Plagiarism Detection in Natural Language

Jeffrey Bagdis

Advised by Brian Kernighan

5 May 2008

A senior thesis submitted to the Computer Science Department of Princeton University in partial fulfillment of the requirements for the degree of Bachelor of Science in Engineering.

It is with deepest gratitude that I thank my advisor, Professor Brian Kernighan, for his thoughtful advice, patient guidance, and indefatigable humor as I struggled through the process of writing this thesis.

Without him I would have been lost a thousand times.

For Kim

Contents

Chapter 1

Plagiarism

1.1 Plagiarism Defined

Any thorough discussion of the identification of plagiarism must first begin with a thorough definition. As easily as that is said, however, it is not as easily accomplished. Every student from elementary school onward has heard about plagiarism. Schools and universities alike look very unfavorably on this serious breach of academic integrity. Many have adopted zero-tolerance policies – punishing infractions with failure, suspension, or, in many cases, even expulsion. Many consider plagiarism to be among the most serious acts of academic dishonesty – on par with cheating on a final examination [1]. The transgression is not solely confined to the ivory tower of academia, either. Plagiarism is a severe violation of journalistic ethics, and many reporters and columnists have been summarily fired for the offense [3]. Indeed, any profession or endeavor in which a creative synthesis of ideas, in written form or otherwise, plays a role has been touched by the crime of plagiarism.

Despite the universal nature of the offense, and the extent to which students of all ages are implored to avoid it, plagiarism remains rather murkily defined in the minds of many, including a number of upstanding intellectuals. Despite the preponderance of literature attempting to elucidate the subject, for some who should certainly know better the specific acts which might constitute plagiarism remain (allegedly) a mystery left to the probation of empiricism. Take the case of Senator Joseph Biden, Jr. (Senior Senator from Delaware, at the time of this writing) for example. While an law student at Syracuse University, Senator Biden plagiarized 5 pages of a 15-page paper from a Fordham Law Review article. When his plagiarism was discovered to the faculty, Senator Biden claimed that he had simply misunderstood the rules of attribution, and had not intended to deceive anyone. Ultimately, he was given a failing grade for the course, which was stricken from his record when he retook that course the following year [2].

Whether or not Senator Biden willfully plagiarized that law-school paper, one must accept the possibility that he was indeed ignorant of the proper practices of academic integrity. This sad episode should make the necessity of a formal definition of plagiarism all the more clear. It does nothing, however, to further one toward that end. How, then, should plagiarism be defined? It seems that every university worth its salt has published some form of pamphlet or brochure imploring its students to follow some manner of guidelines for academic integrity. These guidelines tend to differ slightly in the details of their description, and more than slightly in the details of their enforcement, but they all tend to share some overarching tenets that drive to the heart of a concept of plagiarism. The description of plagiarism written by Dr. Earl Babbie (Campbell Professor Emeritus in Behavioral Sciences at Chapman University) [1], which seems reasonably straightforward and commensurate with descriptions from many other sources, is used as a basis for the following discussion.

Plagiarism, at its root, is academic dishonesty. It is stealing the ideas of another and claiming them as one's own. Plagiarism can occur in any form of human communication. This discussion, however, is primarily concerned with instances of plagiarism in written documents. Thus, a more narrow definition of plagiarism, solely concerning the written word, must be developed to facilitate that discussion. First, however, a brief word of warning. The definition sought herein is sought principally for the purpose of discussing the detection of written plagiarism. Thus, it shall ideally describe not the ways that one might inadvertently (or purposefully) commit plagiarism, with a mind toward preventing or reducing academic dishonesty, but rather shall attempt to parameterize the entire gamut of ways in which plagiarism might occur, with a mind toward quantifying particular instances of plagiarism pursuant to discussing their detectability by automated processes.

Written plagiarism, however, suffers a basic definition no more narrow than does plagiarism in general. It is, fundamentally, stealing the ideas of another and claiming them as one's own within a written document. Practically, however, there are only so many ways in which this dishonesty can be accomplished. Specifically, the domain of written plagiarism spans exactly the various ways that one can take another's idea, put it into his own document, and not attribute it. Both actions, "put it into his own document" and "not attribute it," are necessary to constitute plagiarism. Clearly, if one considers an idea from another author, but ultimately does not put that idea into his own document, no plagiarism can have occurred. Similarly, if one correctly attributes a borrowed idea that has been placed into one's own document, then there is again no plagiarism.

It is here that the discussion truly begins its foray in the realm of the technical. The two fundamental actions necessary to create plagiarism developed above can be used to project the entirety of written plagiarism (actually the entirety of intellectual borrowing, legitimate or otherwise) into a 2-dimensional vector space. One axis covers the method by which the borrowed idea is put into one's document – call this the *insertion* axis. The other axis covers the degree to which the borrowed idea is attributed to the original author – call this the *attribution* axis. Any particular instance of plagiarism must fall at some point on the 2-dimensional plane defined by these basis vectors.

At one end of the *attribution* axis is complete and proper citation of every borrowed assertion. At the other is no citation or attribution whatsoever. Varying continuously along this axis are citations and attributions with a varying degree of propriety or rigor. At points near the proper end, one finds citations that may be slightly vague, and borrowings attributed with less than perfect rigor. The borrowed idea is still by-and-large attributed to its original source, and critics may rest easy that proper academic integrity has been maintained. Near the other end of this axis, one finds incorrect or sporadic citation, and attribution of only a subset of the borrowed idea, possibly as an attempt to conceal the extent of the borrowing. Assuming that some non-negligible amount of idea (however that may be quantified) has indeed been borrowed from another author, points near the proper end of the axis constitute legitimate borrowing, and points near the other end of the axis constitute plagiarism, pure and simple.

Clearly, there must be some point along this axis where borderline legitimate borrowing becomes borderline plagiarism. This is an unbelievably tricky area. However, for purposes of this discussion, a simple answer is evident: Deciding whether or not a borrowed idea has been properly attributed and/or cited requires a semantic understand of the surrounding text. This is a task of trivial simplicity for any literate human, but incredible difficulty for even the most advanced computer algorithms. For this reason, this discussion will not concern itself with assessing proper attribution of borrowed ideas, but will instead focus entirely on identifying instances of academic borrowing. When presented with a list (or, better yet, a visual display) of the borrowed ideas that have been so identified, a human operator can then easily decide which have been properly attributed and which have not [4]. It should be reasonably apparent that regardless of how another author's idea is inserted into one's document, the degree of proper attribution alone governs whether or not the action constitutes plagiarism. For purposes of clarity and simplicity, this discussion will hereafter assume that all instances of academic borrowing lack proper attribution, and thus constitute plagiarism.

Since the *attribution* axis has been thus summarily eliminated from consideration herein, the discussion can now return to the *insertion* axis. At one end of the *insertion* axis, one directly inserts the exact written words of another author, unaltered, into one's own work. (Although this action can certainly be legitimate, as in the case of a direct quotation, this discussion will assume that such is not the case.) At the other end of the *insertion* axis, one inserts the ideas of another author, written entirely in one's own words, into one's own work. Varying continuously along this axis is the degree to which the original author's original words are preserved. At points near the exact-copy end, most of the original author's words are preserved, with only minor changes or alterations – possibly with the motive of obfuscating the plagiarism to make it less readily detectable. At points near the other end, a few short phrases or terms of art from the original author might remain, or the argument might follow the general structure of the original author's argument, with little, if any, textual similarity.

For points near the exact-copy end of the spectrum, a simple string comparison – searching for exact matches will produce excellent results. For points near the written-in-one's-own-words end of the spectrum, however, it is clearly necessary to understand the meaning of each argument in order to assess their similarity. Unless computers can one day understand meaning in human language, automatically detecting this form of plagiarism will forever remain an unattainable goal. Intermediate points on the *insertion* axis, however, contain varying amounts of textual similarity - similarity that can be detected without regard for meaning. A reasonable goal, then, in seeking to improve automated plagiarism detection techniques, would be to extend their effective domain further down the *insertion* axis from the trivial case of exact-copy plagiarism. That is exactly what this discussion hopes to achieve.

Ultimately, plagiarism as the stealing of another's ideas is beyond the capability of today's computers to detect. To do so in general would require a semantic understanding of human language. Certain forms of plagiarism, however, are susceptible to computer detection. To plagiarize, in a written document, an idea expressed in another's written document, one must necessarily use some percentage (between 0 and 100%) of the original author's original words in one's own writing. If one uses a non-negligible percentage of those words, one's plagiarism may, indeed, be detectable by a computer process using some manner of string comparison algorithm. This is the specific form of plagiarism that will be examined in this discussion, which will seek, ultimately, to improve the effectiveness of the string comparison algorithms used for this purpose.

1.2 Past Efforts in Plagiarism Detection

There are a large number of plagiarism detection techniques discussed in the technical literature [4], and a similarly large number of commercial plagiarism detection products [5]. Many of these commercial products do not release the technical details of their implementations, making analysis difficult. However, in most cases, the general paradigm of the commercial algorithm in question is known to a reasonable degree [5]. In a nutshell, just about every plagiarism detection algorithm currently in use or under research is, at its heart, a glorified string comparator. These algorithms may differ wildly, however, in how they implement their comparison operations and in how they score their results [4]. Some are suitable only for detecting unadulterated exact-copy plagiarism, while others are capable of identifying exact-copy plagiarism that has been slightly obfuscated [5].

For the most part, pairwise document comparison is the most widespread and effective technique. However, some commercial detection products are designed to compare a query document against a large corpus of potential source documents, or against the internet itself. In these situations, some form of filtering is typically used to narrow down the corpus to a small number of documents with a high likelihood of containing similarity. Ultimately, however, document comparison and similarity scoring is performed in a pairwise fashion [6]. Thus, this discussion will limit itself solely to the consideration of pairwise document comparison techniques.

A most naive plagiarism detection algorithm might divide up one document into a set of many small, fixed-length substrings. This set of substrings is then compared against another document. When a match is found, it is recorded. Matches that are adjacent in both documents can be combined into a longer match. The documents are then given a similarity score based on the number and length of matches.

Clearly, this naive approach can only detect exact-copy plagiarism (or, depending on the size of the substrings, very sparse obfuscation). A similar approach would be to identify the locations of words shared between documents, and construct longer matches as sequences of shared words [7]. Some variation on this word pairs algorithm is likely at the heart of commercial plagiarism detection products like CopyCatch [5]. Additionally, CopyCatch also appears to have the ability to improve this similarity measure by identifying different parts of speech of the same word [8]. As its internal algorithms are not public knowledge, however, the exact method by which this is accomplished is impossible to determine [5].

