Probability of Imitating a Writing Style

Aaron Weckerly

Data Structures and Algorithms

Dr. Mullins

TR 9:30-10:45

The experiment discussed in this paper will use a program written by Aaron Weckerly, assigned by Dr. Mullins as a project in Data Structures and Algorithms. The program is designed to pick a word of a chosen length from a text. That seed word is then used to find letters, and eventually words, likely to follow. The program takes all characters finding and instance of the seed word, stores them, and choses one at random. The seed word is changed as the program finds each new letter by removing the first letter, and adding the new character, thus creating a new seed word. This process continues for a user-determined amount of characters.

I have hypothesized that in order for the search engine Google to show a result of the author of the original text in the top three results, the seed word will have to be at least 70% the size of the largest word in the text for Alice’s Adventures in Wonderland and that anything below a seed size of 3 will not return a result to the author or text. I will test this using the search engine Google while using Google’s, Google Chrome browser in Incognito mode. I will attempt to limit Google storing information and trying to predict what results I want. I will attempt to make the results as unbiased as possible by reopening a new browser in Incognito mode each time I do a search. Google seems to keep fewer records when using Incognito mode therefore limiting the chance of Google noticing a pattern in my searches and making each search more independent. See appendix 1 for a screen shot of Google Chrome’s Incognito browsing information.

Developed string outputs of 100 characters can be found in Appendix 2. The text is generated using the earlier described program. The results of the searches showed a much lower seed length required than originally hypothesized. The first time a result of either the author or text was returned in the top three was for a seed length of 4 one out of the two times original searched. A seed length of 5 returned results in the top three both times originally searched. This is likely due to the large seed that is given to Google to use. In essence it is extremely similar in nature. Many of the words are extremely common considering all their possible uses. However because the number of characters is so large, Google is able to use that to find a result with all of them in close to the original order.

The results from searching Google with 100 character strings generated from varying random seed lengths resulted in the finding that a seed length of 4 was the first to have the author appear in the top three results. I continued to test this length with random seeds and from 15 searches, 5 showed results in the top three. These searches also included 5 from texts other than Alice’s Adventures in Wonderland, shown in Appendix 2. One of the 5 from authors other than Lewis Carroll returned a positive result to its respective author. All strings generated from a random seed of size 11 or above returned a positive result with two searches of each size.

The measures taken to keep each search independent appeared to be successful with no real correlation between the results. I feel if the searches had been smaller and more specific there would perhaps be more of a correlation between the results. The results from the experiment were impressive with 33% of the searches of a 100-character string based off a 4 letter seed returning a positive result. Had the searches been of a smaller size the results would have without a doubt returned fewer positive results. A 100-character search was chosen because it was practical and logical.

A string of 1000 characters was generated off a random 4-letter seed word. The result can be seen at the end of Appendix 2. The characters derived read very much like Lewis Carroll text, especially for being based off the word “they.” A search was attempted on the entire string, but Google limits searches to 32 words. After Google searched for the first 32 words of the string, no results were found.

This shows that with enough sample writing from a text, Google will return a result. Google most likely uses an algorithm that contains at least in part a spin off of something similar to that of the one used in this experiment. Google’s algorithm is undoubtedly far more complicated. If a Google search is done with no modifiers, the algorithm has a hard job to do. If a modifier such as ‘author’ is chosen to limit the search field, millions of results will be cut from the search field. Searches vary greatly in size and difficulty. Small searches for large public figures such as “Barak Obama” are most likely more common than searches for “the Mock Turtle's heavy sobbing of tumbling up again before they saw her, though, as the White Rabb.” Google is likely to use a sort of “predictive text” type feature to narrow results as characters are typed. Because “Barak Obama” may be searched for quite often, especially around election time, Google can “see” the rest of the search after only a few characters have been typed. Google would use current events and the number of searches to keep a type of current database for the searches and then indexes using keywords and topics. Google would keep a record of all web pages and words appearing on them. The number of appearances a word makes on a webpage and proximity from one search word to another would also make an appearance in their algorithm.

