Cosine Similarity Based Plagiarism Detector

Evaluation and Improvements

Catherine Yu, Juliette Mangon, Martin Lapczyk, Walker Tupman

Project Implementation

https://github.com/catherineyu2014/NLP-Plagiarism-Detector

Abstract

The reliance on automatic plagiarism detection software has increased in academia, but the convenience of using this software often comes at a cost—a high rate of false positives. We decided to write a plagiarism detection program with the express focus of ensuring a low rate of false positives. We implemented existing research on plagiarism detection techniques which we had found in several scholarly articles, primarily experimenting with TF-IDF scores, cosine similarity, Jaccard similarity, and n-gram classes. Once we replicated the results stated in the published articles, we improved the model by using multiprocessors to decrease our runtime and changed the computation technique for our TF-IDF scores to improve our F-score. Our model's runtime decreased from 480 seconds to 98 seconds for our training corpus, and we improved the maximum F-score in our training corpus from 72.73% to 73.68%. The results on our larger evaluation corpus were not as successful as we would have hoped, with a continued improvement in runtime but a significantly worse maximum F-score of 62.07%.

Introduction

Plagiarism is the act of copying work done by another person without citation. Recently, there has been an increase in the creation and usage of plagiarism detection programs, and while many are successful, there are often high rates of false positives that are flagged. Our goal was to write a plagiarism detection program that improved prior research while lowering the rate of false positives.

In Part I of our project, we focused on comparing potential plagiarism detection methods to determine the model that produces the highest accuracy rate. This rate is measured by F-scores. F-scores are a harmonic mean of a given model's precision and recall. We implemented methods discussed in published academic articles and examined their effects on accuracy rates. We specifically focused on the utilization of different n-gram classes, cosine similarity, and Jaccard similarity. After we selected the best combination of techniques, we improved on these techniques in Part II by rewriting the selected method to increase accuracy and decrease runtime.

Related Work

Jahan et al. presents research that compares Jaccard and cosine similarity models using unigrams, bigrams, and trigrams. This paper proposes that using a trigram model with cosine similarity detects plagiarism with the highest degree of accuracy. In Part I, we replicate their results using the PAN-PC-11 corpus. PAN-PC-11 has been cited in multiple academic papers on plagiarism detection, including Bensalem et al. and Davoodifard's work.

Part I — Evaluating Existing Plagiarism Detection Methods

We implemented the research by Jahan et al. on the PAN-PC-11 corpus. The PAN-PC-11 corpus was split into two sections – external and intrinsic plagiarism. External plagiarism detection involves comparing suspicious documents to source documents to find if content has been copied from a source document into a suspicious one. This is detected using direct text matching and text similarity measures. Intrinsic plagiarism detection focuses on suspicious texts without a reference or source document for comparison. Detection is based on internal features of the text, including changes in sentence structures or writing styles. We focused solely on external plagiarism detection, using similarity measures to detect plagiarized documents.

The PAN-PC-11 corpus contains suspicious documents that may contain plagiarism from source documents. Each suspicious text document has a corresponding XML file that indicates if there is plagiarism and which source documents the plagiarism originated from. The corpus has two directories: one with source documents and one with suspicious documents. Each folder has 21 "parts" or subdirectories each with 500 text files and their corresponding XML files inside. Plagiarism in the suspicious documents can be copied from any of the 10,500 source documents. Therefore, checking all suspicious documents would be incredibly computationally expensive, requiring $10{,}500^2$ comparisons. To solve this computational issue, we created three corpora: training, testing, and evaluation sets, each with 60% non-plagiarism cases and 40% plagiarism cases. Our training corpus had 20 suspicious documents and 400 source documents, our testing corpus had 40 suspicious documents and 800 source documents, and our evaluation corpus had 80 suspicious documents and 1,600 source documents. These corpora were not made with an algorithm, so there is a possibility that the 60/40 ratio was not exact due to human error, especially on the larger corpora. Additionally, the amount of plagiarism per suspicious document in the PAN-PC-11 varied, and was not clearly labeled. Since we sourced PAN-PC-11 to create our own corpora, we were unable to standardize the plagiarism amount within and across corpora. The possible human error and lack of a standardized plagiarism amount per document should be considered when looking at our results as these variables may have caused discrepancies in our data.

Subsequently, we needed a program to score our plagiarism detector against our new corpora. We created an answer key program that scores metrics for our program, specifically looking at accuracy and F-score. We used this program throughout our project to score our system as we improved it.

