UMSL
University of Missouri–St. Louis

University of Missouri – St Louis.

Project 2 Report

Student :
Mihir Bhakta
Scott Sanchez

Professor :
Dr. Badri Adhikari

October 30, 2024

Contents

1 Introduction

In this project, we implemented three different methods for text analysis, with the goal of comparing a set of essays to find the similarity between the essays.

Namely, the three methods were:

1. **Semantically Matching Paragraph Counter (SMPC) Method** - A method for text analysis that uses word frequency as its primary method of comparison.
2. **Cosine Similarity Method** - A method that vectorizes both texts to compare, and geometrically compares those two vectors, in order to determine the similarity. We suspected this may prove faster than the SMPC method.
3. **Fingerprint Method** - A method that hashes segments of text into "fingerprints". Khuat, Tung, et al. [KHTMH15] found this method to be more accurate than Cosine Similarity.

2 Collaboration

2.1 Division of Labor

We divided the work based on our interests and strengths. This allowed us to collaborate in a way where each of us was giving as much as we could.

Mihir lead us in research for the project. It was him who identified the Fingerprint and Cosine Similarity methods as viable candidates for Methods 2 and 3. Before that, neither of us had known about these algorithms.

Mihir was also the lead developer for our implementation of the Cosine Similarity method.

Scott conceived and coded the SMPC method. He also handled the coding for the Fingerprint method.

We worked jointly on the main class, ‘language_project.py‘

Together, we tested each method and analyzed results.

Additionally, we worked together to complete the project report.

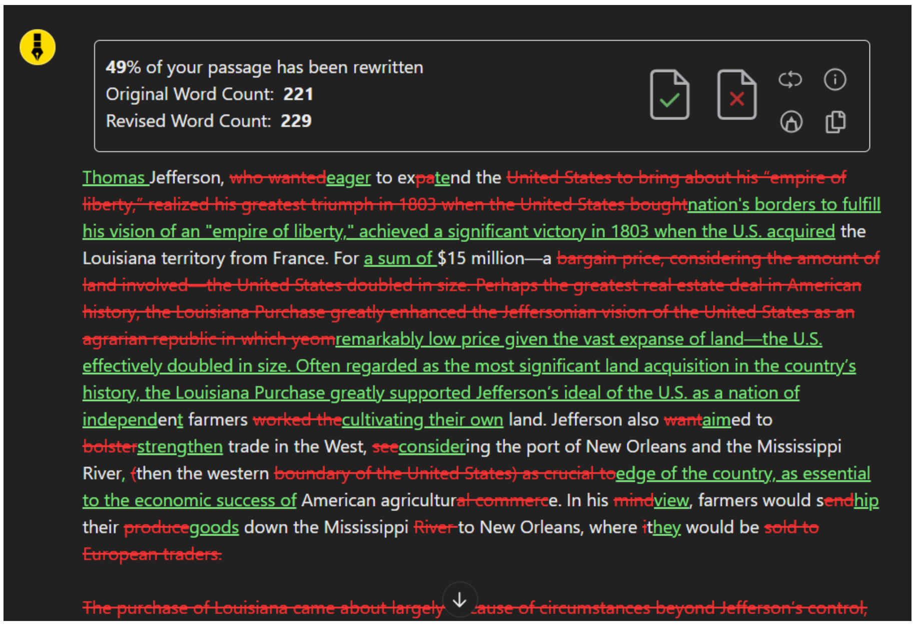
2.2 Meetings

During this project, we met regularly to divide tasks, talk about the work we performed, and ensure that each component of the project worked as expected.

We held a total of 11 phone calls between October 17th and October 29th to discuss the work being done. The shortest of these calls was between 45 minutes; the longest was 7 hours.

In addition to this, we kept in continuous communication through text communication.

We were unable to meet frequently in person due to conflicting schedules and fall break.


49% of your passage has been rewritten
Original Word Count: **221**
Revised Word Count: **229**

Thomas Jefferson, ~~who wanted~~eager to ex~~pa~~tend the ~~United States to bring about his "empire of liberty," realized his greatest triumph in 1803 when the United States bought~~nation's borders to fulfill his vision of an "empire of liberty," achieved a significant victory in 1803 when the U.S. acquired the Louisiana territory from France. For a sum of $15 million—a ~~bargain price, considering the amount of land involved—the United States doubled in size. Perhaps the greatest real estate deal in American history, the Louisiana Purchase greatly enhanced the Jeffersonian vision of the United States as an agrarian republic in which yeom~~remarkably low price given the vast expanse of land—the U.S. effectively doubled in size. Often regarded as the most significant land acquisition in the country's history, the Louisiana Purchase greatly supported Jefferson's ideal of the U.S. as a nation of independent farmers ~~worked the~~cultivating their own land. Jefferson also ~~want~~aimed to ~~bolster~~strengthen trade in the West, ~~see~~considering the port of New Orleans and the Mississippi River, ~~(~~then the western ~~boundary of the United States) as crucial to~~edge of the country, as essential to the economic success of American agricultur~~al commerce~~e. In his ~~mind~~view, farmers would s~~end~~hip their ~~produce~~goods down the Mississippi ~~River~~ to New Orleans, where ~~i~~they would be ~~sold to European traders.~~

~~The purchase of Louisiana came about largely~~ ~~ause of circumstances beyond Jefferson's control,~~

Figure 1: A Conversation with EditGPT, showing that two texts with similar meanings can appear very different to a system that compares them word-by-word. Sample text taken from Openstax: US History [Ope14]

3 Preliminary Exploration: EditGPT

We first explored the idea of comparing texts using the EditGPT Chrome Add-On. EditGPT compares texts in a manner similar to GitHub. It compares word-for-word and uses green text to denote additions, and red text to denote removals.

We immediately noticed something striking, shown in Figure 1. When using word-for-word comparisons, two texts with similar meanings can appear very different.

We found that we could give EditGPT a piece of text, and ask it to change its vocabulary and structure, while keeping the meaning. The result was that the newly-produced text appeared to be entirely different - the change comparison showing mostly red and green.

This is a false positive - the two texts appear different to the machine, but are quite clearly similar to a human.

This informed our intuition with this project. We proceeded with the rest of the project on the understanding that simple measures like word frequency and word order alone are unlikely to produce accurate comparisons.

4 Literature Review

Our next step in developing this project was to research existing methods. Our initial investigation was important for understanding what was possible for us to do in the python program that we would create. The literature we reviewed gave us guidance on how to structure our project.

Perhaps most importantly, we read Chapter 23 from the textbook, *Artificial Intelligence: A Modern Approach* by Russel & Norvig [RN20]. Their discussion of WordNet was particularly important to this project, because it showed WordNet as a tool for distinguishing word meanings. We therefore used WordNet in our project, as a tool to help find synonyms.

We also drew from Khuat, Tung, et al.[KHTMH15], who wrote a document comparing several similarity algorithms. The study covered, among others, the Cosine Similarity Method and the Fingerprint Method. Both of these, we chose to implement in our project. The formulas it gave for those methods formed the basis for our code implementations.