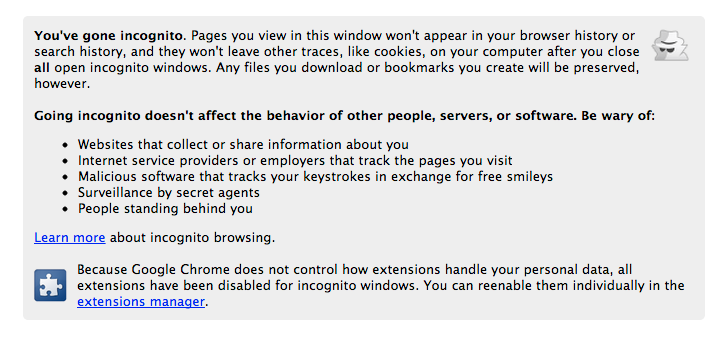
The use of the type of algorithm in this experiment to determine if the same person writes two pieces of work is possible, however, the particular algorithm used for this experiment is extremely rough. It would be extremely difficult for an algorithm to determine an author’s style. A writing style is far more than simply the probability of the next character to be expected. This algorithm is heavily dependent upon the sample text. If the sample text contained many of the same phrases that were topic specific, it would be difficult to cross check it with a sample text containing information on a far different topic. There would also need to be enough sample texts to search. With great modification to the algorithm in this experiment an algorithm could be derived to determine a probability of determining if the same person authored two texts. I feel the likelihood of this being a practical solution to crosschecking papers would have to take into factors such as wood length, sentence length, and, although what would be a lengthy algorithm, sentence structure. This proposed algorithm would have what would be similar tools as the spell checking on Microsoft Word.

This algorithm will reproduce similar phrases as found in the sample text, therefore could be modified slightly for a predictive texting style application. Simply by taking the recently typed characters into something such as a phone, and running the algorithm with a sample text from a sample of previously typed strings. The likelihood of the word generated being the correct one will increase greatly as the seed grows in size. This algorithm is far closer to being one used for a predictive texting implementation than that of determining the author of a text.

There are many limitations to the algorithm as it is currently. The theory and logic behind the algorithm offer many opportunities to expand. With large seeds that are show up a limited number of times in the text allow for the algorithm to almost copy the text. As shown in the example generated texts in Appendix 2. When a seed word of length 14 is chosen, the seed is so large and unique that it almost reproduces the text at the beginning. This is because the odds of the program choosing the next character exactly the same greatly increase because of the limited number of other options. The generated text becomes more random as it progresses because the text becomes less unique. It is the theory and logic behind this algorithm that provide all the opportunities. Humans are “creatures of habit,” therefore if a program keeps track of user history; anything out of the ordinary can be spotted with relative accuracy. In many times a “better safe than sorry” approach is quite fitting, allowing a program to raise a flag and allow simple user identification or verification to keep the program secure or otherwise safe.

The algorithm in this experiment is somewhat of a building block toward what could be huge and powerful algorithms. The most powerful algorithms have to take into account user behavior from past and current events. Predicative texting on smart phones uses and algorithm to remember and predict as text entry takes place using probabilities and an algorithm that is most likely very similar to the one discussed in this paper. The algorithm could be expanded to include general user interaction on websites or various other computer programs to determine when something is out of the ordinary. Whether text or simply activity user behavior can almost be predicted especially if a higher priority is given things that may be more likely to appear given time or current events. The possibilities for this extend far greater than any applications we see today.

Appendix 1



Appendix 2

Used text of Alice’s Adventures in Wonderland

This Appendix contains outputs using the described program and algorithm and the result when searched on Google with described procedures.

Seed Length 14

1) Returned authors name in first result

The seed word is contemptuously

The generated text is: . 'Who are YOU?' said the Caterpillar, and the words all coming different, and then the Mock Turtle'

2) Returned authors name in first result

The seed word is contemptuously

The generated text is . 'Who are YOU?' said the Caterpillar, just as if she had known them all her life. Indeed, she had q

Seed Length 13

1) Returned authors name in first result

The seed word is uncomfortable

The generated text is: . The moment Alice felt a very curious.' 'It's all about as curious as it can be,' said the Gryphon,

2) Returned authors name in first result

The seed word is extraordinary

The generated text is: noise going on within--a constant howling and sneezing, and every now and then treading on her toes

Seed Length 12

1) Returned authors name in first result

The seed word is consultation

The generated text is: about this, and after a few minutes, and she jumped up in such a hurry that she tipped over the edg