Another technique employed by some plagiarism detection products is to compare two documents based on certain characteristic vectors. A characteristic vector might contain frequencies of certain words and word pairs, in addition to other statistical information about the document. Two documents can then be compared for similarity simply by taking the dot product of their characteristic vectors [4]. It appears, however, that no commercial algorithm that uses this type of technique specifies any additional details on its implementation. This approach is likely reasonably robust in the face of obfuscation. However it lacks a simple way to identify the location of the alleged plagiarism within each document, making it difficult for a human operator to confirm or deny the machine's initial assessment.

A somewhat more advanced technique often used for document comparison is *fingerprinting*, on which much publicly-available research has been done. In a nutshell, *fingerprinting* is an optimization technique for rapid pairwise document comparison. To start, each document is preprocessed to remove punctuation and whitespace, and to covert all alphabetic characters to lowercase. Then, each document is divided up into k-grams, short strings of k characters. k-grams are selected to overlap, such that there is one k-gram beginning on every character in the document, up until the last $k - 1$ characters. (To illustrate this point, consider the string "helloworld". Assuming $k = 3$, this string would be divided into the 3-grams: "hel", "ell", "llo", "low", "owo", "wor", "orl", and "rld".) Each k-gram is then hashed, and a subset of the hashes are selected to serve as the document's fingerprints [9].

There are a number of different techniques for selecting which hashes to use as fingerprints. Some select hashes that are 0 *mod p* for some number p. Assuming the hash function is strong, this will produce a good random distribution of k-grams, with the happy side effect that potentially matching fingerprints are selected for each document (clearly, two hashes cannot be the same if they are different *mod p*). It is conceivably possible, however, that a long sequence of k-grams may not contain any that hash to 0 *mod p*. In fact, the size of the gap between successive fingerprints is unbounded with this approach. Thus, another technique is to divide the document into windows of a fixed size, and select the lowest hash value within each window to be a fingerprint. This guarantees that the maximum gap between fingerprints is at most twice the length of the window, preventing long instances of plagiarism from randomly going undetected [9].

Regardless of how the fingerprints are chosen, however, once the fingerprint hashes have been selected and calculated, documents can be compared quite quickly. For a pairwise inquiry, one need simply check to see if the two documents have any fingerprints in common. Common fingerprints indicate a likely textual match. As in the naive case above, mutually consecutive matches can be combined to obtain longer matches. Document pairs with little similarity can be immediately discarded. For document pairs with a substantial number of fingerprint matches, a more thorough analysis of those regions with identified similarity can then be conducted, before a similarity score is ultimately assigned [9].

The final method to be discussed herein is a sentence-level comparison technique. In this approach, one first divides each document into sentences. Each sentence is then represented as a set of words. Duplicates are removed, and ordering does not matter (punctuation and capitalization and such are also removed). Very common words that do not contain substantive information (i.e. the articles and conjunctions) are also removed from these sets. Each sentence in one document is then pairwise compared with each sentence in the other document. Each sentence pair is assigned a similarity score based on the number of words in common (i.e. the cardinality of the intersection of the two sets) and the lengths of the original sentences. Sentence pairs with low similarity (a score below some arbitrary threshold) are ignored. Overall document similarity is then calculated as an aggregate of the similarity scores of the remaining (high-scoring) sentence pairs [6].

While it may be slightly tangential to this discussion, the authors of this sentence-level comparison technique also make some interesting comments on detecting similarity among a large corpus of many documents (for example, all papers submitted in a class for a particular assignment) that are worthy of a mention herein. To begin, as one would expect, each document in the corpus is compared pairwise with each other document. This generates a large number of links (with attached scores) between sentences in documents in the corpus. In order to eliminate false positive matches arising when many students all cite the same source material, sentences with widespread similarity to many other documents in the corpus are removed. The authors suggest that occurrences of willful plagiarism are likely to occur within only a small number of documents, and that such broad similarity is likely due to a quotation from a communal source. Additionally, when assigning each document an overall similarity score, a weighting factor is applied that accounts for the percentage of matches that occur between the document in question and its most similar counterpart. This, ostensibly, favors documents with a large amount of pairwise similarity to one document relative to communal similarity to a large number of other documents [6].

There are large number of plagiarism detection techniques in use and in development today. At their cores, they all boil down to little more than string comparison techniques. However, not all plagiarism detection algorithms are created equal. Some are only capable of detecting reasonably sized blocks of exact-copy plagiarism (plagiarism where one author inserts an exact copy of another author's words into his own paper). Some are capable of identifying plagiarism where the direct copying of another's words is obfuscated to some small degree (possibly by word alteration and/or rearrangement). A further discussion of techniques for obfuscating plagiarism to avoid detection and existing algorithms' robustness against these methods will occur in the following section.

1.3 Effectiveness of Current Methods

Before one can reasonably attempt to improve existing methods, one must first determine the ways in which existing methods require improvement. This is true for all things, but in the particular case of plagiarism detection, the rules of the game are slightly modified. The world of plagiarism detection algorithms is divided into two domains. On the one side, there are algorithms described in the technical literature. These come with a reasonable (although variable) degree of specification, but generally no working reference implementations. On the other side, there are commercial plagiarism detection products. These are clearly working implementations (for the most part) but generally come with virtually no specification of the internal algorithms used. Thus, one must choose between analyzing theoretical algorithms (about which one knows much) in a purely theoretical manner and analyzing practical implementations (about which one knows little) in a purely empirical manner. For this discussion, the choice is made simpler by the fact that commercial implementations of plagiarism detection algorithms are not free, and are in some cases quite expensive. Thus, a purely theoretical analysis of algorithm performance and effectiveness will be undertaken.

This discussion would be fairly boring if it limited itself to exact-copy plagiarism alone. Such plagiarism is trivially easy to detect, even with naive techniques. A more useful and interesting inquiry might concern itself with algorithms' effectiveness at detecting plagiarism that has been intentionally obfuscated to avoid detection. Before one can discuss how a particular algorithm might perform against an instance of obfuscated plagiarism, however, one must examine the ways in which plagiarism might be obfuscated. This list is not intended to be exhaustive, merely to span a reasonable gamut of possible techniques.

The most basic techniques for plagiarism obfuscation involve first making an exact copy and then altering parts of it, permuting parts of it, or otherwise changing it in minor ways that do not involve a creative synthesis of new ideas but merely a working knowledge of grammar and usage. For example, one might simply flip the order of certain words in a clause, or rearrange the order of clauses in a sentence (I.e. change "Jack's car" to "the car of Jack" or change "if A then B" to "B, if A".) Getting a little more involved, one might change the tense or number in a sentence. (I.e. change "he runs" to "he ran" or change "one does..." to "people do...".) With further effort still, one could replace certain words with close synonyms. (I.e. change "he ran quickly" to "he ran swiftly".) A reasonably industrious combination of these techniques should produce a modified sentence with little textual similarity to the original, with little effect on meaning.

Finally, one more (somewhat more creative) technique to obfuscate plagiarism deserves a mention here: paraphrasing. This is not really a method of obfuscating exact-copy plagiarism as it is a separate method of plagiarism itself. Indeed, if one were to rephrase a plagiarized idea completely in one's own words, it should (one would think) be completely impervious to detection by any manner of textual analysis technique. However, as any individual's attempt at paraphrasing is unlikely to be perfect, there is a reasonable chance that certain key phrases from the original text will survive intact. Obscure diction or particularly clever phraseology seem prime candidates for such an occurrence. Thus, in the end, when considering imperfect paraphrasing, one is considering a situation similar to those described above, except with a much smaller percentage of the plagiarized passage maintaining textual similarity to the original.

To begin with, what effect will simply altering word order, without changing any individual words, have on the effectiveness of the various algorithms described above? This obfuscation will likely stymie the naive and fingerprinting approaches to some degree. Since these approaches search for long sequences of textual identity, flipping two words within a longer phrase will effectively split that one long match into two smaller matches. Depending on the size of the k-grams used in the comparison, a sufficient number of word-order changes might even prevent a match from being found. It is most likely, however, that the number of frequency of word-order changes that are feasible to make without destroying readability will only reduce the overall length of the matches identified, not eliminate them entirely. This will probably reduce the similarity score assigned to document pair somewhat, but not completely conceal the plagiarism.

This obfuscation technique will have virtually no effect on the sentence-based word-set method. Since this approach ignores the order of words in a sentence from the beginning, simply altering the order of words within a sentence will conclusively have no effect. Structural changes concerning multiple sentences, however, have the potential to stymie this algorithm to a small degree. For example, splitting one compound sentence into two simple sentences will change the percentage of the original sentence matched by each of the plagiarized sentences. This will slightly reduce the score of each match. Combining two simple sentences into one compound sentence will have a similar effect. An algorithm based on word frequency analysis will similarly be unaffected by this manner of obfuscation.

Obfuscating an instance of plagiarism by changing tense or number will introduce several small changes to the words in each sentence. In English, at least, verb tenses differ primarily by a few characters or possibly the presence of an auxiliary verb. Changing the number of the subject (from singular to plural or vice versa) will also affect the conjugated verbs, again generally by only a few characters. These small character changes throughout the sentence will break up long matches detected by naive techniques, and possibly even by fingerprinting. Indeed, if a single character in a k-gram is changed, that k-gram's hash will become radically different. The sentence-based and word-frequency approaches will similarly be stymied (slightly) by the alterations of certain words. Since only a small percentage of the words in a sentence are likely to change for a tense or number shift, these techniques will likely still detect the plagiarism with reasonably efficacy. It would seem possible to use a dictionary of some kind to equate different forms of the same word (possibly to convert all words to some base form before processing). Some commercial techniques appear to do this [8], but little detail about their implementation is available.

Replacing words with close synonyms will likely have a similar overall effect as changing tense or number. However, as nearly every word in the language has an appropriate synonym, it is possible to make many more alterations within a sentence by synonym substitution than by altering tense or number. With sufficient substitution, one could likely produce a sentence that (although possibly grammatically awkward) bears little textual similarity to the original, while preserving gross meaning. Such obfuscation would stymie pretty much every algorithm discussed, assuming the substitution were thorough enough. Less thorough attempts are likely to remain vulnerable to the sentence-based word-set method, as a reasonable percentage of words will still match. One might be able to use a thesaurus to detect possible synonym substitution, but it does not appear that such a technique has ever been implemented. It is likely computationally intractable.