Additionally, we looked at a paper by Shanmathi et al. [SSAS24], who focused on automated plagiarism detection. This helped us to appreciate what text similarity detection schemes look like accross real-world use-cases. Furthermore, this paper helped us understand the Cosine Similarity Method better.

5 The Python Project

Next, we went about creating a python program that would implement our chosen methods. We would then use this program to compare all the essays from the Automated Essay Scoring 2.0 Database[Kag24].

We will include our code in the submission for this assignment. Its documentation should make the particulars of its workings quite clear. For this reason, we will not include the specifics of our implementations.

5.1 Project Structure

The most important files and directories in this project are as follows:

1. **language_project.py** - This is the main class for the project. It loads the essay, dataset, uses each method to compare each pair of essays, and writes the results to similarity_results.csv.
2. **language_project_parallel.py** - A revised version of language_project.py that ran comparisons using multi-threading
3. **cosine_similarity.py** - This file contains the implementation of the Cosine Similarity Method.
4. **fingerprint_method.py** - This file contains the implementation of the Fingerprint Method.
5. **semantically_matching_paragraph_counter_method.py** - This file contains the implementation of the Method.
6. **requirements.txt** - This is a text file containing necessary dependencies that need to be installed.
7. **setup.sh** - This is a bash script that installs the dependencies from the requirements.txt file. It also downloads the resources that the SMPC method uses to swap words with synonyms.
8. /**resources** - This directory contains input files, for example, the set of essays to compare, or the wordslists used by the SMPC method.
9. /**output** - This directory contains the CSV files outputted by the main class.

5.2 Method 1: Semantically Matching Paragraph Counter (SMPC)

5.2.1 Explanation

The Semantically Matching Paragraph Counter (SMPC) method is a text comparison algorithm designed by Scott. It uses word frequency as its primary measure in comparing two texts. The main idea behind the algorithm is this:

Two texts are likely similar if several of their paragraphs mean the same things. So, we may measure the similarity between two texts by counting the number of “matching pairs” (P_A, P_B) Where:

- P_A is a paragraph from the first text, and
- P_B is a paragraph from the second text, and
- P_A and P_B mean the same thing.

The question might arise how one might measure the similarity of two paragraphs. To do this, one must consider the different kinds of words in English. The SMPC method distinguishes between three categories of words:

1. **Function Words -** These are the most common words in the English language, like "and," "the," and "is." These words contribute little to the semantic content of a paragraph.
2. **Core Vocabulary -** These are the “medium-common” words in English. They tend to contribute somewhat to the semantic content but often have multiple common synonyms. Because of synonyms, the specific choice of word may not change the meaning of the text. For instance, "cheery" and "jolly" may be used interchangeably.
3. **Specialized Words -** These are rarer and more domain-specific terms such as "neume" or "tourbillon." These words are assumed to contribute significantly to the meaning of a paragraph or text, as they often lack common synonyms and are more specific. These words tend to only occur in texts where their meaning significantly impacts the meaning of the text.

The SMPC method works as follows:

1. **Step 1: Function Words Removal -** The first step in the SMPC method is to remove function words from both texts. Because these words contribute little to the meaning of the text, we may remove them without losing much of the original meaning. And because they are so frequent, removing them significantly reduces the amount of data that we must process. For the list of words considered "function words" in our python project, I used the list of the top 100 most common English words from the Oxford Corpus[Oxf11].
2. **Step 2: Core Vocabulary Replacement -** Next, core vocabulary words are replaced with their most common synonym. For example, “cheery” and “jolly” are both replaced with “happy”. This step ensures that two texts that use synonyms can still be recognized as semantically similar. To accomplish this step in the python program, I used WordNet - a tool which I discovered in Chapter 23 of Russel &

Norvig [RN20]. For the list of words considered "core vocabulary" In the python implementation, I initially wanted to use a wordlist from WordFrequency.info, but due to cost constraints, I opted for the freely-available "English Word Frequency" Database from Kaggle[Tat17]. That database contains 333,333 words, though for the "core vocabulary" set, I took only the commonest 5500. Ultimately, this was a matter of judgment - I would be interested to see the results if I were to choose a larger list.

3. **Step 3: Large-Scale Text Matching -** Next, the algorithm calculates the word frequencies for both of the now-processed texts. It finds the top 10 most frequent words from each. If the two top-10 lists share fewer than 3 words, then they probably aren't talking about the same thing. In that case, the algorithm returns a score of 0. If the two top-10 lists share 3 or more words, then the algorithm continues.

4. **Step 4: Paragraph Matching -** Next, the algorithm compares each paragraph from the first text against each paragraph from the second text. The same method of comparison is used as was described in Step 3. The 10 most frequent words are found from both paragraphs. If the two top-10 lists share 3 or more words, then the paragraphs are said to be a match.The SMPC method counts all matches and reports the number as a final "similarity score". The python program ultimately normalized these scores to a range between 0 and 1.

In short, the SMPC method:

1. Filters out irrelevant words,
2. Replaces common terms with synonyms,
3. Does a large-scale check to see if the documents match as a whole, and
4. Counts the number of paragraphs that seem to match.

5.2.2 Time Complexity

Most steps in SMPC are of class O(n). Filtering words from a list, and replacing certain words with their synonyms are both O(n).

The most expensive step is matching the paragraphs. This is potentially O(n^2) at the worst because each paragraph in Text A must be compared with each paragraph in Text B.

However the paragraph matching would only ever dominate the compute time if there were a very large number of paragraphs in both texts. In the best case (two texts with one paragraph each), the paragraph-matching would take place in O(n) time.

5.2.3 Limitations of the Method

There is a concern that the synonym-replacing step in SMPC could introduce some inaccuracy when dealing with figurative language.

For instance, the substitution of synonyms might turn "spill the beans" (meaning "to share a secret") into "drop the seeds", which means something very different. In texts with a lot of figurative language, this could cause false matches with other texts that use the same words in a non figurative way. In the example of "drop the seeds", it might tag it as similar to a text about farming.

However, the risk of this impacting a text-pair's score is low unless the whole of the text is written figuratively. This is because SMPC considers each paragraph of both texts. So even if one paragraph used a lot of figurative language, the distortion created would only affect the portion of the comparison involving *that particular paragraph.*

5.3 Method 2: Cosine Similarity

5.3.1 Explanation

The Cosine Similarity measure works off of the following idea: if two texts contain the same amounts of the same words, then they are the same.

We can easily model each text's word frequency as a vector, where each component represents the number of times that word appears in the text.

For instance, suppose we were comparing two texts:

- "foo foo bar"
- "foo bar bar"

Then we would generate two vectors. The first component would represent the count of "foo", the second would represent the count of "bar".

So, "foo foo bar" becomes $\langle 2, 1 \rangle$ (as it contains 2 "foo" and 1 "bar")
and "foo bar bar" becomes $\langle 1, 2 \rangle$ (as it contains 1 "foo" and 2 "bar")
This is plotted in figure 2.