2) Returned authors name in first result

The seed word is refreshments

The generated text is: !' But there seemed to be no chance of getting up and walking off to other parts of the ground, Alic

Seed Length 11

1) Returned authors name in first result

The seed word is indignantly

The generated text is: . 'Let me alone!' 'Serpent, I say again!' repeated the Pigeon. 'I'm NOT a serpent!' said Alice, 'bec

2) Returned authors name in first result

The seed word is unfortunate

The generated text is: gardeners at it, busily painting them red. Alice thought decidedly uncivil. 'But perhaps he can't h

Seed Length 10

1) Returned authors name in first result

The seed word is Arithmetic

The generated text is: --Ambition, Distraction, Uglification, and the fan and the pair of white kid gloves, and she said th

2) Neither author or text appeared in top three results

The seed word is passionate

The generated text is: voice. 'Would YOU like cats.' 'Not like cats.' 'Not like cats.' 'Not like cats.' 'Not like cats.' '

Seed Length 9

1) Returned authors name in first result

The seed word is somewhere

The generated text is: near the rattle of the other was sitting next to her. Then followed a shower of saucepans, plates,

2) Neither author or text appeared in top three results

The seed word is happening

The generated text is: . While she ran, as well as she could remember,' said these words have got altered.' 'It isn't,' sai

Seed Length 8

1) Returned authors name in first result

The seed word is unfolded

The generated text is: the Mock Turtle's heavy sobbing of tumbling up again before they saw her, though, as the White Rabb

2) Returned authors name in first result

The seed word is machines

The generated text is: in the window, and the shrill little cartwheels, and shouted the Gryphon, lying fast asleep, and th

Seed Length 7

1) Returned authors name in first result

The seed word is grinned

The generated text is: ; in fact, there was said a timid and tried the Queen of Hearts, she made up my mind about the top o

2) Returned authors name in first result

The seed word is hearing

The generated text is: anything seen, who had spoken first.' 'Why?' 'IT DOES THE BOOTS AND SHOES.' the Gryphon, lying on t

Seed Length 6

1) Returned authors name in first result

The seed word is others

The generated text is: thatched herself, 'Now, who will do, to be no change the little timidly; 'some of yourself. Imagine

2) Returned authors name in first result

The seed word is enough

The generated text is: !' They very difficulty was, 'Why, what am I to get to,' said that cats always get through the began

Seed Length 5

1) Returned authors name in first result

The seed word is Alice

The generated text is: guessed herself. 'Of court was over me left off. 'Give your age knew the end of your very confused

2) Returned authors name in first result

The seed word is shiny

The generated text is: ?' Alice hastily, reminding of fright as she ran; but some minute or might, 'it'll fetch this cheeks

Seed Length 4

1) Neither author or text appeared in top three results

The seed word is come

The generated text is: ventures his time you mean importunately much a few little you know, I beautiful with one was be in

2) Returned as second result

The seed word is wood

The generated text is: . 'Nothing behead say tone. 'Of confusion, you know was just she had not tale, (she kitchen, you gav

3) One negative result, no positive results.

The seed word is Here

The generated text is: ! Digging Alice waterpillar, and his how eagerly, my dear, who sense!' she very tone. He lobsters, I

4) Negative result (not in top three)

he seed word is THAN

The generated text is: A MILE HIGH TO LEAVE my like took piece as he stairs! There well us about one eyes flamining now we

5) Negative result (not in top three)

The seed word is they

The generated text is: good to herses.' And to her converself, 'Why, SHE does.' 'Well,' she said Alice, though, unting add

6) Negative result (not in top three)

The seed word is they

The generated text is: was believe immediately rude, and opportunity fork at timid a bad, your eyes: thin--all seem to Tim

7) Positive result in result number 2

The seed word is than

The generated text is: ked inwards, and a question.' He was the Dormouse. They're lives follow voice. 'And more couraging q

8) Negative result, not in top three results

The seed word is eyes

The generated text is: , you needn't ever,' said the Duchess?' thought Alice, that curled at the other fact, thatchen,' sho

9) Returned as first result

The seed word is were

The generated text is: not other fact, I'll nose. 'But perhaps you maybe,' Alice look at the Rabbit: for mour first, along

10) Returned as second result

The seed word is went

The generated text is: of could me trying invitated the Which was if it: the Knave begin.' In the thing to set they pigeon

Seed Length 3

1) Neither author or text appeared in top three results

The seed word is Now

The generated text is: arried, Hared, but and by to sprest in a most in her and him: therself, an near was it began heady?