Finally, poor paraphrasing will likely withstand attempts at detection by most techniques. Despite the fact that there may be short regions of high similarity, naive methods and fingerprinting will almost certainly miss these short matches. If overall sentence structure is dissimilar to the original, sentence-based methods will detect little similarity, although there is a chance that they will find a conserved phrase, especially if it is a large percentage of a particular sentence. Word frequency techniques will be unable to detect similarity.

Overall, it would seem that naive string matching, as well as the more advanced fingerprinting technique, fall victim to roughly the same kinds of obfuscation. Any modification (no matter how insignificant) that can break up a potential long, exact match into several smaller ones will impair the detection ability of these algorithms. If such modifications occur with sufficient frequency, no matches of substantial length will be found, and the plagiarism might very well go undetected by these algorithms. Word-frequency-based techniques remain resilient in the face of structural changes, but will fail in the face of large numbers of word modifications (i.e. synonym substitution or paraphrasing). Sentence-based methods will similarly withstand structural changes and ultimately succumb to extensive synonym replacement. However, they have the potential to detect some amount of poor paraphrasing in certain cases.

Plagiarism is a difficult and elusive beast. Many algorithms have been developed to detect plagiarism in written documents. As has been shown, however, only certain types of plagiarism are detectable at all with these methods, and basically every method ultimately requires a human operator to make a final determination. While certain methods are more effective than others at detecting plagiarism that has been obfuscated in different ways, in the end, all techniques succumb to the same basic problems. Plagiarism obfuscation is the practice of making an exact copy of another's words, and then altering them in ways that do not require additional intellectual thought so as to make the copy appear unrelated to the original. Plagiarism detection is the practice of finding the relationship between the copy and the original (ideally only when such a relationship actually exists) despite attempts at obfuscation. At its heart, plagiarism obfuscation is the practice of altering a character string to make it appear unrelated to the original, without actually destroying the relationship. Plagiarism detection, therefore, is fundamentally the practice of finding hidden relationships between apparently dissimilar character strings. In any attempt to design a more effective algorithm for the purpose of plagiarism detection, it is important to keep these fundamentals in mind.

Chapter 2

Detection

2.1 Borrowing from Bioinformatics

Fundamentally, automated plagiarism detection is an attempt to detect similarities between two strings of characters. More specifically, it is an attempt to detect hidden similarities between two strings of characters that may appear very dissimilar at first glance. Over the last decade, a number of techniques have been developed to tackle this specific problem: string comparison for plagiarism detection. However, the fundamental problem of string comparison is not peculiar to plagiarism detection, but rather underlies higher level problems in many other branches of computer science. It seems reasonable, therefore, to think that an algorithm developed to tackle one of these problems (a problem with string comparison at its heart) might simply and effectively be adapted to perform plagiarism detection as well.

One field from which one might draw an effective string comparison technique, ultimately to be applied to plagiarism detection, is the budding field of bioinformatics. Bioinformatics has virtually exploded over the past decade, with huge advances being made in almost every area. Indeed, it is one of the most active areas of research within computer science, as well as within biology [10]. With such incredible progress being made, it is quite possible that an algorithm has been developed with excellent prospects for use in plagiarism detection that has yet to be applied to that comparatively-more-neglected problem. Indeed, there are several major components of bioinformatics that require some manner of sequence comparison, particularly those related to DNA and protein sequence analysis.

There exists a straightforward analogy between the detection of evolutionary changes in a genome and the detection of plagiarism in a document. Specifically, a particular gene will gradually accumulate small mutations as it evolves over time. These mutations most commonly consist of changes to a single element of the sequence, or insertions and deletions of small subsequences [11]. In many cases, especially where the gene in question performs a vital task, natural selection prevents the gene's function from straying too far afield, even though its sequence may be drifting slightly. Analogously, when one attempts to obfuscate an instance of exact-copy plagiarism, one is attempting to perturb the sequence of characters while preserving the semantic meaning of the passage as much as possible. Therefore, identifying genes of similar function by analyzing their DNA sequences (or, similarly, identifying proteins of similar function by analyzing their sequences) is much the same task as identifying passages of likely plagiarism by analyzing their character sequences. It seems reasonable that one could adapt an algorithm developed for this problem in bioinformatics to perform plagiarism detection.

The *de facto* standard algorithm for aligning and comparing DNA and protein (amino acid) sequences is called BLAST [12]. BLAST stands for Basic Local Alignment Search Tool. It is typically used for finding subsequences with high similarity among two (or more) longer sequences of DNA or amino acids. In bioinformatics, finding an "alignment" of two sequences means finding a way to line them up next to each other (usually with some offset at the beginning of one of them, and possibly with some number of gaps inserted into each) that maximizes the similarity of adjacent elements in the sequences. A "global alignment" seeks to maximize this similarity along the entire length of the two sequences, while a "local alignment" (the kind sought by BLAST) seeks only to maximize this similarity over a small region (accepting a small number of gaps and mismatches to make the alignment as long as possible). Since there can be many different regions of local similarity, a local alignment will typical produce many possible alignments, each with an associated alignment score.

BLAST will not return an overall measure of similarity between two entire sequences; it will simply return a large number of potential local matches, each with its own similarity measures.

The classic approach to performing local sequence alignment is the Smith-Waterman algorithm [13]. Smith-Waterman uses dynamic programming to find the optimal local alignment relative to a particular scoring metric (basically a specification of penalties for gaps and mismatches). However, although Smith-Waterman is provably optimal, it requires $O(m \times n)$ time to align two strings of lengths m and n respectively [13]. This can be quite slow, especially when applied to genomic data with many millions of elements in each sequence. To speed this process up, BLAST uses a heuristic approximation to the Smith-Waterman algorithm that trades off a small amount of accuracy and specificity for an increase in speed.

To begin with, BLAST searches for small, fixed-length, exact matches between the two strings. These initial matches are used as *seeds* upon which to build full local alignments. BLAST then attempts to improve the alignment score by extending the matches in both directions. Gaps (insertions and deletions) are still not considered at this stage. Additional matches will improve the score of the alignment, while mismatches will reduce it. BLAST will continue to expand the alignment outward past a small number of mismatches, as long as sufficient additional matches are found to keep the overall score from falling. Then, using each of these candidate alignments as a starting point, BLAST uses a modification of the Smith-Waterman algorithm to introduce the possibility of gaps, and continue to improve the alignment. This process results in a small number of high scoring local alignments at various points on the query string.

BLAST is intended to find similarity between small sections of long sequences, where the (presumed) original sequence identity has been obscured by small changes, small insertions, and small deletions. Plagiarism obfuscation is, in a nutshell, exactly that. The techniques for plagiarism obfuscation discussed in the previous chapter effectively create small changes, small insertions, and small deletions (although the definition of “small” may vary) in the original exactly-copied passage. Thus, plagiarism obfuscated in these manners should be readily detectable by a BLAST-based approach.

The most basic obfuscation technique discussed above – simply changing the word order or sentence structure – could easily have no effect on BLAST's ability to identify the plagiarism. If two clauses of a sentence are reversed, each clause will still register as a substantial local alignment match. A human operator could then easily see that the two smaller alignments together constitute a match of the entire sentence. Smaller structural changes (i.e. flipping the order of a few words) will also likely break an alignment into two smaller ones, but might also (depending on overall length and mismatch scoring penalties) still show up in one single alignment as a section of dissimilarity. Similarly, the obfuscation technique of changing tense or number in a sentence will likely not stymie BLAST either. This technique, as discussed in the previous chapter, will primarily have the effect of changing the inflections of a few nouns and/or verbs – effectively altering, adding, or removing a few characters from the ends of a few words. BLAST should easily recognize this as the simple mutation that it fundamentally is, and the overall length of the alignment will likely not be decreased at all.

However, replacing words with synonyms is a more insidious technique for plagiarism obfuscation. There is no guarantee that two synonyms will have any letters in common, or would even be close to the same length. Thus replacing a single word with a synonym will quite easily add a large number of mismatches as well as several gaps. This might be substantial enough to split an alignment into two. However, depending on the particular circumstances, an individual synonym substitution may not split an alignment, just decrease its score a bit. As long as the number of synonym substitutions is not too large compared to the overall length of the passage, BLAST still has a decent chance of detecting this manner of obfuscation.

Paraphrasing obviously has the potential to completely disguise the instance of plagiarism, Poor plagiarism, as described above, is likely to leave a certain number of phrases and obscure terms from the original source in the obfuscated passage. BLAST will almost certainly find these conserved bits, even if they are fairly short. It is then basically up to the human operator to be able to infer plagiarism from those isolated spots of similarity.

Ultimately, BLAST shows great potential to be used in a plagiarism detection solution. The field of bioinformatics in general has produced a considerable amount of progress in computer algorithms in the last decade. In that time, BLAST has risen to become the standard for sequence alignment.

There is an apt analogy between obfuscated plagiarism and the evolution of genes. In addition, alignment of DNA and protein sequences requires a similar degree of sequence comparison to the problem of plagiarism detection. Thus, it seems reasonable to expect that an effective plagiarism detection algorithm can be built with BLAST as a base. Implementing this algorithm and assessing its effectiveness will be the goal of the remainder of this discussion.

2.2 Implementation

BLAST is designed to analyze sequences of DNA or amino acids. In order to use BLAST for plagiarism detection, it is necessary to adapt the BLAST engine for use on English text. There are fundamentally two ways to go about this task. The first of these is to modify the BLAST internals to operate on a different type of sequences. The second approach is to create a mapping that converts English text into a sequence format that BLAST already understands and can compare. As the latter option is much simpler, and (as shall be discussed below) ultimately not likely to be substantially less effective, it has been chosen for this investigation.

BLAST can operate on two types of sequences: DNA and amino acids. DNA sequences consist of 4 characters, representing the 4 nucleotide bases. Amino acid sequences consist of 24 characters: 20 characters represent the 20 different amino acids that occur in proteins[14]; the remaining 4 characters represent certain combinations of chemical similar amino acids as well as a wildcard amino acid [15]. As there are 26 letters in the English alphabet, the alphabet size of an amino acid sequence is almost sufficient by itself. Thus, it seems not particularly daunting to convert sequences of English text into sequences of amino acids. This is the approach that will be used in this investigation to allow English text to be processed by BLAST.