It is immediately apparent that texts with similar sets of words will have more similar vectors (and so, vectors with a smaller angle between them). Two vectors will overlap if their texts contain all the same words. Two texts with no words in common will produce vectors that are at right angles to each other.

This angle, between $0°$ and $90°$, allows us to measure how similar the texts are.

We can normalize this measure to between 0 and 1 by looking at the cosine of the angle (because $\cos(0°) = 1$ and $\cos(90°) = 0$)

The cosine of the angle between two vectors is, luckily, uncomplicated. It is given as the dot product of the two vectors, divided by their lengths multiplied together. So:

$$\cos(\theta) = \frac{A \cdot B}{||A||\,||B||}$$

And this is the formula that the Cosine Similarity method uses to calculate a similarity score.

5.3.2 Time Complexity

Counting the word frequencies in both papers may be done in O(n). The vector multiplication involved in finding the cosine is also O(n).

So The Cosine Similarity Method should be of time complexity O(n) + O(n) = O(n).

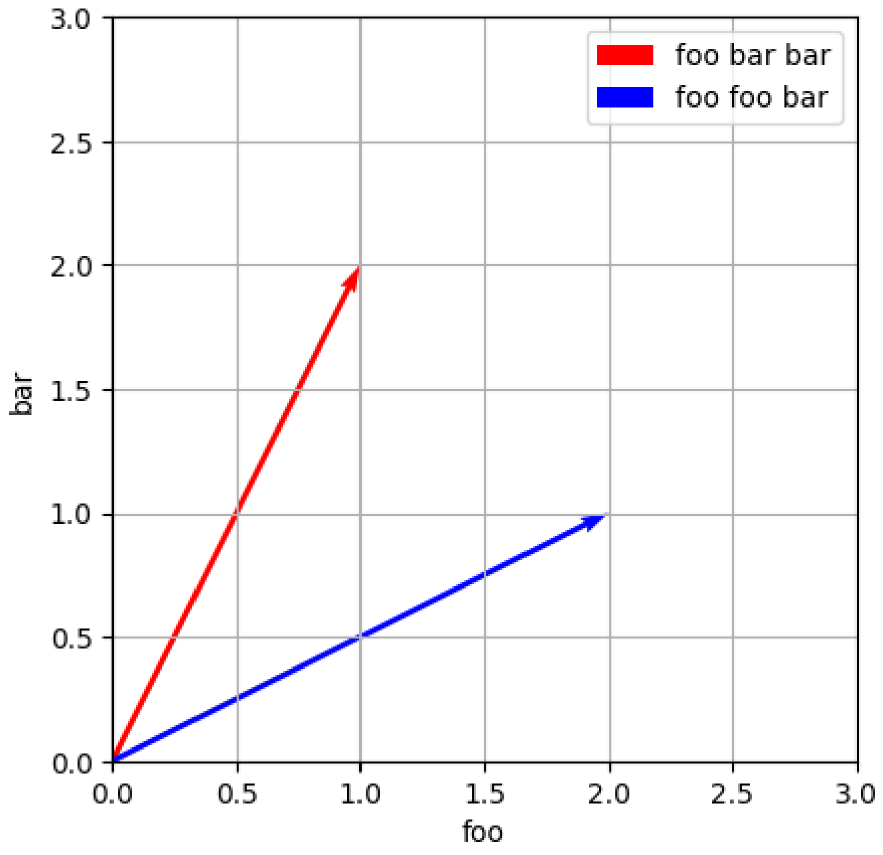


Figure 2: Word Frequency Vectors plotted for "foo foo bar" and "foo bar bar"

5.3.3 Limitations of the Method

Cosine is susceptible to marking opposite-meaning texts as similar if the words contained are the same. For example, "They think before they act" means the opposite as "They act before they think". But they have the same numbers of the same words in them, so Cosine would mark them as a perfect match.

5.4 Method 3: Fingerprint Method

5.4.1 Explanation

The fingerprint method works based on the idea that if two texts are similar, then they will contain similar sequences of characters within them.

It begins by extracting all of these distinct letter sequences. It cuts the text into a sequence of overlapping so-called 'n-grams'. These n-grams have a length k, which can be predetermined or chosen by the user. For this project, the k value will not be chosen by the user, so the predetermined length k = 4.

For instance, if the input text were "abcdefghijklmnopqrstuvwxyz", then the resulting n-grams of k = 4 would be:

$$["abcd", "bcde", "cdef", ..., "vwxy", "wxyz"]$$

This ensures that all sets of adjacent characters are well-represented.

These n-grams are hashed using MD5 (though any hashing algorithm that outputs an integer should work).

It would be possible to continue our analysis on the full set of hashes that we have just generated. However, for the sake of efficiency, this algorithm reduces the number of n-gram hashes in question. We remove all hashes that are not cleanly divisible by some number (Khuat et al. give an example of removing all hashes not divisible by 4 [KHTMH15])

In our implementation, we chose to remove all hashes not cleanly divisible by 3.

What hashes we have left from each text is called the "Set of Fingerprints". And again, each fingerprint represents some character sequence.

From here, the logic is straightforward:

When comparing document A and B, we want to look at how many of these distinctive character sequences are in both. So we look at the ratio between:

- The number of fingerprints that are common to documents A and B (multiplied by two, to account for the fact that there are two documents in question) and
- The number of total fingerprints, across both documents.

This ratio is called the "Dice Coefficient", and is the output of the algorithm.

In mathematical notation, the formula for this is:

$$Dice(A,B) = \frac{2|f(A) \cap f(B)|}{|f(A)| + |f(B)|}$$

If the Dice Coefficient is high (and so documents A and B have mostly the same letter sequences), then the documents are similar. If the Dice Coefficient is close to 0, then the documents share few letter combinations, and so the documents are dissimilar.

In short, the steps are as follows:

1. **Step 1: N-grams -** Generate all the sequences of characters contained in the text - these are the text's "n-grams". This is done for both texts.
2. **Step 2: Hashing -** Take both texts' sets of n-grams and hash each n-gram. This step is vital because it prepares us to be able to systematically remove some of them.
3. **Step 3: Selecting fingerprints -** Remove some of the hashes from consideration. Do this using a modulo operation. This reduces the size of the calculation that we will have to perform next.
4. **Step 4: Calculating the Dice Coefficient -** Calculate the Dice Coefficient, which effectively measures the similarity between the two texts. The Dice Coefficient is a ratio that measures how many 'fingerprints' (letter sequences) the two texts have in common in proportion to the sum of the fingerprints from both documents.

5.4.2 Time Complexity

The steps involved in the fingerprint method (i.e. n-gram generation, set generation & intersection) are all O(n), and so Fingerprint is O(n) as well.

5.4.3 Limitations of the Method

Fingerprint's logic revolves around spelling, so it's susceptible to false positives when opposite words are spelled similarly. For example, Fingerprint would mark "Emit" (to send out) as similar to "Omit" (to keep inside), because they are spelled similarly - even though they mean opposite things.