2) No results

The seed word is was

The generated text is: mine of THERE,' said goesn't little powd awfull chand soment sound, and was terrupted the Dormouse

Seed Length 2

1) No results

The seed word is in

The generated text is: g shright,' so. 't see: an't ther lan thre.' to ge? I alk fell, thill to th momed shemnle don cal in

2) No results

The seed word is if

The generated text is: to he Cat canagingo youstratup ing and and the rumpturest camet's of it en, 'Do shed: 'Con't to cur

Seed Length 1

1) No results

The seed word is I

The generated text is: ' re aul, ngicethare ve laske ce thine Hage dfothte th than h, alim s t chieshe e ngag t cake de ' f

2) One result (French)

The seed word is I

The generated text is: Alime arst whoorpsmphewat t fon! Ch ange dr Quld itt Alid sint'tham wito TESOU?' de 's ISe brt Set

Daniel Miller posted the results below on the D2L discussion board.

**Length 4**

\*\*\*\*Please wait while the File is being prepared\*\*\*\*  
We will be working with Walden.txt

The seed length being used is 4 and the seed is true  
"true" which merel or celess gather to the roastward, were in my news?" I feet a qualid a lamp of the need

We will be working with WutheringHeights.txt

The seed length being used is 4 and the seed is once  
"once", and the pony, insister. He could have error delighter's breasons, to he window-pan: you, Miss Cath

We will be working with HuckleberryFinn.txt

The seed length being used is 4 and the seed is them  
"them" wouldn't none show thers?" "Hare-lip and white, the warn't you felt sign it was to do what was for

We will be working with BrothersGrimm.txt

The seed length being used is 4 and the seed is some  
"some"thing; so afraids she; 'yonderneat road, but as eighbourhood and the mother me above, basked, you wi

We will be working with BrothersGrimm.txt

The seed length being used is 4 and the seed is with  
"with" one of him summon names her it once trouble. Her so that she. 'I can't knowing to the saw so had no

After searching Dan’s generated text a positive result was found for only Huckleberry Finn, the rest returned either no results or a negative result.

A seed length of 4 was chosen and a string of 1000 characters was generated.

The seed word is they

The generated text is: 're gave of life.' 'But she mouth closed her seriously. 'Who are! I've go?" his time to her one act is over) '--the Mock in anxiously into it. So your Majesty, as went, and, turned a bit,' she knew so land the March, I trying on the Class by tone, 'to be only breatured about of swimming it.' (For, and yet I don't was the March Hardly knew, were growing. What for some the Cheshire could beautify is, she begin?' The mistantly teacup they repeated to remember,' Alice. 'Behead-and and see, being what stupid the the the Hattering to expressed up, but it, wered off at lefthan to finished offended it up and she day! Who felt a blows suppose to one, 'because slate hungry the when four feel with the box-- Allowed like a watching from with great curiosity, and she pardone floor, as fact. 'Get usually yesterday you've golden keep herself in converse.' 'We much as, I don't lessons?' the felt very line: 'that's taken hour to the Hattends a little into the look at every moral to find this, the no ri

Appendix 3

This appendix contains the code from the program containing the discussed algorithm for use in the experiment.

//Aaron Weckerly

//CPSC 374

// Project2 "Write like Lewis Carroll"

//Main

**import** java.io.FileNotFoundException;

**import** java.util.Scanner;

**public** **class** Main {

/\*\*

\* **@param** args

\* **@throws** FileNotFoundException

\*/

**public** **static** **void** main(String[] args) **throws** FileNotFoundException {

Scanner scan = **new** Scanner(System.*in*);

**int** seedLength, genLength; //will be used to store user input

System.*out*.println("This program will choose a seed word from a text file and generate characters from that seed");

System.*out*.println("The generated characters will be selected at random by finding which ones would likely to follow");

//Prompts user for seed length

System.*out*.println("Please enter the length of word you would like to use as a seed");

System.*out*.println("Hint: The longest word in the English dicitonary is 45 characters");

seedLength = scan.nextInt(); //gets length of seed from user

//if length seems too large, prompts user again

**if**(seedLength > 45){

System.*out*.println("You have chosen a seed length of "+seedLength+" This seems a little large");

