**Preprocessing**: A minimal amount of pre-processing was needed on the documents. Introductory statements needed to be removed for exact duplicates – the plagiarizer’s name differed from the original author. I considered removing speech actions (laughter, applause, etc), but decided that did not meet the criteria for exact matches, and would be detected by the similarity detection. In all cases, punctuation was removed. Finally, text tags like ‘\n’ for newline were removed and replaced with a space (and multiple spaces were replaced with single ones for consistency).

**Comparison**: Fingerprinting, using two different methods, was used for comparison. For the exact matches, I used the adler32 algorithm, because the comparison (exact match of values) did not warrant complexity. Iterating over the list of files, if there was a collision of the fingerprint, it was assumed an exact duplicate. If not, I just added it to the hash map.

For similarity detection, I used SimHash with an md5 hashing argument. Md5, with a size of 128 bits, represents a more versatile view of the possibilities a document can take. I converted each word into 128 bits, summed their weighted columns and converted the final value into binary.

Once I had a text document’s binary fingerprint, I needed to discover all subsections where plagiarized text occurred. To do so, I subdivided the 128 md5 into 16 subsections of 8 bits long. 8 bits represents 256 different combinations, which reduces the amount of false positives, while only representing a 1/16th of the document. This grants sixteen chances to identify a plagiarized document. If we were searching for finer levels of plagiarism (i.e. minor subsections), we would want to reduce this number to 4, and deal with the increased false positive rate.

After hashing the matches to a subsection of the fingerprint, I needed to compare the values. Since our representation was already in bit form, I decided to use the Hamming Distance. I chose 10 after a number of iterations on the training data – it is a large enough value to detect a number of changes (i.e. 5 misses a near-document), but small enough not to classify everything as plagiarized (30 detects far too much). Alternative methods considered were using tf-idf to rank documents (which would require further iterations through the document set to find the required statistics) and comparing subsections of the documents (taking subsets of the documents and comparing their equivalents using set theory/Cosine coefficient). Both would result in a finer tuned detectors, but I did not have time to test these implementations.

**Finn’s Method:** I implemented Finn’s method using the algorithm described in the notes. This involved iterating over a preprocessed document’s words, and incrementing/decrementing the values L, M and R depending on if the current word was a number or not. Every time the maximum value of L+M+R was larger than the global maximum, I stored the values of a and b. After discovering the maximum values of a and b, I used the fingerprinting method for similarity detection on this subset of the document (containing the plateau). Then, after all plateaus had been found, I compared them using the Cosine Coefficient method. This was chosen because word weights would have little bearing on a subset of data (most numbers and words would only appear once). Just to be safe, I used the fingerprinting method of the similarity detection to check the document subsets as well. If a document appeared using the fingerprinting method, I would also add it to ‘finn.txt’.

When implementing Finn’s method, multiple instances would occur where the algorithm did not detect a plateau. This would result in a and b values that differed by one. In testing, these seemed to always be alphabetic examples, but in the case of a digit appearing, the duplication of a single number is not relevant. To improve its implementation, I discarded the values where the a and b values were too small.

**Statistics:** Duplicates detected all were of minor variations – single words/paragraphs added or subtracted. This is the result of considering the entire document when comparing the hamming distance. Only blatant plagiarism demostrations would be caught in this manner. A refined technique would use a metric that allows for subsections of documents (tf-idf, or a weighted vector representation) and would be applied to these subsets.

Finn’s method also assumes that there is only one section of numerical interest (a single plateau). If there are multiple places where numbers have been copied from, Finn’s method would only detect the largest subsection (and subsequently only compare it).