To calculate the accuracy and F-score, we explored the threshold of similarity between a suspicious document and a source document that would deem a suspicious document to be plagiarized. We first created a script to output similarity scores for a given corpus. A similarity score of 0.0000 means there was no similarity found; however, cosine similarity often creates scores that are greater than 0.0000 when comparing long texts like the ones in our corpora, so our threshold value could not be 0.0000. Lower thresholds result in higher false positives, while a higher threshold lowers instances of false positives. Due to the serious consequences of a false plagiarism accusation, we wanted to maximize our similarity threshold to lower false positives while also maximizing our F-scores. In order to evaluate our corpora in a standardized way, we created a function that compares n-grams (1-12) against cosine and Jaccard similarity scores using different similarity thresholds and outputs their corresponding F-scores into a graphical heatmap – see figures 1 - 4 below.

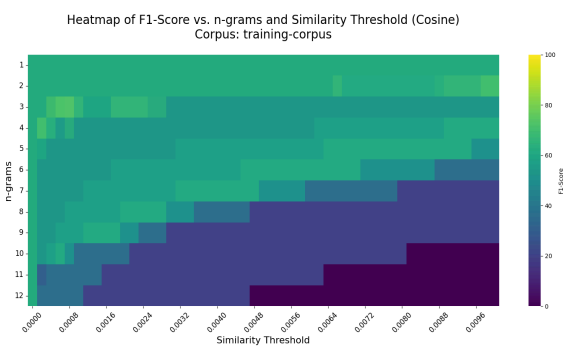

Figure 1: Cosine Similarity for Training Corpus

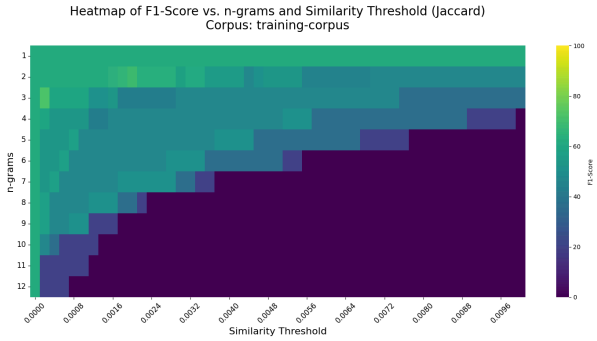

Figure 2: Jaccard Similarity for Training Corpus

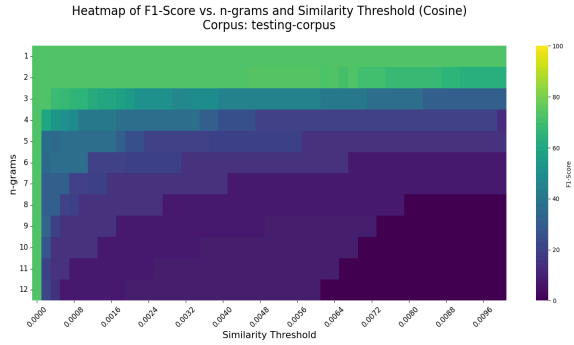

Figure 3: Cosine Similarity for Testing Corpus

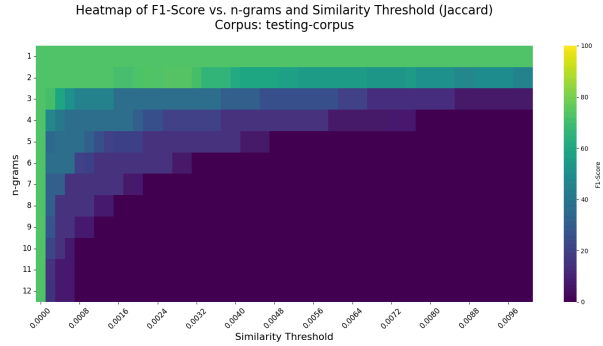

Figure 4: Jaccard Similarity for Testing Corpus

We discovered that cosine similarity is better than Jaccard similarity in almost all n-gram ranges. A combination of cosine similarity and trigrams produced the highest F-scores, reproducing the results of the paper by Jahan et al – see figures 1 and 3 above. At high n-gram chunks, vectors become increasingly sparse. Cosine similarity works better than Jaccard similarity when given sparse vectors, providing better results at higher n-gram ranges. Furthermore, higher n-grams produce less false positives, providing another advantage for cosine similarity. Our corpus consists of excerpts from novels which typically have long sentences, so using higher n-gram ranges was a viable option for our plagiarism checker and produced less false positives.