6 Challenges & Setbacks

While working on this project, we faced several major challenges & setbacks. We had to adapt and problem-solve during each step of the process.

One of the biggest challenges we faced while working on this project was the number of essays we were tasked with comparing. Once we finished implementing our code, we realized that running all comparisons would take an unrealistic amount of time, even for smaller sets of essays.

We explored a number of ways to speed up our program:

- We tried uploading the project to Google Colab, and running it there - thinking that this would be faster than running it locally in PyCharm.
- We tried looking at using a pre-built library for the cosine similarity method - thinking this would be faster than our custom-built implementation.
- We tried using different runtimes on the Colab notebook.
- We explored parallelization and matrix-based approaches to speed up the process. To this end, we created **language_project_parallel.py**. But even when we *could* make these optimizations (the matrix approach proved impossible), the performance of the system was still not sufficient.

Despite our best efforts, we could not cause our project to compare all 17307 essays. Due to constraints of computing power and time, we scaled the project down. The data given in this paper, then, is based on a run conducted on only 1000 essays.

The professor informed us that this smaller dataset was acceptable for submission.

Regardless of these setbacks, we do not see this project as a failure any more than a scientist would consider an inconclusive experiment a failure. Experiments are, by their nature, experimental. The outcomes aren't always what we hope for - but this is a good thing. Learning what doesn't work is just as important as achieving conclusive results.

7 Results

For this analysis, we ran our program twice:

1. First, we ran our program on a sample of the first 1000 essays from the dataset. We did so on Google Colab. The results from this were written to the file **output/similarity_results_500.csv**

2. Second, we ran the program with a sample of the first 500 essays from the dataset in Pycharm, on Mihir's personal quad-core Windows 10 laptop. This computer used an 11th Gen Intel i7-1165G Processor, at 2.7 GHz. It had 16.0 GB of RAM. The results from this were written to the file **output/similarity_results_1k.csv**

The general results are as follows:

In the run on 1000 essays, SMPC Method took a total time of 41.56 minutes; Cosine took 33.32 minutes; and Fingerprint took 44.28 minutes. So the total time spent computing was roughly 119 minutes.

In the run on 500 essays, SMPC took 6.5 minutes; Cosine took 4.2 minutes; and Fingerprint took 7.7 minutes. So the total time spent computing was approximately 18 minutes.

The second run was expectedly faster, for two reasons:

First, each pair ran faster because Mihir's computer was evidently more powerful than the computers on which Colab chose to run our first test.

And again, there are fewer comparisons to make:

$$\frac{1000(1000-1)}{2} - \frac{500(500-1)}{2} = 499500 - 124750 = 374750$$

So the run of 1000 essays would have had to test 274750 more pairs than the run of 500.

In both runs, Cosine was the fastest, and Fingerprint the slowest, with SMPC in the middle. In both runs, Fingerprint was the least accurate.

These results are discussed in more depth in the sections below.

7.1 Results for 1000 essays

7.1.1 Most Similar Essays for Method 1 on the Set of 1000 Essays

Table 1 shows the top 5 pairs for SMPC, along with the first sentence from each essay.

The SMPC method proved accurate at picking out messages whose meanings matched. Among its top 5 pairs, there were only 7 unique essays. All were messages arguing in favor of abolishing the Electoral College.

7.1.2 Technical Performance for Method 1 on the Set of 1000 Essays

In the discussion of the theoretical complexity of SMPC, we found that it could be, in the worst case, $O(n^2)$

In the dataset used in this project, the texts had relatively few paragraphs. For that reason, on this dataset, SMPC gave results that were approximately O(n).

Figure 3 plots the results of the trials for all essay pairs in the sample of 1000. The sizes of the inputs are plotted on the X, and the times taken are plotted on the Y.

It is visually apparent from the chart that the complexity was linear on this set of essays.

Somewhat surprisingly, however, was that the graph shows *multiple* different groups of linear patterns. This indicates that something was impacting the time performance other than the length of the essay. We originally suspected the following cause:

Different comparisons could have been made by different cores, and perhaps on different machines. Because we ran this on Google Colab, we had little knowledge of the

Table 1: Top 5 SMPC Similarity Pairs for the Subset of 1000 Essays

SCORE	ESSAY IDS	FIRST SENTENCES
1.00	**Essay A**: 02d481d **Essay B**: 04a99cb	**Essay A**: Dear, Senator The Electoral college has been around for centuries and as time changes, things start to evolve and grow along with the time period. **Essay B**: "It's hard to say this, but Bob Dole was right: Abolish the electoral college!" Dear Senator I strongly suggest you on changing to election by popular vote and getting rid of the Electoral College.
0.85	**Essay A**: 04a99cb **Essay B**: 639535	**Essay A**: "It's hard to say this, but Bob Dole was right: Abolish the electoral college!" Dear Senator I strongly suggest you on changing to election by popular vote and getting rid of the Electoral College. **Essay B**: We should not keep the Electoral college because; first, it's unfair, and also, it can quickly turn into a disater.
0.73	**Essay A**: 01ae85f **Essay B**: 02d481d	**Essay A**: Dear State Senator, I am feeling the need to change the Ellectoral College to election by popular votes for the president of the united states. **Essay B**: Dear, Senator The Electoral college has been around for centuries and as time changes, things start to evolve and grow along with the time period.
0.70	**Essay A**: 02d481d **Essay B**: 0a67e11	**Essay A**: Dear, Senator The Electoral college has been around for centuries and as time changes, things start to evolve and grow along with the time period. **Essay B**: Hello Senator of the state, I think we should change the Electoral College to the election by popular vote because whats really the point in having people vote for president if it doesn't contribute hardly any to which person gets elected? The Electoral College in highly unfair to voters simply because of the winner-take-all system.
0.70	**Essay A**: 02d481d **Essay B**: 0f81127	**Essay A**: Dear, Senator The Electoral college has been around for centuries and as time changes, things start to evolve and grow along with the time period. **Essay B**: It is often said that "change is good.

actual hardware running our code. Not all cores are created equal, and so performance would have differed between cores - but the results each core created would still have been linear. So we originally thought that each linear grouping on the chart is the work of one core, or one machine.

The second run of our program (described in section 7.2) later disproved this theory. This is discussed further in section 7.2.2.

SMPC performed much better on time-consuming essay-pairs than Cosine Method did. Specifically, Cosine had 2,388 essay pairs that took more than 0.15 seconds to process, whereas SMPC had only 1,381 essay pairs above this threshold.

On average, SMPC's performance was marginally faster than Fingerprint (by 0.0003 seconds per comparison), and somewhat slower than Cosine (by 0.001 seconds per comparison).