It is important to note explicitly at this point an aspect of BLAST that should by now be self-evident. BLAST operates on sequences of characters at the character level. While one might imagine that a plagiarism detection algorithm would generally include some concept of *words* and *sentences*, BLAST sees none of these distinctions. Thus, a plagiarism detection solution based on BLAST will, by nature, view the English text on which it operates as nothing more than a sequence of alphabetic characters. This plagiarism detection approach will, at its most basic level, look purely for similarity in the alphabetic character sequences. Clearly, in an instance of exact-copy plagiarism, the two character strings in question will be completely identical (and this identity will be trivially detectable by BLAST). One can readily imagine that, in an instance of plagiarism that has been somewhat obfuscated, the two strings will still share a large degree of similarity at the character level. A fundamental question is the degree to which BLAST will still detect the similarity in such partially obfuscated sequences.

In order to convert free-form English text into an amino acid sequence, one must do more than simply map a 26 character alphabet into a 24 character one. For one, natural English text contains many more than 26 characters. Capitalization alone will double that number. Numerals, punctuation, and whitespace will increase it even further. None of these characters convey much semantic information, however. Thus, their removal should not substantially diminish the overall information content of the prose, and thus doing so should not adversely effect the ability of a plagiarism detection algorithm to detect plagiarism. The first step in processing, therefore, will be to remove all non-alphabetic characters from the text, and to convert all alphabetic characters to lower case.

This first step will yield a sequence with a 26 character alphabet representing the original free-form English text. This is closer to the goal of a 24 character alphabet, but not quite sufficient. To eliminate an additional character, one could simply replace all instances of one character with another already in the alphabet, effectively merging the two characters. Doing this twice will yield a 24 character alphabet. Since the amino-acid alphabet is a strict subset of the English alphabet, a simple solution presents itself: replace the two characters that occur in the English alphabet but not in the amino-acid alphabet with a character that occurs in both alphabets. If the chosen replacement character occurs infrequently in English text, most of the semantic information contained in the English character sequence should be preserved into the amino-acid sequence. The characters “j” and “o” do not occur in the amino-acid alphabet for the sequence format used in this discussion[15]. In this investigation, these characters are replaced by “q” in the second step of processing. These two steps combined form the preprocessing phase of the BLAST-based plagiarism detection algorithm.

This manner of conversion from free-form English text to an amino-acid sequence is clearly not a reversible mapping. A huge amount information is lost in the process. As established above, this information loss is not particularly detrimental to the efficacy of the plagiarism detection algorithm, since most semantic information is preserved;[1] the information loss is generally localized to formatting and stylistic information (i.e. punctuation and capitalization). Regardless, this information loss makes it infeasible to convert back to the original text from the amino-acid sequence version. One would ideally like to view the results of the BLAST query as they occur in the original, formatted version of the documents in question. However, BLAST will naturally output its results in the context of the sequences input to it. In order to display these results in a useful manner, then, one must somehow contrive to convert coordinates in the amino-acid sequence representation back to coordinates in the original representation.

To allow for this coordinate reconstruction, it is necessary to save some additional information during the first two processing steps. In the approach used herein, an integer array is constructed during the preprocessing phase that maps coordinates in the amino-acid version of the sequence back to coordinates in the original version. The array is the same length as the amino-acid sequence, and the value of the ith element corresponds to the coordinate in the original document of the ith character in the amino-acid sequence. This allows for a quick, easy lookup of coordinates produced by BLAST into coordinates germane to the display of the results.

With this mapping established, one can begin to use BLAST as a plagiarism detection engine. To compare two documents for plagiarism, one need simply convert the two documents into their respective amino-acid sequences, and run BLAST on these sequences. BLAST will produce a

[1]There is a fine distinction between semantic and syntactic information, here. Ideally, one would like the computer to be able to perform a purely semantic analysis – to compare the actually linguistic meaning contained in the two documents in a search for similarity. This is, unfortunately, not possible for today's computers. The goal, then, is to approximate a full semantic analysis as closely as possible, using comparisons that a computer is physically capable of performing. The BLAST-based algorithm implemented in this discussion seeks to find instances of semantic identity by searching for instances of character-sequence identity (a purely syntactic undertaking). It is a reasonable assumption that if two sentences contain exactly the same letters in exactly the same order then they most likely contain the same semantic information (and almost certainly have originated from a common source). As the syntactic identity between two phrases decreases, so does the likelihood that the semantic information they each represent is the same. Throughout this discussion, reference is made to the semantic information contained in a document's text. In each of these cases, the information referred to is more specifically the semantic information suggested by the syntactic information actually being analyzed.

large number of candidate local alignments, likely ranging in size from a dozen or so characters to (potentially) hundreds of characters. Each of these alignments represents a particular instance of similarity between the two sequences, potentially corresponding to a particular instance of plagiarism between the two documents. Along with each alignment, BLAST returns an array of statistical information concerning that alignment, which will become useful later in this discussion.

In order to determine if the document pair contains an actual instance of plagiarism, it is necessary to some degree to have a human operator examine the BLAST results. For one, BLAST will produce a large number of inconsequential matches, which must be discounted. While a 19-character-long matches may be substantial when dealing with proteins, such a match is likely not useful in the context of plagiarism detection. (For example, a 19-character-long match could easily correspond to a common 2-word phrase, such as "interestingly enough," that occurs independently in both documents.) At the same time, such an alignment could be part of a larger instance of plagiarism that has been obfuscated to a sufficient degree as to break up long matches. Thus, in the end, to make a conclusive judgement for or against the existence of plagiarism requires a human operator to assess the results.

As BLAST is quite likely to return a large number of candidate alignments, most of which are uninteresting, it would behoove a human operator to be able to filter through them effectively, and, to some extent, separate the wheat from the chaff. For this purpose, a simple visual interface was conceived, which displays both documents side-by-side, with matches highlighted in a varying color. (The color varies continuously across the spectrum with the coordinate of the match in the first document, and corresponding phrases in each document are highlighted in the same color. This allows the user to readily see where a particular match occurs in each document, and analyze the surrounding text for insight.) In addition to the color coding scheme, a simple filtering interface allows the operator to restrict displayed matches according to the matches' statistical information returned by the BLAST query.

The methods described in this section articulate a manner in which BLAST can be adapted to perform plagiarism detection. The approach described will produce a large number of candidate matches within a document pair and display them in a manner that allows a human operator to determine if a candidate match corresponds to an actual incidence of plagiarism. The hope is that BLAST will be effective at finding matches even between plagiarized passages that have been intentionally obfuscated (and that the human operator will be effective at paring down the false-positive matches, but that's another story). At this point in the discussion, the BLAST approach cannot quantify the likelihood that a particular document pair contains plagiarism, merely indicate places where such plagiarism might occur. The goal of the subsequent discussion, then, will be to develop a method of quantifying the BLAST results in order to compare them across document pairs, and ultimately to assess the effectiveness of this approach to plagiarism detection.

2.3 Results

Before the effectiveness of this approach to plagiarism detection can be analyzed, it is necessary to understand the output of the plagiarism detection engine. This section will explore the intermediary and final results of a BLAST-based analysis of a hypothetical document pair. In so doing, the nature and significance of each phase of the results can be analyzed. This will lay the groundwork for the analysis that will occur in the following chapter. To begin this hypothetical discussion requires a hypothetical document pair for comparison. For this purpose, two pieces of pseudorandomly generated *Lorem Ipsum* text was chosen, each 5 paragraphs in length. Each document was generated by the Lorem Ipsum generator at `www.lipsum.com` [16]. For reference, and to allow future researchers to make more apt comparison to the results described in this section, the full text of each document used herein is included in Appendix A. This section will follow the computation process through each step necessary to make an assessment on the likelihood of plagiarism within this document pair.

The first step in the computation is the conversion of free-form English text into an amino acid sequence capable of being analyzed by BLAST. This operation is performed on each document separately. The techniques used to effect this conversion are described in the previous section. However, to make the action of the conversion process more readily apparent, a sample conversion of the first sentence of the first example document is displayed in Figure 2.1. In the top box is a free-form sentence; in the bottom box is that sentence rendered into an amino-acid sequence. As the reader can clearly see, all capitalization, punctuation and whitespace have been removed in the resultant sequence, and the two non-occurring letters in the amino-acid alphabet have been replaced by "q." As "q" does not occur very commonly in the original text, the instances of substitution are readily apparent, and little semantic information is lost.

Mauris pellentesque, odio sit amet accumsan laoreet, libero libero volutpat nulla, tempus rutrum lectus leo malesuada mi.
`maurispellentesqueqdiqsitametaccumsanla qreetliberqliberqvqlutpatnullatempusrutr umlectusleqmalesuadami`

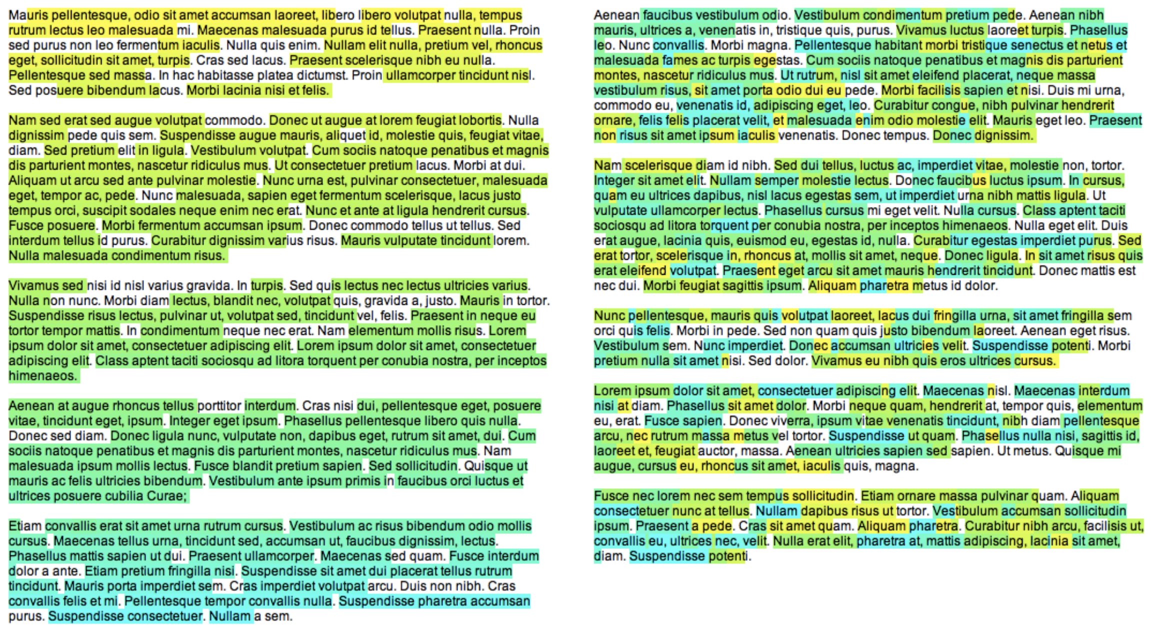
Figure 2.1: Conversion of English Text into Amino-Acid Sequence