Part II — F-score and Time Improvements

After evaluating existing plagiarism methods, we aimed to improve both the efficiency and accuracy of a cosine similarity and trigram based plagiarism detector.

We first improved the efficiency of our program by implementing multiprocessing to compute the TF-IDF and similarity scores for each n-gram with multiple processes. This allowed us to generate similarity scores for each n-gram simultaneously, which improved our runtime.

To fully utilize our improvements, we also used the CIMS servers, specifically *crunchy1.cims.nyu.edu*. With the multiprocessing features implemented, the runtime of our program for the training set decreased from 480 seconds to 98 seconds – an 80% decrease. We ran into issues with larger corpora taking up far more memory per n-gram, requiring us to lower the amount of processes running simultaneously to lower

our memory usage. When memory is filled, our program began utilizing the disk more often which severely affected performance. This issue was less noticeable on higher memory units like *crunchy1.cims.nyu.edu*, which has 256 gigabytes of memory.

We attempted to increase our F-score on our training set by measuring the F-score of cosine similarity and trigrams on similarity thresholds between 0.0000 and 0.0100. Our goal was to get an F-score higher than the previous max F-score – 72.73%. Our previous max F-score resulted from a combination of cosine similarity, trigram, and a similarity threshold of 0.0008.

Since we implemented already existing plagiarism detectors from academic journals, it was challenging to find additional techniques to improve F-scores. Initially, we tried to implement different libraries from the NLTK toolkit such as WordNetLemmatizer and PorterStemmer, but rather than exceeding an F-score of 72.73%, all of the highest F-scores decreased to 62.07% These methods were likely unsuccessful because we are using corpora with external plagiarism, where excerpts of the source documents in our corpus are copied into the suspicious documents. Thus, lemmatizing words would not improve our accuracy as this technique would even out the similarity amongst all documents, leading to a smaller standard deviation for the F-scores, rather than increasing our max F-score. We suspect that these methods may work better for plagiarism checkers looking for intrinsic plagiarism.

Next, we changed the parameters in the vectorizer lines, but the results did not improve. In the methods we tested, all of the F-scores leveled out to 62.07%. We changed other parts of the code, which resulted in either a decrease in the F-scores or did not change them. Since the corpus we tested on was small due to runtime constraints, the 62.07% F-score could have been a common percentage for the F-score fraction to come out to; however, we were unable to verify if that was the reason for this leveling phenomenon.

Finally, we ended up changing the TF-IDF vector calculations by not removing stop words. Following the logic we used to deduce why the NLTK toolkit attempts did not work, we realized that there was no reason to remove stop words since that would only decrease the similarity between source and corresponding suspicious documents. Following this change, our highest F-score was 73.68% for cosine similarity with trigrams with a similarity threshold of 0.0038 on our training corpus – see figure 5 below.

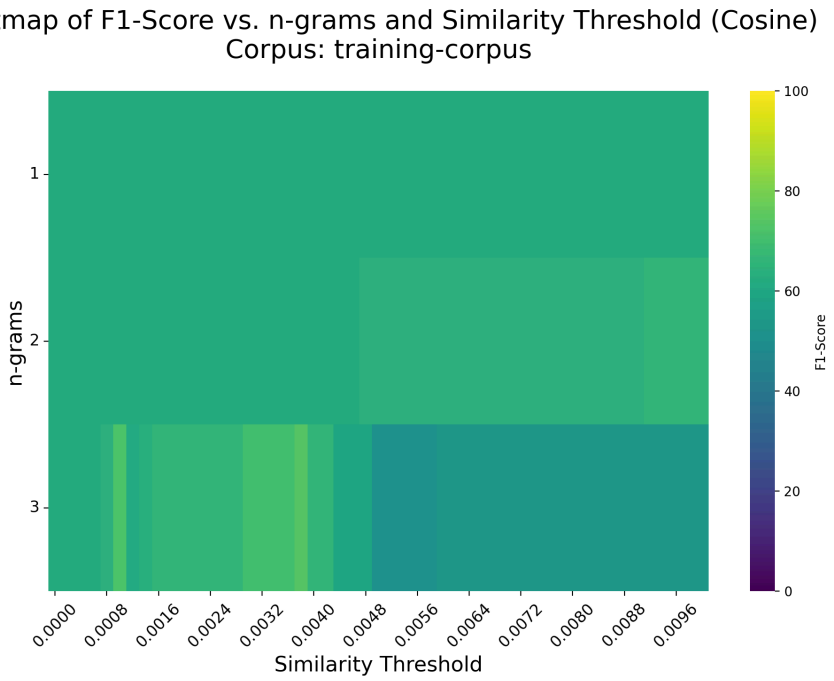

Figure 5: Unigram, Bigram, Trigram Cosine Similarity for Training Corpus (After Change)