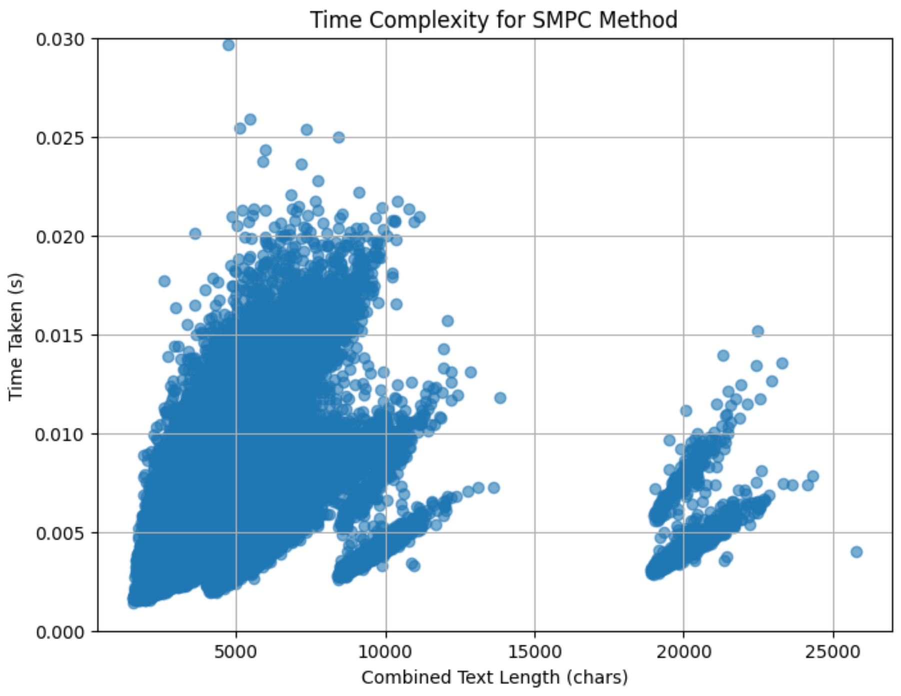


Figure 3: A plot of all the essay-pair comparisons made by the SMPC method for the dataset of 1000 essays

7.1.3 Most Similar Essays for Method 2 on the Set of 1000 Essays

Table 2 shows the top 5 pairs for Cosine, along with the first sentence from each essay. In terms of accuracy, it performed *remarkably* similarly to SMPC. Like SMPC, it found that the most similar essays were from the set of essays arguing against the electoral college. Of the 6 essays represented in SMPC's top ten pairs, and the 6 essays represented in Cosine's top ten pairs, 2 of those essays are the same.

7.1.4 Technical Performance for Method 2 on the Set of 1000 Essays

As expected, The Cosine Method performed as O(n), and the graph (Figure 4) shows linear performance.

7.1.5 Most Similar Essays for Method 3 on the Set of 1000 Essays

Table 3 shows the top 5 pairs for Fingerprint, along with the first sentence from each essay. It was good at finding essays that contained the same letter sequences (e.g. it picked up on two essays - one of which contained the phrase "Seagoing Cowboy" and the other, "sea cowboy"), it was surprisingly bad at grouping texts whose messages were actually similar.

The essay pair that Fingerprint graded as most similar are not similar at all. One argues that humans *should* explore Venus, and the other argues we *should not*.

Table 2: Top 5 Cosine Similarity Pairs for the Subset of 1000 Essays

SCORE	ESSAY IDS	FIRST SENTENCES
0.93	**Essay A**: 973664 **Essay B**: 0bf39e6	**Essay A**: The Electoral college should be abolished. **Essay B**: The electoral college is a process established in the constitution as a compromise between election of the president by a popular vote of qualified citizens.
0.91	**Essay A**: 077377d **Essay B**: 0f81127	**Essay A**: The Electoral College is a process, not a place. **Essay B**: It is often said that "change is good.
0.91	**Essay A**: 0bf39e6 **Essay B**: 0f81127	**Essay A**: The electoral college is a process established in the constitution as a compromise between election of the president by a popular vote of qualified citizens. **Essay B**: It is often said that "change is good.
0.91	**Essay A**: 077377d **Essay B**: 0c8f97b	**Essay A**: The Electoral College is a process, not a place. **Essay B**: The Electoral College is not a place, but a process.
0.91	**Essay A**: 0a67e11 **Essay B**: 0bf39e6	**Essay A**: Hello Senator of the state, I think we should change the Electoral College to the election by popular vote because whats really the point in having people vote for president if it doesn't contribute hardly any to which person gets elected? The Electoral College in highly unfair to voters simply because of the winner-take-all system. **Essay B**: The electoral college is a process established in the constitution as a compromise between election of the president by a popular vote of qualified citizens.

7.1.6 Technical Performance for Method 3 on the Set of 1000 Essays

Again, as expected, The Fingerprint Method performed as O(n), and the graph (Figure 5) shows linear performance.

7.2 Results for 500 Essays

7.2.1 Most Similar Essays for Method 1 on the Set of 500 Essays

Table 4 shows the top 5 pairs for SMPC from the run of 500.

The SMPC method was still accurate when it came to pairing essays based on meaning.

The top 5 pairs contained only 5 unique essays - likely a result of having fewer available essays than in the first run, which had 7 unique essays for the SMPC method's top 5 pairs.

Several of the top similarity scores were also relatively lower for the run of 500. In the run of 1000 essays, the lowest two similarity scores of the top five were both 0.70. However, for the 500 essays, the lowest scores of the top 5 were 0.65. This makes sense because, for both runs, the scores were normalized to a range between 0 and 1. So, the exact score of each pair will depend on what other pairs exist in the sample.

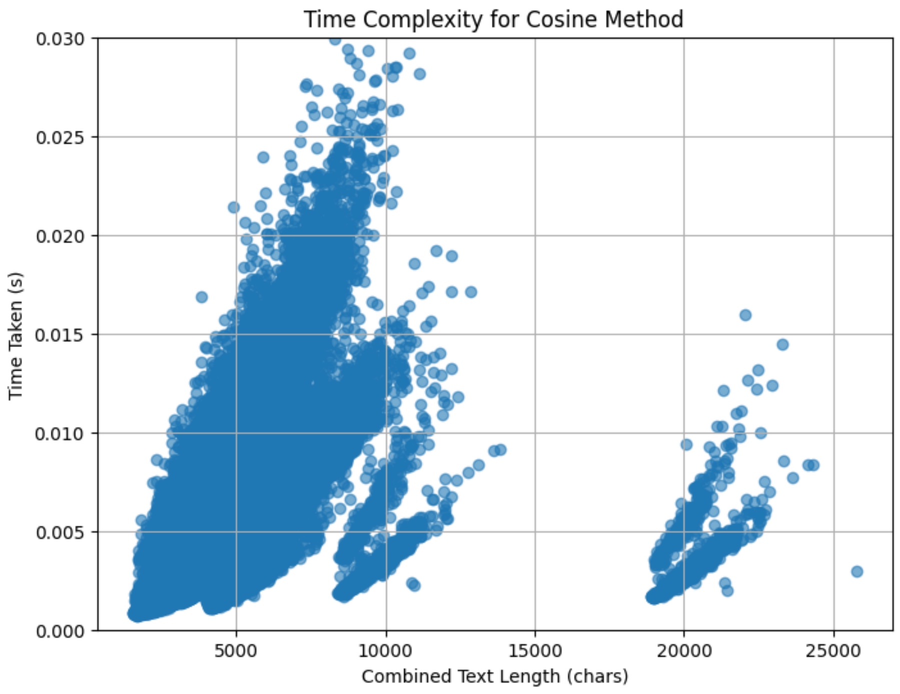


Figure 4: A plot of all the essay-pair comparisons made by the Cosine method for the dataset of 1000 essays