Once the free-form text has been converted into amino-acid sequences, BLAST can tackle the brunt of the document comparison operation. As described above, given the two sequences, BLAST will produce a large number of candidate local alignments. Each of these alignments will likely contain some regions of high similarity, some mismatches, and possibly even some gaps. Figure 2.2 presents a small subset of the raw BLAST output representing a single candidate alignment. (This particular alignment is long enough that it spans two lines of output.) While this display contains a large amount of information about this particular match, it does not present that information in such a way as to enable a human operator to readily assess the likelihood of actual plagiarism within this region. Further processing of this raw output is required to create a more useful visualization.

```
          Length = 80

 Score = 33.1 bits (74), Expect = 6e-06
 Identities = 34/93 (36%), Positives = 42/93 (45%), Gaps = 33/93 (35%)

Query: 7  PELLENTESQQEODIOSITAMETACCQMSANLAOREETLIBEROLIBEROVOLQTPATNQ 66
          PELLENTESQQE               ++AN   R+ T          R   +Q  + N+
Sbjct: 1  PELLENTESQQEH------------ABITANTMORBIT----------RISTIQQESENE 38

Query: 67 LLATEMPQSRQTRQMLECTQSLEOMALESQADA 99
             +E  +          TQS E MALESQADA
Sbjct: 39 CTQSETNE----------TQS-ETMALESQADA 60
```

Figure 2.2: Example of Raw BLAST Output

To allow a human operator to more readily judge the similarity of portions of a document pair, a visualization is created where the two documents are displayed side-by-side, and each potential match is highlighted in a varying color. As discussed above, the highlighting color varies smoothly and continuously throughout the first document, and corresponding phrases in the second document are highlighting with the appropriate color. Figure 2.3 should make this visualization scheme clear.



Mauris pellentesque, odio sit amet accumsan laoreet, libero libero volutpat nulla, tempus rutrum lectus leo malesuada mi. Maecenas malesuada purus id tellus. Praesent nulla. Proin sed purus non leo fermentum iaculis. Nulla quis enim. Nullam elit nulla, pretium vel, rhoncus eget, sollicitudin sit amet, turpis. Cras sed lacus. Praesent scelerisque nibh eu nulla. Pellentesque sed massa. In hac habitasse platea dictumst. Proin ullamcorper tincidunt nisl. Sed posuere bibendum lacus. Morbi lacinia nisi et felis.

Nam sed erat sed augue volutpat commodo. Donec ut augue at lorem feugiat lobortis. Nulla dignissim pede quis sem. Suspendisse augue mauris, aliquet id, molestie quis, feugiat vitae, diam. Sed pretium elit in ligula. Vestibulum volutpat. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Ut consectetuer pretium lacus. Morbi at dui. Aliquam ut arcu sed ante pulvinar molestie. Nunc urna est, pulvinar consectetuer, malesuada eget, tempor ac, pede. Nunc malesuada, sapien eget fermentum scelerisque, lacus justo tempus orci, suscipit sodales neque enim nec erat. Nunc et ante at ligula hendrerit cursus. Fusce posuere. Morbi fermentum accumsan ipsum. Donec commodo tellus ut tellus. Sed interdum tellus id purus. Curabitur dignissim varius risus. Mauris vulputate tincidunt lorem. Nulla malesuada condimentum risus.

Vivamus sed nisi id nisl varius gravida. In turpis. Sed quis lectus nec lectus ultricies varius. Nulla non nunc. Morbi diam lectus, blandit nec, volutpat quis, gravida a, justo. Mauris in tortor. Suspendisse risus lectus, pulvinar ut, volutpat sed, tincidunt vel, felis. Praesent in neque eu tortor tempor mattis. In condimentum neque nec erat. Nam elementum mollis risus. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos.

Aenean at augue rhoncus tellus porttitor interdum. Cras nisi dui, pellentesque eget, posuere vitae, tincidunt eget, ipsum. Integer eget ipsum. Phasellus pellentesque libero quis nulla. Donec sed diam. Donec ligula nunc, vulputate non, dapibus eget, rutrum sit amet, dui. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Nam malesuada ipsum mollis lectus. Fusce blandit pretium sapien. Sed sollicitudin. Quisque ut mauris ac felis ultricies bibendum. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae;

Etiam convallis erat sit amet urna rutrum cursus. Vestibulum ac risus bibendum odio mollis cursus. Maecenas tellus urna, tincidunt sed, accumsan ut, faucibus dignissim, lectus. Phasellus mattis sapien ut dui. Praesent ullamcorper. Maecenas sed quam. Fusce interdum dolor a ante. Etiam pretium fringilla nisi. Suspendisse sit amet dui placerat tellus rutrum tincidunt. Mauris porta imperdiet sem. Cras imperdiet volutpat arcu. Duis non nibh. Cras convallis felis et mi. Pellentesque tempor convallis nulla. Suspendisse pharetra accumsan purus. Suspendisse consectetuer. Nullam a sem.

Aenean faucibus vestibulum odio. Vestibulum condimentum pretium pede. Aenean nibh mauris, ultrices a, venenatis in, tristique quis, purus. Vivamus luctus laoreet turpis. Phasellus leo. Nunc convallis. Morbi magna. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Ut rutrum, nisl sit amet eleifend placerat, neque massa vestibulum risus, sit amet porta odio dui eu pede. Morbi facilisis sapien et nisi. Duis mi urna, commodo eu, venenatis id, adipiscing eget, leo. Curabitur congue, nibh pulvinar hendrerit ornare, felis felis placerat velit, et malesuada enim odio molestie elit. Mauris eget leo. Praesent non risus sit amet ipsum iaculis venenatis. Donec tempus. Donec dignissim.

Nam scelerisque diam id nibh. Sed dui tellus, luctus ac, imperdiet vitae, molestie non, tortor. Integer sit amet elit. Nullam semper molestie lectus. Donec faucibus luctus ipsum. In cursus, quam eu ultrices dapibus, nisl lacus egestas sem, ut imperdiet urna nibh mattis ligula. Ut vulputate ullamcorper lectus. Phasellus cursus mi eget velit. Nulla cursus. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Nulla eget elit. Duis erat augue, lacinia quis, euismod eu, egestas id, nulla. Curabitur egestas imperdiet purus. Sed erat tortor, scelerisque in, rhoncus at, mollis sit amet, neque. Donec ligula. In sit amet risus quis erat eleifend volutpat. Praesent eget arcu sit amet mauris hendrerit tincidunt. Donec mattis est nec dui. Morbi feugiat sagittis ipsum. Aliquam pharetra metus id dolor.

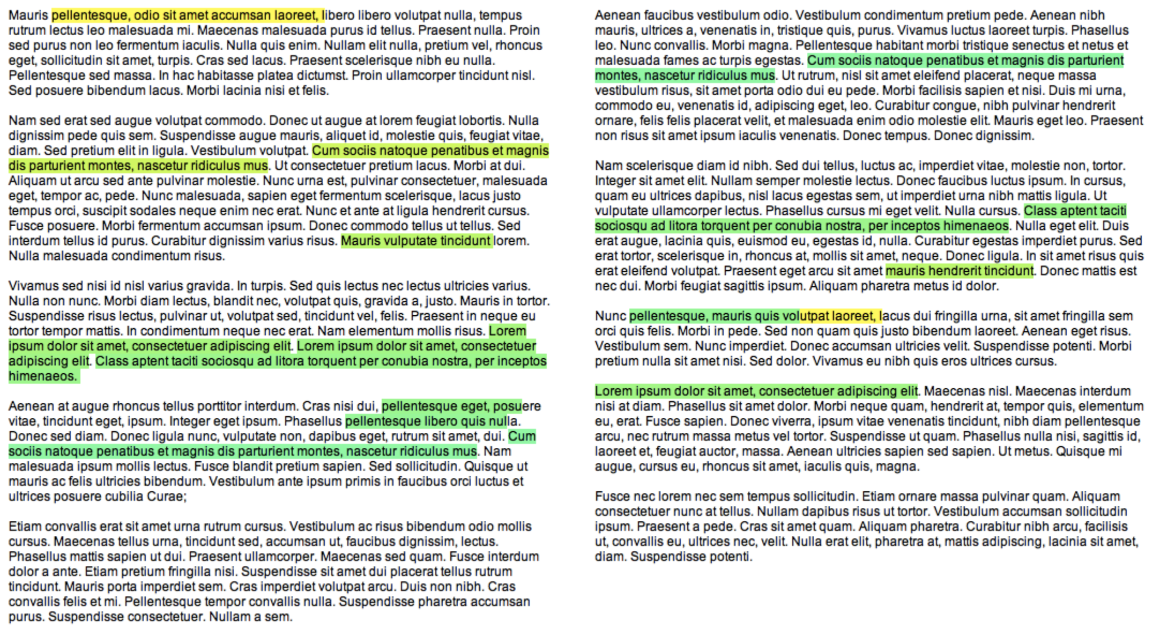
Nunc pellentesque, mauris quis volutpat laoreet, lacus dui fringilla urna, sit amet fringilla sem orci quis felis. Morbi in pede. Sed non quam quis justo bibendum laoreet. Aenean eget risus. Vestibulum sem. Nunc imperdiet. Donec accumsan ultricies velit. Suspendisse potenti. Morbi pretium nulla sit amet nisi. Sed dolor. Vivamus eu nibh quis eros ultrices cursus.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Maecenas nisl. Maecenas interdum nisi at diam. Phasellus sit amet dolor. Morbi neque quam, hendrerit at, tempor quis, elementum eu, erat. Fusce sapien. Donec viverra, ipsum vitae venenatis tincidunt, nibh diam pellentesque arcu, nec rutrum massa metus vel tortor. Suspendisse ut quam. Phasellus nulla nisi, sagittis id, laoreet et, feugiat auctor, massa. Aenean ultricies sapien sed sapien. Ut metus. Quisque mi augue, cursus eu, rhoncus sit amet, iaculis quis, magna.