However, when we tested this technique on our evaluation corpus, we did not see the same results. In fact, retaining stop words in our evaluation corpus actually lowered our max F-score, but made our trigram model more viable up to higher similarity thresholds. Retaining stop words increased our recall, due to the higher average cosine similarity scores, but lowered our precision – see figures 6 and 7 below.

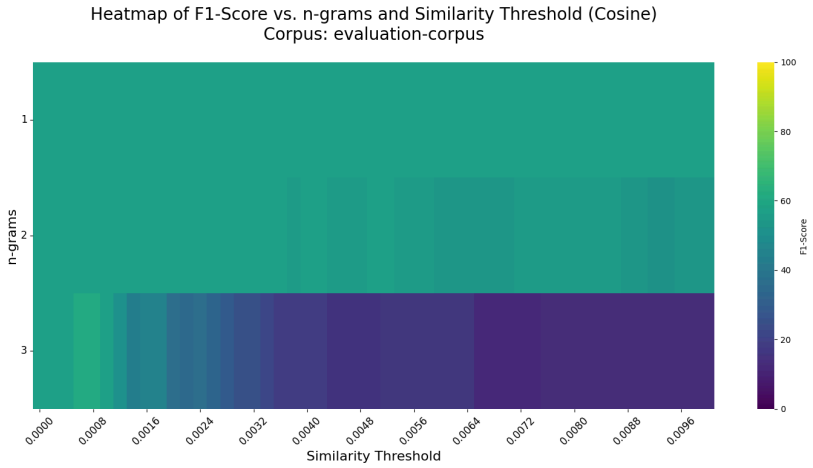

Figure 6: Unigram, Bigram, Trigram Cosine Similarity for Evaluation Corpus (Before Change)

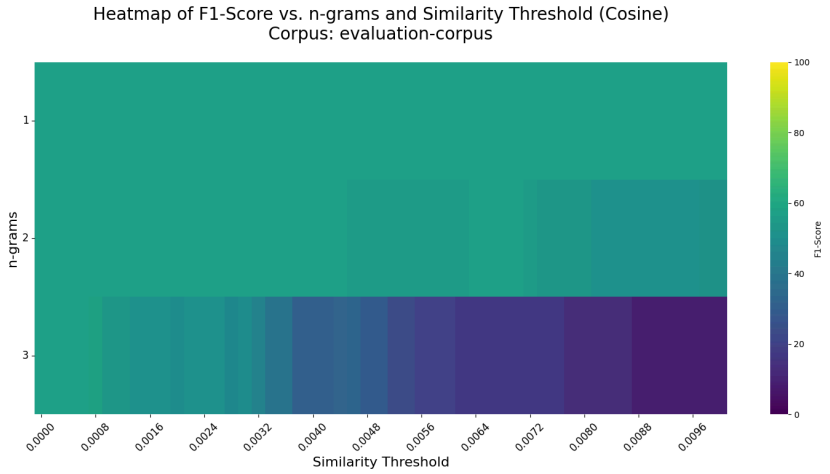

Figure 7: Unigram, Bigram, Trigram Cosine Similarity for Evaluation Corpus (After Change)

Unfortunately, due to the sheer amount of time that it took to run our algorithm on the evaluation corpus, we were unable to test other alterations of our algorithm on it, or try any debugging with our final algorithm to see if there was a distinct reason for such strikingly different results. Our new algorithm which retained stop words could have created more false positives or our evaluation corpus could have had more stop words in the documents simply because of the nature of the randomly selected corpus. This could have led to lower F-scores, but in theory, these false positives would have been canceled out when using a higher similarity threshold value. Our results across different corpora were inconsistent throughout the project, potentially due to inconsistencies when creating them. This could explain why our results were different for the evaluation corpus. Using much larger corpora could smooth out these differences and allow for more precise testing. Our results for the full 1-12 n-gram range follow the same pattern as the trigram heatmaps of a smaller standard deviation across F-scores – see figures 8 and 9 below.