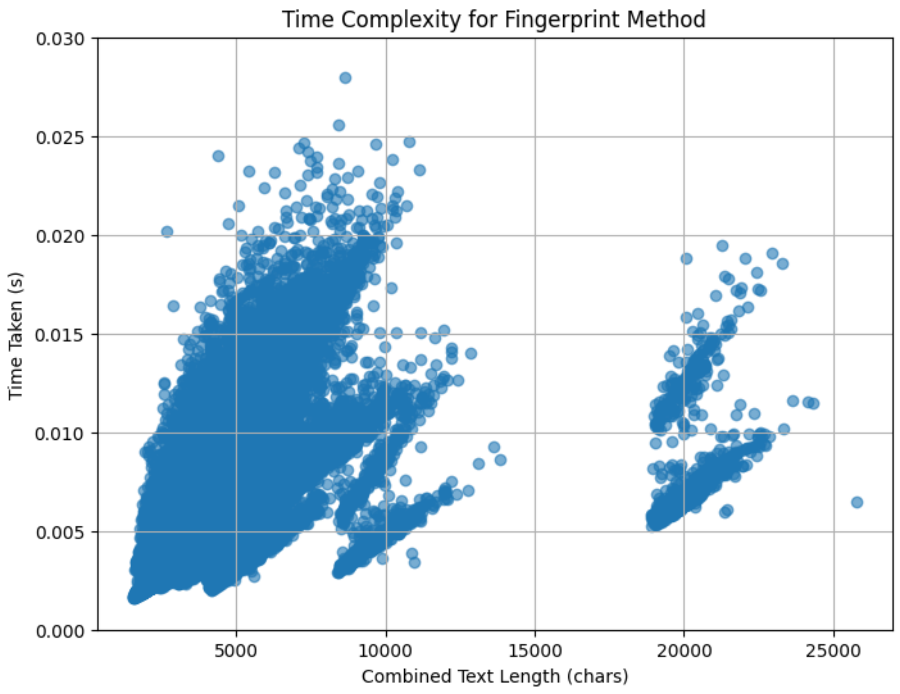


Figure 5: A plot of all the essay-pair comparisons made by the Fingerprint method for the dataset of 1000 essays

Like in the first run, all essays in the top five SMPC pairs argue for abolishing the Electoral College. Again, the algorithm seemed to have little drop-off in terms of accuracy.

Table 3: Top 5 Fingerprint Similarity Pairs for the Subset of 1000 Essays

SCORE	ESSAY IDS	FIRST SENTENCES
0.54	**Essay A**: 00a3575 **Essay B**: 0a93900	**Essay A**: The challege of exploring Venus is not a great idea. **Essay B**: Studying Venus is a worthy persuit despite the dangers because in the article it says that numerous spacecraft survived the landing for more than a few hours, maybe thiss issue explains why nit a single spaceship has touched down on Venus more than three decades, Numerous factors contribute to Venus's reputation as a challenging planet for humans to studydespite its proximity to us.
0.51	**Essay A**: 02f14c6 **Essay B**: 0a0d3b4	**Essay A**: The face is just a natrual landform. **Essay B**: The Face is a natural landform and not an alien monument.
0.50	**Essay A**: 00a3575 **Essay B**: 06cb1df	**Essay A**: The challege of exploring Venus is not a great idea. **Essay B**: Why is the planet Venus called Venus though? This story explains how the solar system is used.
0.48	**Essay A**: 909271 **Essay B**: 0dd48cd	**Essay A**: So there was a time when Luke signed up to be a sea cowboy because he said it was an opportunity of a lifetime to do this journy. **Essay B**: Seagoing Cowboys program is filled with adventures, and you could go to uniqe places.
0.48	**Essay A**: 06d6248 **Essay B**: 07a14a5	**Essay A**: Would you want to be burned at 800 degrees Fahrenheit? No, that just sound to painful you might say. **Essay B**: The author's claim of studying Venus is a worthy pursuit because Venus is closely related to Earth, Venus has a enviroment that is similar to Earth, and scientists want to explore more of what Venus has to offer.

7.2.2 Technical Performance for Method 1 on the Set of 500 Essays

As expected, The SMPC Method performed as O(n), and the graph (Figure 6) shows linear performance.

As in the first run, there are multiple linear groupings appearing on the graph. Again, something was affecting runtime other than input length. In our first run, we had explained this by attributing it to the specific hardware on which Colab had chosen to run our project.

Therefore, it was surprising to see that the same pattern appeared when running our program on Mihir's laptop. Each core on that laptop should be relatively similar - and so hardware could not explain the pattern.

It would be tempting to say that something structural about the essays themselves

Table 4: Top 5 SMPC Similarity Pairs for the Subset of 500 Essays

SCORE	ESSAY IDS	FIRST SENTENCES
1.00	**Essay A**: 02d481d **Essay B**: 04a99cb	**Essay A**: Dear, Senator The Electoral college has been around for centuries and as time changes, things start to evolve and grow along with the time period. **Essay B**: "It's hard to say this, but Bob Dole was right: Abolish the electoral college! Dear Senator I strongly suggest you on changing to election by popular vote and getting rid of the Electoral College."
0.85	**Essay A**: 04a99cb **Essay B**: 639535	**Essay A**: "It's hard to say this, but Bob Dole was right: Abolish the electoral college!" Dear Senator I strongly suggest you on changing to election by popular vote and getting rid of the Electoral College. **Essay B**: We should not keep the Electoral college because; first, it's unfair, and also, it can quickly turn into a disater.
0.73	**Essay A**: 01ae85f **Essay B**: 02d481d	**Essay A**: Dear State Senator, I am feeling the need to change the Ellectoral College to election by popular votes for the president of the united states. **Essay B**: Dear, Senator The Electoral college has been around for centuries and as time changes, things start to evolve and grow along with the time period.
0.65	**Essay A**: 02d481d **Essay B**: 639535	**Essay A**: Dear, Senator The Electoral college has been around for centuries and as time changes, things start to evolve and grow along with the time period. **Essay B**: We should not keep the Electoral college because; first, it's unfair, and also, it can quickly turn into a disater.
0.65	**Essay A**: 02d481d **Essay B**: 760842	**Essay A**: Dear, Senator The Electoral college has been around for centuries and as time changes, things start to evolve and grow along with the time period. **Essay B**: Many people think or believe that they should get rid of the Electoral College vote for the president.

could have caused this pattern. However, something like vocabulary choice could not have caused the pattern, because this would not have affected the Fingerprint Method - and the Fingerprint Method displays the same pattern.