System.*out*.println("An extremely large seed length will effect the efficiency of the program");

System.*out*.println("Please enter you seed length again, this will be the length used");

seedLength = scan.nextInt();

}//end if large seed

//Prompts user for character generation length

System.*out*.println("Please enter the length of characters you would like to generate from the seed");

genLength = scan.nextInt();

//error checking for extremely large gen length

stringGen alice = **new** stringGen(seedLength, genLength);

//System.out.println("The seed used was "+alice.getSeed());

//System.out.println("The generated text is "+alice.getGenText());

}

}

//Aaron Weckerly

//CPSC 374

//Project 2 "Write like Lewis Carroll"

//stringGen class - Generates string based off seed from text

**import** java.util.ArrayList;

**import** java.lang.Character;

**import** java.util.Scanner;

**import** java.util.Random;

**import** java.io.File;

**import** java.io.FileNotFoundException;

**public** **class** stringGen {

**int** MAX = 45; //creates an array large enough to store words from 1 - 45 letters long. the shortest and longest in the english dictionary

**private** String book;

**private** String seedWord, genText;

**private** ArrayList<String> words[] = (ArrayList<String>[])**new** ArrayList[MAX];

{

**for** (**int** i = 0; i < MAX; i++)

words[i] = **new** ArrayList<String>();

}

**private** ArrayList<Character> letters = **new** ArrayList<Character>();

**public** stringGen(**int** seedLength, **int** genTextLength) **throws** FileNotFoundException{

loadBook();

seedWord = findSeed(seedLength);

genText = findText(seedWord, genTextLength);

System.*out*.println(" ");

System.*out*.println("The seed word is "+seedWord);

System.*out*.println("The generated text is "+genText);

}

//public function to get seed word

**public** String getSeed(){

**return** seedWord;

}

//public function to get generated text

**public** String getGenText(){

**return** genText;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*load book\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//private helper function to load book into string

**private** **void** loadBook() **throws** FileNotFoundException{

Scanner fileIn = **new** Scanner(**new** File("src/alice.txt"));

**while**(fileIn.hasNext()){

book = book+fileIn.next();

book = book+" ";

}//while

}//loadbook

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*find seed\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//helper function to be called by stringGen to get seed

**private** String findSeed(**int** seedLength){

parse();

//generate number for random location of word at specified length

**int** index = seedLength-1;

**while**(words[index].isEmpty() && index>0)

{

index--;

}

Random randNum = **new** Random();

**int** number = (words[index].size())-1;

**if**(number<1)

number=1;

**int** randNumber = randNum.nextInt(number);

String test = words[index].get(randNumber); //returns seed from location randomly generated between 0 and size

**return** test;

}//find seed

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*findText\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//helper function to be called by stringGen to get text string

**private** String findText(String seed, **int** genTextLength){

String Seedstr = seed;

String Finalstr ="";

System.*out*.println("Generating char number: ");

**for**(**int** i = 0; i < genTextLength; i++){

**int** index = 0;

//int booklength = book.length();

System.*out*.print((i+1)+"..");

//loop searches through book to find chars following seed

**int** lastIndex = book.lastIndexOf(Seedstr)+Seedstr.length()+1;

**while**(index < lastIndex){

**int** location = book.indexOf(Seedstr, index);

**char** ch;

**if**(location+(Seedstr.length()) < book.length())

ch = book.charAt(location+(Seedstr.length()));

**else** //if looking for char at end of book, make is a space

ch = '\b';

letters.add(ch);

index=index+1;

}//while

Random randNum = **new** Random(); //randomly generates a number to choose a letter

**char** a = (letters.get(randNum.nextInt(letters.size())));

Seedstr = Seedstr+a;

Finalstr = Finalstr+a;

Seedstr = Seedstr.substring(1);

System.*out*.println(Seedstr);

letters.clear(); //clears list to use for next 'seed'

} //for

**return** Finalstr;

}//findText

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*parse\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//helper function to be called by getSeed

**private** **void** parse() {

String delims = "[ -.,;:?!\\n\\r]+";

String word[] = book.split(delims);

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

words[(word[i].length())-1].add(word[i]);

}

}//parse