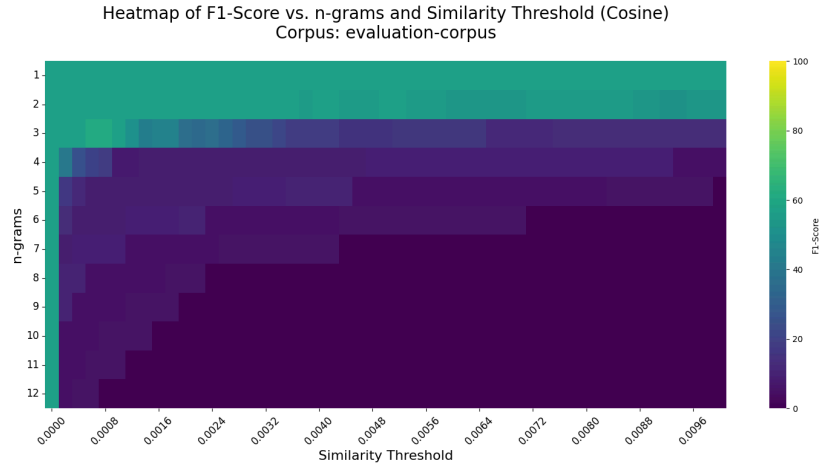

Figure 8: All N-Grams Cosine Similarity for Evaluation Corpus (Before Change)

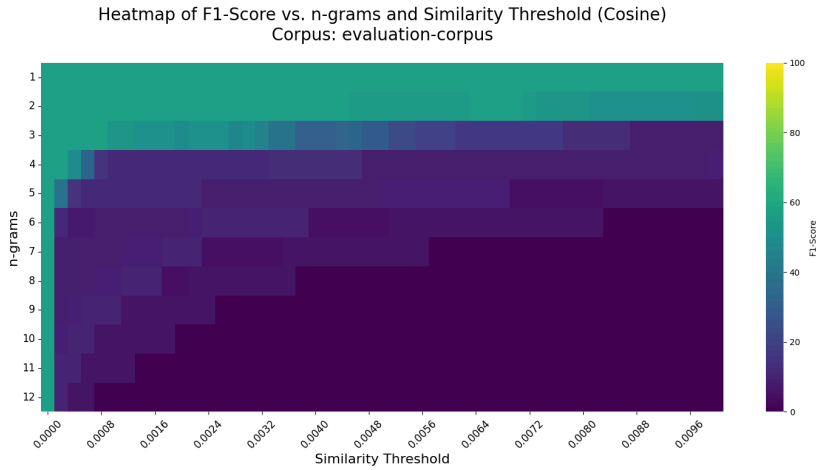

Figure 9: All N-Grams Cosine Similarity for Evaluation Corpus (After Change)

Conclusion

We evaluated and enhanced an existing external plagiarism detection technique using TF-IDF vectors, cosine similarity, Jaccard similarity, and n-gram models. Through rigorous experimentation with existing methodologies, we verified that cosine similarity combined with trigrams outperforms other approaches. Our efforts to optimize the detection process by introducing multiprocessing significantly reduced runtime, enabling our system to handle large corpora. Furthermore, adjustments to the TF-IDF vectorizer, particularly retaining stop words, resulted in a notable improvement in the F-score on the training corpus, although the results across different corpora were inconsistent. These discrepancies highlight the need for more robust testing frameworks and much larger corpora to generalize findings.

Despite these limitations, our improvements demonstrate the potential for enhancing existing plagiarism detection models in both accuracy and efficiency. Future work should explore additional preprocessing techniques and their impact on diverse datasets to further refine detection capabilities and reduce false positives. Through combining increased computational efficiency with refined accuracy metrics, our study lays a foundation for developing more reliable plagiarism detection tools tailored to academic and professional applications.

Works Cited

Bensalem, Imene, et al. *Intrinsic Plagiarism Detection Using N-Gram Classes*, aclanthology.org/D14-1153.pdf.

Davoodifar, Mahshad. *Automatic Detection of Plagiarism in Writing*, files.eric.ed.gov/fulltext/EJ1340053.pdf.

Jiffriya, et al. *Plagiarism Detection on Electronic Text Based Assignments Using Vector Space Model*, arxiv.org/pdf/1412.7782.