We have no current theory as to why this pattern has occurred.

7.2.3 Most Similar Essays for Method 2 on the Set of 500 Essays

Table 5 shows the top 5 pairs for Cosine from the run of 500 essays. These 5 pairs represent 8 unique essays. The first 4 pairs discuss the electoral college while the fifth pair debates whether or not Venus should be explored.

In terms of the values reported as similarity scores, we see once more here that Cosine tends to report higher numbers than SMPC. That is, Cosine tended to produce higher

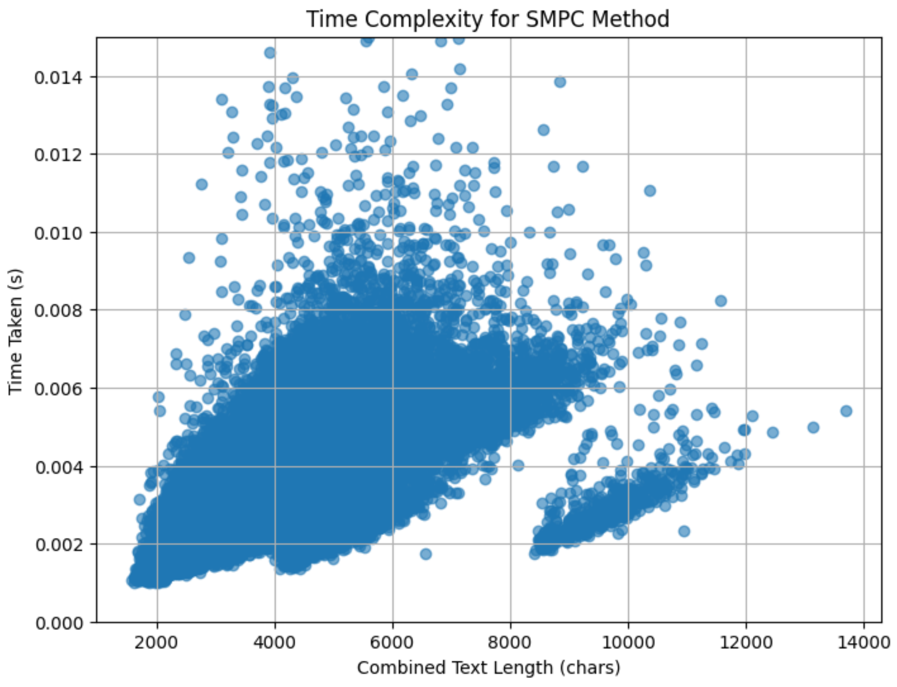


Figure 6: A plot of all the essay-pair comparisons made by the SMPC method for the dataset of 500 essays

valued similarity scores for its top-rated pairs. All pairs in Table 5 received scores greater than .89. However, while Cosine chose higher score values than SMPC for its top 5 pairs over both runs, this does not seem to reflect in accuracy. Both Cosine and SMPC seem to be of equal accuracy. Both produced accurate pairs in both runs.

7.2.4 Technical Performance for Method 2 on the Set of 500 Essays

As expected, The Cosine Similarity Method performed as O(n), and the graph (Figure 7) shows linear performance.

7.2.5 Most Similar Essays for Method 3 on the Set of 500 Essays

Table 6 shows the top 5 pairs for Fingerprint Method from the run of 500.

Like in the previous run, it falsely tagged one pair about Venus to be similar even though their tones are very different. Regardless, the rest of the pairs do seem similar when reading their full texts (though the first sentences shown in the table may not always reflect that).

Unlike Cosine, Fingerprint does not seem to have degraded in terms of accuracy when the sample size was reduced. This seems to indicate that Fingerprint is *more robust*, and less sensitive to changes in the quality of input data than Cosine.

7.2.6 Technical Performance for Method 3 on the Set of 500 Essays

The Fingerprint Method performed as O(n), and the graph (Figure 8) shows linear performance.

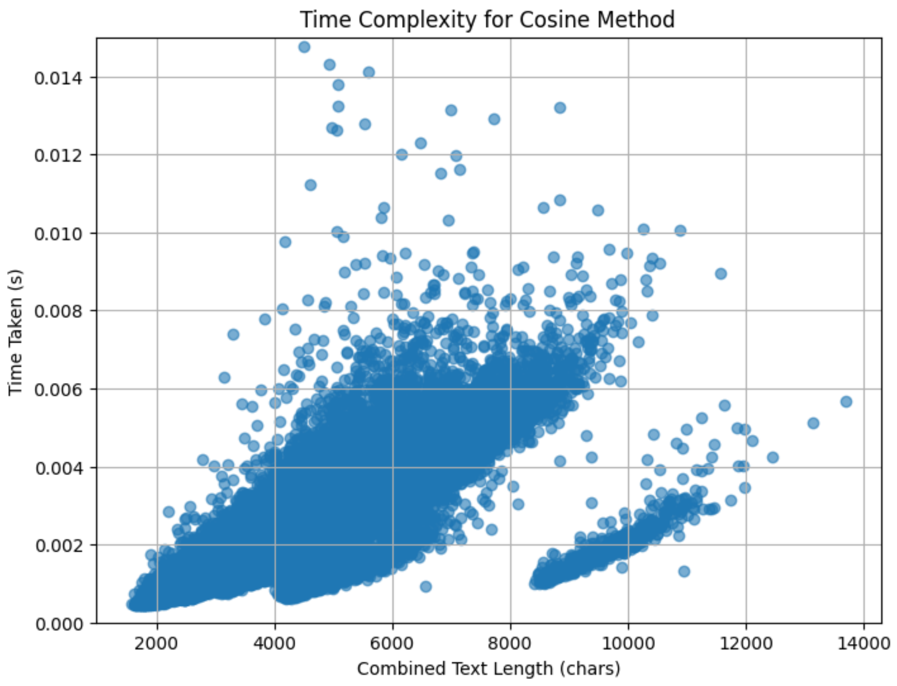


Figure 7: A plot of all the essay-pair comparisons made by the Cosine method for the dataset of 500 essays