Fusce nec lorem nec sem tempus sollicitudin. Etiam ornare massa pulvinar quam. Aliquam consectetuer nunc at tellus. Nullam dapibus risus ut tortor. Vestibulum accumsan sollicitudin ipsum. Praesent a pede. Cras sit amet quam. Aliquam pharetra. Curabitur nibh arcu, facilisis ut, convallis eu, ultrices nec, velit. Nulla erat elit, pharetra at, mattis adipiscing, lacinia sit amet, diam. Suspendisse potenti.

Figure 2.3: An Hypothetical Comparison with Complete Markup

It is readily apparent from the visualization in Figure 2.3 that BLAST has identified the vast majority of each document as representing a possible match. It is equally apparent from a cursory reading of both original documents (in Appendix A) that the two documents do not contain textual similarity to the degree suggested by the visualization. Indeed, BLAST seems sensitive enough that it will flag matches as short as 5 characters, as well as matches with as little as 25% character identity. While such extreme sensitivity might conceivably be useful to a dedicated operator searching for an elusive bit of well-obfuscated plagiarism, it generally just results in a huge amount of false-positive matches. For this reason, a rudimentary filtering interface was devised, which allows the operator to restrict the displayed matches to a (hopefully) more useful and informative subset. BLAST provides several statistical values associated with each candidate alignment that can be used for filtering. Most of these are fairly straightforward: *Alignment Length* represents the overall length of the alignment; *Percent Identity* represents the percentage of positions in the alignment occupied by the same character in each sequence; *Mismatches* represents the total number of positions in the alignment occupied by different characters in each sequence; and *Gap Openings* represents the total number of whole gaps (of whatever length) in either sequence in the alignment. Two additional statistical values are provided by BLAST. These are the *Bit Score* and the *Expectation Value* (typically referred to as *E-Value*) of the alignment. The *Bit Score* is a normalized representation of the alignments raw score (calculated based on the number of matches, mismatches, and gaps and the scoring rules being used). The *E-Value* is the number of alignments of the same or better score that one would expect to find in between two random sequences of the same lengths as the documents in question. The *E-Value* provides particularly useful insight into the subjective "quality" of a particular alignment, with lower numbers (ideally much less than 1) representing "better" matches.



Mauris pellentesque, odio sit amet accumsan laoreet, libero libero volutpat nulla, tempus rutrum lectus leo malesuada mi. Maecenas malesuada purus id tellus. Praesent nulla. Proin sed purus non leo fermentum iaculis. Nulla quis enim. Nullam elit nulla, pretium vel, rhoncus eget, sollicitudin sit amet, turpis. Cras sed lacus. Praesent scelerisque nibh eu nulla. Pellentesque sed massa. In hac habitasse platea dictumst. Proin ullamcorper tincidunt nisl. Sed posuere bibendum lacus. Morbi lacinia nisi et felis.

Nam sed erat sed augue volutpat commodo. Donec ut augue at lorem feugiat lobortis. Nulla dignissim pede quis sem. Suspendisse augue mauris, aliquet id, molestie quis, feugiat vitae, diam. Sed pretium elit in ligula. Vestibulum volutpat. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Ut consectetuer pretium lacus. Morbi at dui. Aliquam ut arcu sed ante pulvinar molestie. Nunc urna est, pulvinar consectetuer, malesuada eget, tempor ac, pede. Nunc malesuada, sapien eget fermentum scelerisque, lacus justo tempus orci, suscipit sodales neque enim nec erat. Nunc et ante at ligula hendrerit cursus. Fusce posuere. Morbi fermentum accumsan ipsum. Donec commodo tellus ut tellus. Sed interdum tellus id purus. Curabitur dignissim varius risus. Mauris vulputate tincidunt lorem. Nulla malesuada condimentum risus.

Vivamus sed nisi id nisl varius gravida. In turpis. Sed quis lectus nec lectus ultricies varius. Nulla non nunc. Morbi diam lectus, blandit nec, volutpat quis, gravida a, justo. Mauris in tortor. Suspendisse risus lectus, pulvinar ut, volutpat sed, tincidunt vel, felis. Praesent in neque eu tortor tempor mattis. In condimentum neque nec erat. Nam elementum mollis risus. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos.

Aenean at augue rhoncus tellus porttitor interdum. Cras nisi dui, pellentesque eget, posuere vitae, tincidunt eget, ipsum. Integer eget ipsum. Phasellus pellentesque libero quis nulla. Donec sed diam. Donec ligula nunc, vulputate non, dapibus eget, rutrum sit amet, dui. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Nam malesuada ipsum mollis lectus. Fusce blandit pretium sapien. Sed sollicitudin. Quisque ut mauris ac felis ultricies bibendum. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae;

Etiam convallis erat sit amet urna rutrum cursus. Vestibulum ac risus bibendum odio mollis cursus. Maecenas tellus urna, tincidunt sed, accumsan ut, faucibus dignissim, lectus. Phasellus mattis sapien ut dui. Praesent ullamcorper. Maecenas sed quam. Fusce interdum dolor a ante. Etiam pretium fringilla nisi. Suspendisse sit amet dui placerat tellus rutrum tincidunt. Mauris porta imperdiet sem. Cras imperdiet volutpat arcu. Duis non nibh. Cras convallis felis et mi. Pellentesque tempor convallis nulla. Suspendisse pharetra accumsan purus. Suspendisse consectetuer. Nullam a sem.

Aenean faucibus vestibulum odio. Vestibulum condimentum pretium pede. Aenean nibh mauris, ultrices a, venenatis in, tristique quis, purus. Vivamus luctus laoreet turpis. Phasellus leo. Nunc convallis. Morbi magna. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Ut rutrum, nisl sit amet eleifend placerat, neque massa vestibulum risus, sit amet porta odio dui eu pede. Morbi facilisis sapien et nisi. Duis mi urna, commodo eu, venenatis id, adipiscing eget, leo. Curabitur congue, nibh pulvinar hendrerit ornare, felis felis placerat velit, et malesuada enim odio molestie elit. Mauris eget leo. Praesent non risus sit amet ipsum iaculis venenatis. Donec tempus. Donec dignissim.

Nam scelerisque diam id nibh. Sed dui tellus, luctus ac, imperdiet vitae, molestie non, tortor. Integer sit amet elit. Nullam semper molestie lectus. Donec faucibus luctus ipsum. In cursus, quam eu ultrices dapibus, nisl lacus egestas sem, ut imperdiet urna nibh mattis ligula. Ut vulputate ullamcorper lectus. Phasellus cursus mi eget velit. Nulla cursus. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Nulla eget elit. Duis erat augue, lacinia quis, euismod eu, egestas id, nulla. Curabitur egestas imperdiet purus. Sed erat tortor, scelerisque in, rhoncus at, mollis sit amet, neque. Donec ligula. In sit amet risus quis erat eleifend volutpat. Praesent eget arcu sit amet mauris hendrerit tincidunt. Donec mattis est nec dui. Morbi feugiat sagittis ipsum. Aliquam pharetra metus id dolor.

Nunc pellentesque, mauris quis volutpat laoreet, lacus dui fringilla urna, sit amet fringilla sem orci quis felis. Morbi in pede. Sed non quam quis justo bibendum laoreet. Aenean eget risus. Vestibulum sem. Nunc imperdiet. Donec accumsan ultricies velit. Suspendisse potenti. Morbi pretium nulla sit amet nisi. Sed dolor. Vivamus eu nibh quis eros ultrices cursus.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Maecenas nisl. Maecenas interdum nisi at diam. Phasellus sit amet dolor. Morbi neque quam, hendrerit at, tempor quis, elementum eu, erat. Fusce sapien. Donec viverra, ipsum vitae venenatis tincidunt, nibh diam pellentesque arcu, nec rutrum massa metus vel tortor. Suspendisse ut quam. Phasellus nulla nisi, sagittis id, laoreet et, feugiat auctor, massa. Aenean ultricies sapien sed sapien. Ut metus. Quisque mi augue, cursus eu, rhoncus sit amet, iaculis quis, magna.

Fusce nec lorem nec sem tempus sollicitudin. Etiam ornare massa pulvinar quam. Aliquam consectetuer nunc at tellus. Nullam dapibus risus ut tortor. Vestibulum accumsan sollicitudin ipsum. Praesent a pede. Cras sit amet quam. Aliquam pharetra. Curabitur nibh arcu, facilisis ut, convallis eu, ultrices nec, velit. Nulla erat elit, pharetra at, mattis adipiscing, lacinia sit amet, diam. Suspendisse potenti.

Figure 2.4: An Hypothetical Comparison with Filtered Markup

Figure 2.4 shows a visualization similar to that in Figure 2.3 that has also been filtered according to several of the parameters described above. Specifically, this visualization is only highlighting matches with an *E-Value* $< .03$, with an *Alignment Length* < 20 and with a *Percent Identity* $< 50\%$. With these constraints, virtually all of the highlighted phrases appear to have a high degree of similarity. As one can readily see from Figure 2.4, this is a much more tractable number of matches. Three sentences occur entirely unchanged in both documents: “Cum sociis natoque penatibus et magnis disparturient montes, nascetur ridiculus mus.”, “Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos.”, and the classic “Lorem ipsum dolor sit amet, consectetuer adipiscing elit.” Two shorter phrases occur with some degree of internal changes: “Mauris vulputate tincidunt” becomes “mauris hendrerit tincidunt” and “pellentesque, odio sit amet accumsan laoreet” becomes “pellentesque, mauris quis volutpat laoreet.” Whether or not these instances constitute actual plagiarism or accidental similarity is moot in this case, as the two documents in question were pseudorandomly generated. However, in a normal case of inquiry, it would be up to a human operator to examine each of these small number of similarities in order to make a final determination as to whether or not the documents in question contains actual plagiarism.

