ppwd.length()+1];


		for(int i = 1; i< corp.length()+1; i++) {

			for(int j = 1; j< ppwd.length()+1; j++) {

				char ppwdc = ppwd.charAt(j-1);

				char corpc = corp.charAt(i-1);

				if(ppwdc==corpc) {

					lcs[i][j] = 1 + lcs[i-1][j-1];

				}

				else {

					lcs[i][j] = Math.max(lcs[i-1][j], lcs[i][j-1]);

				}

				/* I just left this here, it was used for error finding, but it is shows how to
work out an example which is kind of cool

				for(int k = 0; k< corp.length()+1; k++) {

					for(int l = 0; l< ppwd.length()+1; l++) {

						System.out.print(lcs[k][l] + " ");

					}

					System.out.println("\n");

				}

				System.out.println("\n\n");

				*/

			}

		}

		return lcs[corp.length()][ppwd.length()];

	}
	public static void main(String[] args) {

		long start = System.currentTimeMillis();

		int number_of_files_to_use_in_corpus = 1000000;

		String string_location = "C:\\Users\\jiyih\\Downloads\\NiceFiles";




		String ppwd = "This is a great app";

		String temparay = "";

		String most_matched;

		ppwd = ppwd.toLowerCase();

		for(int i = 0; i< ppwd.length(); i++) {

			if(Character.isAlphabetic(ppwd.charAt(i))){

				temparay = temparay + ppwd.charAt(i);

			}

		}

		ppwd = temparay;

		double lcs = 0;

		double max_similarities = 0;

		double temp = 0;

		int fileCounter = 0;

		String[][] docs = null;

		File file_locations = new File(string_location);

		String[] directory = file_locations.list(); //this is an array that holds all of the names for all
of the csv files

		for(int i = 0; i< number_of_files_to_use_in_corpus ; i++) {

			if(fileCounter >= directory.length) {

				break;
```

```
            }
            try {
                docs = docs(string_location, number_of_files_to_use_in_corpus, i);
            }
            catch(Exception e1) {
                //System.out.println("Can't read file \n" +e1);
                continue;
            }
            fileCounter++;
            for(int j = 0; j< docs[0].length; j++) {
                lcs = LCS(ppwd, docs[0][j]);
                temp = lcs/Math.max((Math.min(docs[0][j].length(), ppwd.length())),
25);//the 25 here is just to
                //prevent a comment that is ~5 letters long from being counted as
plagarism against a longer comment
                //this is not a perfect system, but will keep misclassification to a minimum
                if(temp > max_similarities) {
                    max_similarities = temp;
                    most_matched = docs[0][j];
                }
            }
            if(fileCounter % 500 == 0) {
                System.out.println("Files Checked: " + fileCounter);
                System.out.println("Time Taken: " + (System.currentTimeMillis() - start));
            }
        }

        max_similarities = max_similarities*100;
        if(max_similarities > 24) {
            System.out.println("this document shows signs of potentially being plagiarized and has a lcs that contains " + max_similarities + "% similarity with at least one tested document");

        }

        else{

            System.out.println("this document has similarities that are within tolerance and has not likely been plagiarized (" + max_similarities +"%)");

        }

        System.out.println("done");

        System.out.println("Time Taken: " + (System.currentTimeMillis() - start));

    }

}
```