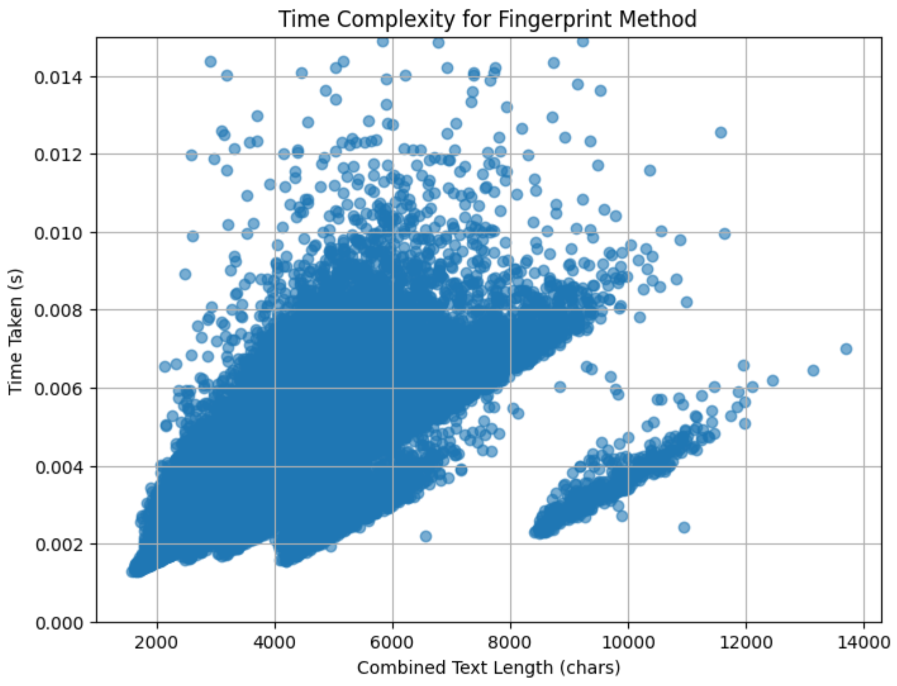


Figure 8: A plot of all the essay-pair comparisons made by the Fingerprint method for the dataset of 500 essays

Table 5: Top 5 Cosine Similarity Pairs for the Subset of 500 Essays

SCORE	ESSAY IDS	FIRST SENTENCES
0.91	**Essay A**: 01ec06b **Essay B**: 0232fa2	**Essay A**: Dear State Senator , The electoral college is not a good idea. **Essay B**: Dear State senator , You should keep the electoral college , beacuse For one theres alot of electors like around 538.
0.90	**Essay A**: 01f5682 **Essay B**: 760842	**Essay A**: Dear State Senate, I feel that the way we, the people, vote today is very unfair. **Essay B**: Many people think or believe that they should get rid of the Electoral College vote for the president.
0.90	**Essay A**: 0232fa2 **Essay B**: 792205	**Essay A**: Dear State senator , You should keep the electoral college , beacuse For one theres alot of electors like around 538. **Essay B**: Dear State Senator, I think that the Electoral College is important becase it is a process.
0.89	**Essay A**: 01f5682 **Essay B**: 02d481d	**Essay A**: Dear State Senate, I feel that the way we, the people, vote today is very unfair. **Essay B**: Dear, Senator The Electoral college has been around for centuries and as time changes, things start to evolve and grow along with the time period.
0.89	**Essay A**: 123792 **Essay B**: 04f47a5	**Essay A**: In "The Challenge of Exploring Venus," the debate of the exploration of Venus is exposed for discussion. **Essay B**: In "The Challenge of Exploring Venus," the author includes many benefits and dangers that come with exploring Venus.

Table 6: Top 5 Fingerprint Similarity Pairs for the Subset of 500 Essays

SCORE	ESSAY IDS	FIRST SENTENCES
0.50	**Essay A**: 00a3575 **Essay B**: 06cb1df	**Essay A**: The challege of exploring Venus is not a great idea. **Essay B**: Why is the planet Venus called Venus though? This story explains how the solar system is used.
0.48	**Essay A**: 06d6248 **Essay B**: 07a14a5	**Essay A**: Would you want to be burned at 800 degrees Fahrenheit? No, that just sound to painful you might say. **Essay B**: The author's claim of studying Venus is a worthy pursuit because Venus is closely related to Earth, Venus has a enviroment that is similar to Earth, and scientists want to explore more of what Venus has to offer.
0.48	**Essay A**: 02f14c6 **Essay B**: 07e4b76	**Essay A**: The face is just a natrual landform. **Essay B**: Do you think that the "Face on Mars" is not a real face but just a natural landform? Well, we think it is just a naural landform because when the first image appeared on JPL web site, it rereal that it's just a natural landform.
0.46	**Essay A**: 027a1f9 **Essay B**: 06be122	**Essay A**: The face was just a natural landform made by butte or mesa which were landforms common around around the American West. **Essay B**: the face wasn't formed by aliens.
0.46	**Essay A**: 06be122 **Essay B**: 07e4b76	**Essay A**: the face wasn't formed by aliens. **Essay B**: Do you think that the "Face on Mars" is not a real face but just a natural landform? Well, we think it is just a naural landform because when the first image appeared on JPL web site, it rereal that it's just a natural landform.

8 Conclusion

This project explored the use of Fingerprint and Cosine methods of text similarity detection. It also explored the use of a novel method, SMPC.

Of the three, Fingerprint performed the poorest in both time and accuracy.

SMPC performed on-par with Cosine in accuracy; both quite accurately picked out essays with the same meaning.

8.1 Future Work

In the future, we plan to optimize our implementation of the Cosine Similarity Method. Several of our classmates were able to compare all 17307 essays from the Database in under 3 minutes. They were able to do so by using partial matrix multiplication in order to find the cosine of all vector pairs, all at once. When we learned this, We intended to attempt something similar, but we could not get it to work. We still intend to come back to this in the future, as a challenge.

Future work could also be done on optimizing these methods for speed or combining them for better accuracy.

References

[Kag24] Kaggle. Learning agency lab: Automated essay scoring 2. https://www.kaggle.com/competitions/learning-agency-lab-automated-essay-scoring-2/data, 2024. Accessed: 2024-10-22.

[KHTMH15] Tung Khuat, Nguyen Hung, and Le Thi My Hanh. A comparison of algorithms used to measure the similarity between two documents. *International Journal of Advanced Research in Computer Engineering Technology (IJARCET)*, 4:1117–1121, 04 2015.

[Ope14] OpenStax. *U.S. History*. OpenStax, updated edition, 2014. Accessed: 2024-10-22.

[Oxf11] Oxford Dictionaries. The oec: Facts about the language. https://web.archive.org/web/20111226085859/http://oxforddictionaries.com/words/the-oec-facts-about-the-language, 2011. Accessed: 2024-10-24.

[RN20] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Pearson, 4th edition, 2020.

[SSAS24] V Shanmathi, K Sowmiya, SH Aishwarya, and M Sakthivel. Document plagiarism checker based on similarity. *International Journal of Research Publication and Reviews*, 5(6):2992–2996, 2024.

[Tat17] Rachael Tatman. English word frequency. https://www.kaggle.com/datasets/rtatman/english-word-frequency, 2017. Accessed: 2024-10-24.

9 Links to ChatGPT Conversations

- https://chatgpt.com/share/67181891-ecc8-8010-86d3-e0eec64e4ac0
- https://chatgpt.com/share/6715ba45-0a08-8008-9449-e42bcfa62fcd
- https://chatgpt.com/share/6711ca0d-8900-800e-becf-99322eae632d
- https://chatgpt.com/share/6716ddbd-e468-800e-87e9-0e976e82130d