The constraints used in the filtering described above do not come out of nowhere. While in practice it is quite straightforward to fiddle around with the parameters until a tractable number of reasonable matches are displayed (as was done in this hypothetical example), the parameters that result from this process have an intrinsic relationship to certain statistical properties of the document pair itself. In theory at least, one could examine these statistical properties and thence determine a reasonable set of filter constraints. Several aspects of the document pair’s statistics are examined in Figures 2.5 through 2.7. These plots present several components of what will henceforth be referred to as the document pair’s *alignment profile*. For a given match parameter, the *alignment profile* is simply the distribution of that parameter’s values plotted in a monotonically increasing manner. The general features of these plots seem typical for a pair of documents with some reasonable degree of similarity, and thus the ensuing discussion should hold true in most cases.

Arguably the most important parameter of a particular alignment is its *E-Value*. The graph in Figure 2.5 displays the distribution of *E-Values* across all of the candidate alignments in this document pair. The graph is plotted on a logarithmic scale for the ordinate because *E-Values* range over many orders of magnitude, and this magnitude is generally all that is important. The graph is plotted on a logarithmic abscissa because only the extreme low end is of interest. Once the *E-Value* approaches 1, the graph flattens out and becomes less useful. In this and in all subsequent of these graphs, the cutoff used in the above filtering is plotted in green. In the case of the *E-Value*, the selected cutoff appears to occur just as the graph is beginning to flatten out. As one can readily see, only a small fraction of the total matches occur in this low end. These are the matches that represent the highest likelihood of potential plagiarism.

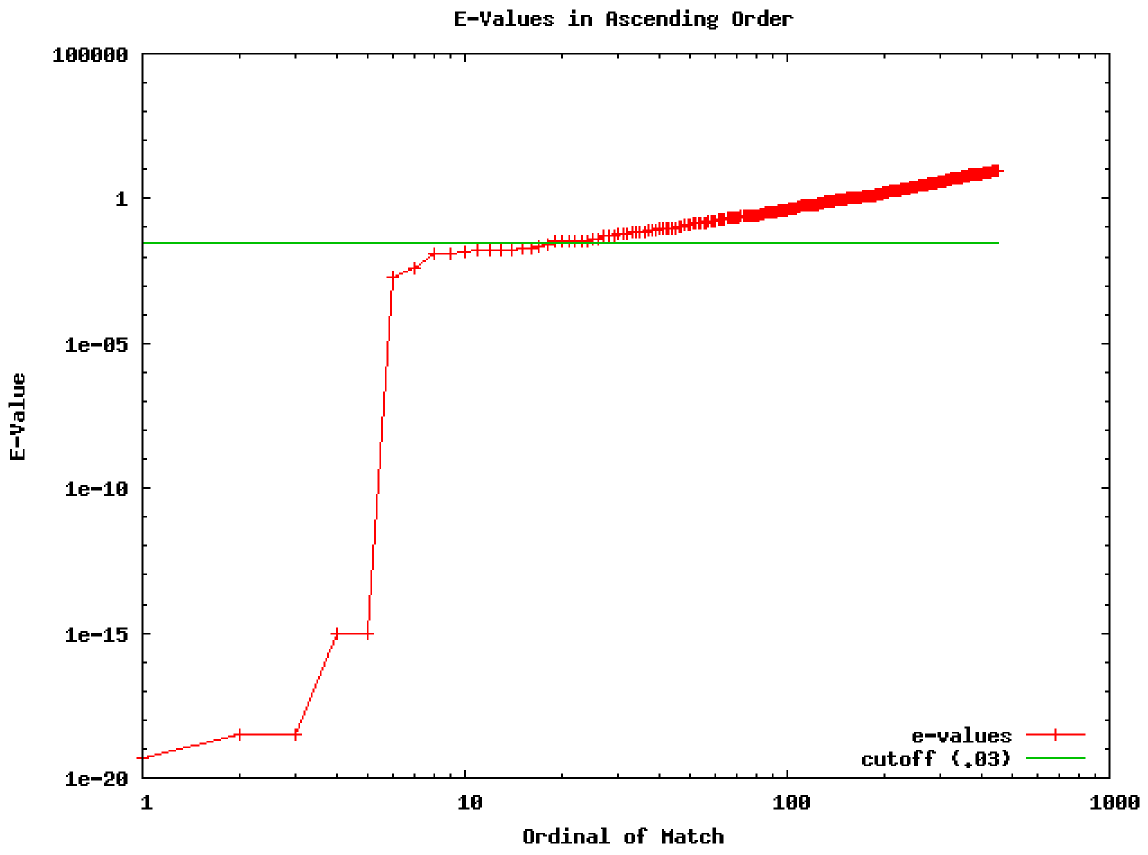


Figure 2.5: E-Values of Matches in Ascending Order

The second parameter that has been used as a filtering constraint is the overall *Alignment Length*. A graph of the distribution of *Alignment Length* is presented in Figure 2.6. As one can see, and as one would expect, the regions of similarity identified in these documents, fall very nicely into regular exponential distribution over the space of *Alignment Length*, with few long matches and many short ones. The chosen cutoff (shown in green) seems to cut right through the “middle” of this exponential curve. The longer matches selected by this operation are less likely to represent words or short phrases that randomly occur in both documents, and more likely to represent substantial phrases with a greater likelihood of containing plagiarism.

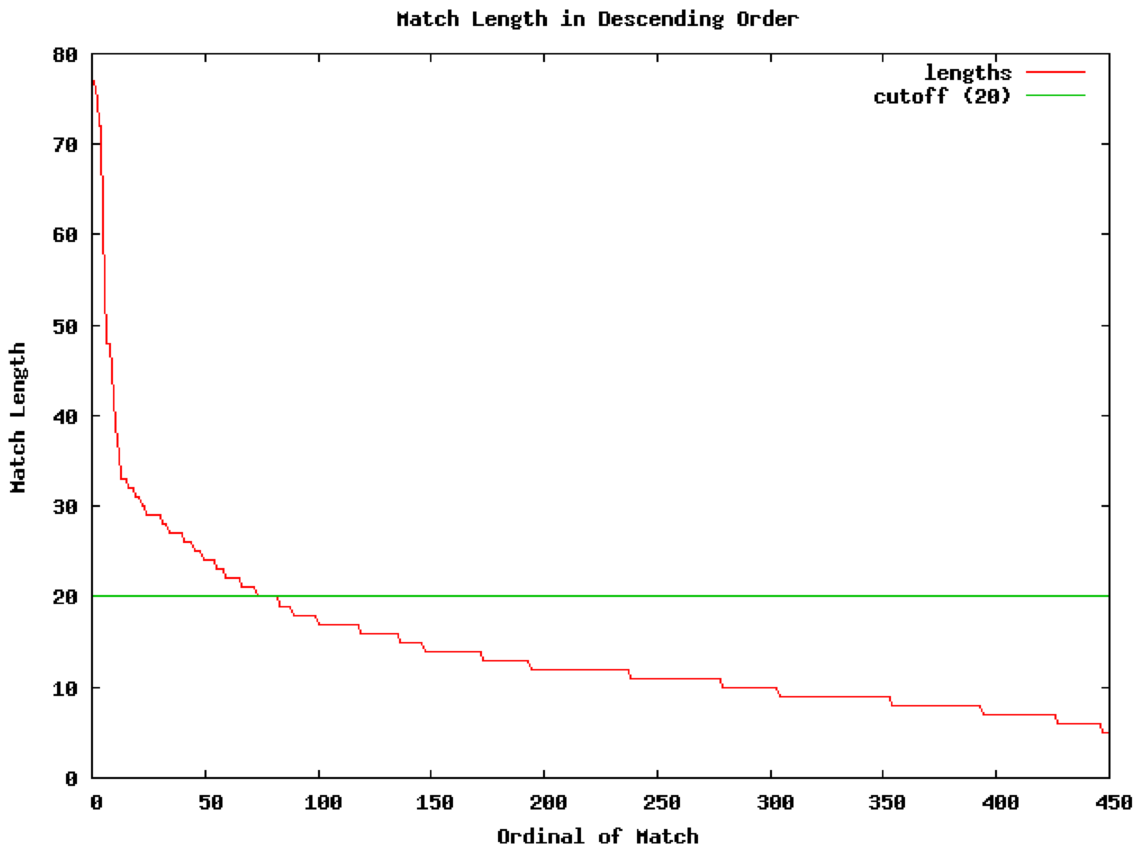


Figure 2.6: Alignment Lengths of Matches in Ascending Order

The final parameter used in the filtering above is the *Percent Identity* of the match. The graph of this parameter’s distribution, provided in Figure 2.7, is much less informative than the first two. Indeed, in this hypothetical discussion, the constraint on this parameter was only placed into the filter to remove the few matches that made it through the other constraints but had little enough textual similarity as to be unlikely to represent actual plagiarism. As there are no particularly remarkable features of this parameter’s distribution that would suggest a logical cutoff, it is likely that the *Percent Identity* parameter will never be used for much more than it has been here, and that its cutoff value will generally be set empirically (with 50% being a natural jumping-off point).

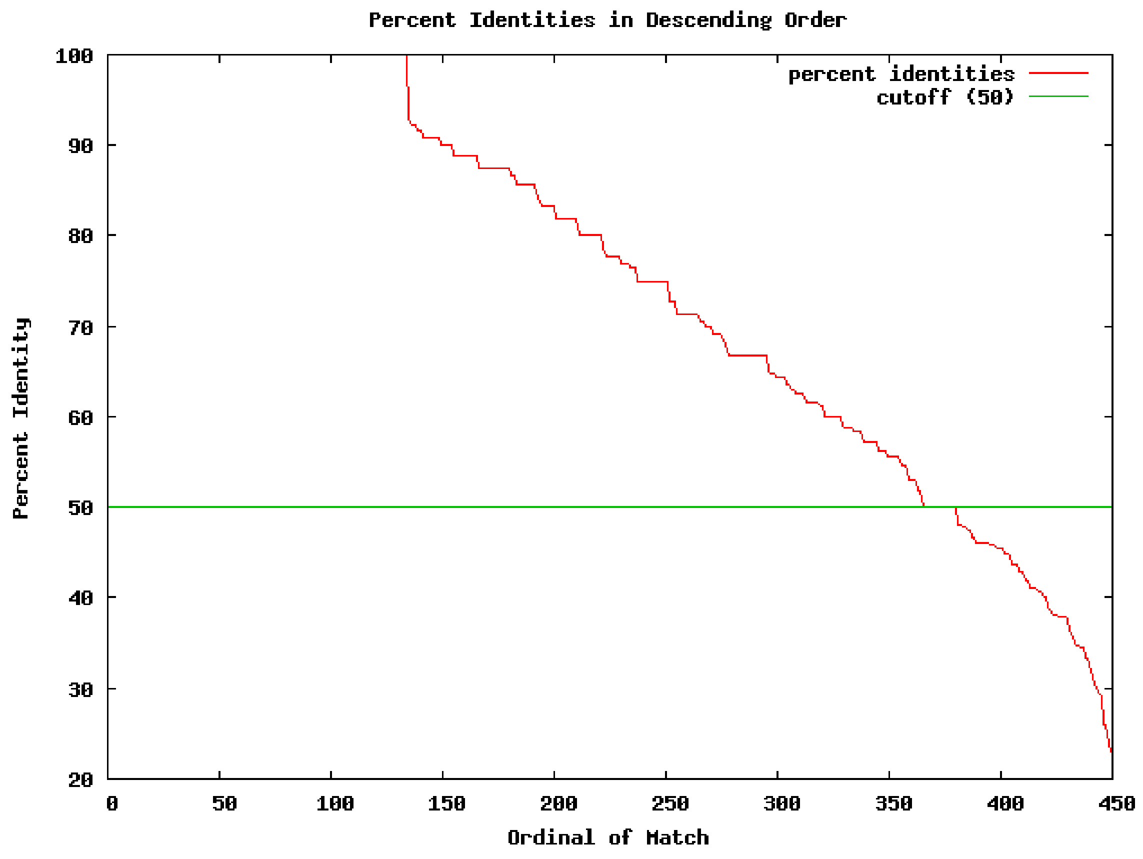


Figure 2.7: Percent Identity of Matches in Ascending Order

This discussion has explored the inner workings of the BLAST approach to plagiarism detection. It has described the method of action of each step in this process, and the various types of output produced at each of these stages. The meaning and usefulness of these results have also been assessed. Ultimately, in the light of several statistical properties of the document pair, the significance of certain constraints have been demonstrated. It is at this point, armed with an understanding of the inner workings of the process, and equipped with the ability to perform rudimentary analysis on its output, that one can finally proceed to a full-fledged analysis of the effectiveness of this process as a whole at the task of plagiarism detection. A full discussion of this analysis will occur in the following chapter.

Chapter 3

Analysis

3.1 Approach

The final task of this discussion is to investigate and analyze the effectiveness of the BLAST-based approach at detecting plagiarism. The simplest and most effective method of determining this effectiveness is to submit a number of document pairs to the algorithm's scrutiny (each with varying but known quantities of internal plagiarism) and assess the frequency with which the algorithm returns the correct answer. By submitting document pairs containing plagiarism that has been obfuscated to a variable but known degree, the robustness of the algorithm in the face of obfuscation can also be assessed.

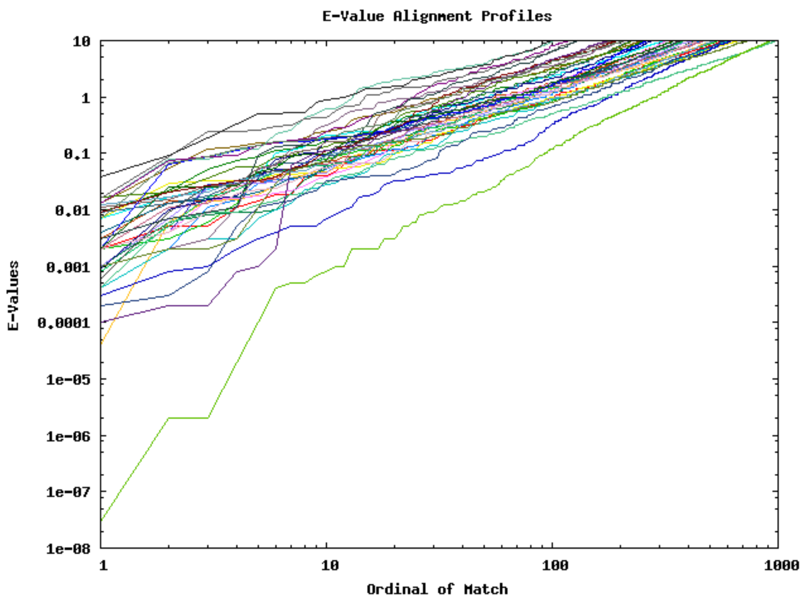
In this situation, however, such an assessment is not quite so easily accomplished. Specifically, the implementation developed in the previous chapter merely produces a large set of candidate alignments (regions of similarity between two documents that might represent instances of plagiarism), along with a set of visualizations and associated filters designed to allow a human operator to examine passages containing those cases of suspected plagiarism and make an educated determination as to whether the document pair actually contains plagiarism or not. Indeed, the vast majority of the candidate alignments produced by the above method represent nothing more than accidental similarity – as innocent as a single word occurring in both documents (potentially in completely different contexts).

In the real world, as mentioned in Chapter 1 of this discussion, the control and oversight of a human operator is required to make a final determination about the existence of plagiarism. This is true primarily because the cost of a false positive (if it were to lead to a real-world accusation) is too great to allow the chance for a computer mistake. However, in analyzing the performance of a particular algorithm (namely this one), a human operator is not only unnecessary; he is also anathema. In a nutshell, if one measures the accuracy of an algorithm that ultimately relies on a human operator to make its final distinction, one is not really measuring the robustness of the algorithm itself but rather the skill of the operator. For this reason, the implementation described in the previous chapter must be extended to form its own conclusions before it can be fully analyzed.

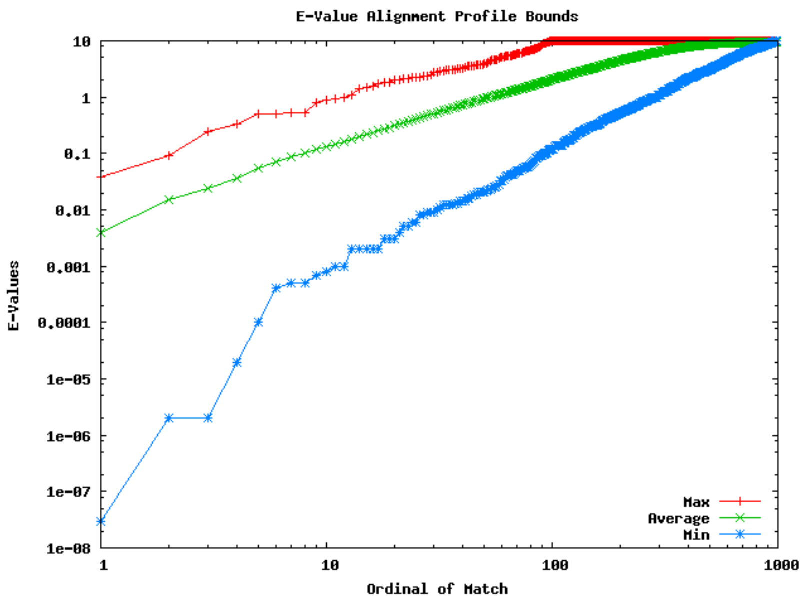
To further complicate matters, a reasonable conclusion about the existence of plagiarism cannot be gleaned directly from the candidate matches returned by the BLAST algorithm. As was just mentioned, BLAST returns a large number of candidate alignment, many of which represent nothing more than random coincidental similarity: a sort of background radiation that will inevitably occur between any two documents. In order to draw a reasonable conclusion about the existence of plagiarism, therefore, one must somehow cancel out this background radiation, and examine the (few) candidate matches that stand out in the face of the random clutter. A good sampling of this background similarity level should be readily gleaned by analyzing a large number of comparisons between documents known (or at least strongly assumed) to be dissimilar. To this end, a corpus of 10 document was constructed (each an article randomly selected from Wikipedia [17]). As each document is written on a completely unrelated subject[1], it is reasonable to assume without further manual analysis that there exists no more than trivial similarity between them. A full set of pairwise comparisons within this corpus will yield candidate alignments for 45 document pairs. This should be sufficient to establish a reasonable measure of background similarity levels.

It was established in the previous chapter that the most effective metric for determining the quality of a particular alignment seems to be its expectation value. Thus a logical starting point for examine

[1]Further details concerning these articles can be found in Appendix B



(a) Raw Alignment Profiles



(b) Alignment Profile Bounds

Figure 3.1: E-Value Alignment Profiles Within a Corpus

the background similarity levels is to analyze the e-value alignment profiles of the 45 background document pairs. Figure 3.1 presents the first two steps in this analysis. To begin with, the e-value alignment profiles of all 45 document pairs are plotted in Figure 3.1a. Both axes are plotted with logarithmic scales, for the same reasons as in the previous chapter. One can see that all of the alignment profiles of these document pairs tend to follow roughly the same trend.

Figure 3.1b attempts to quantify this trend somewhat. This plot presents the minimum, maximum and mean e-value at each match ordinate. A reasonable assumption is that the e-value alignment profile of a document pair containing no more than background similarity should fall generally within the bounds set by the maximum and minimum curves. A document pair containing some additional similarity beyond the background level should fall outside these bounds, at least at the low end of the abscissa. A document pair containing less similarity than the normal background level should conversely fall above the maximum bound to some degree.

In Figure 3.2, the e-value alignment profile of the hypothetical document pair discussed in the previous chapter is overlaid on the alignment profile bounds presented in Figure 3.1b. The low end of this alignment profile diverges from the background trend by many orders of magnitude. Recall that this document pair contains 3 identical sentences, as well as a number of shorter phrases with $> 50\%$ identity. In a real-world document pair, this level of similarity is very likely indicative of plagiarism. As the alignment profile with this similarity differs so markedly from the general levels of background similarity, it should be fairly straightforward to differentiate document pairs with a high likelihood of plagiarism from those with no such likelihood.

The final requirement before the computer can make its own determination as to whether a document pair contains plagiarism or not is to somehow quantify the degree to which the document pair's alignment profile diverges from the established norms. The area between the curve and the normal bounds would seem to be a logical way to quantify this. Since the orders of magnitude of the e-values, rather than their actual magnitudes, are generally being considered in this discussion, the difference of logarithms between the curve and the bounds is probably a more apt measure of the curve's divergence from the norms. In this investigation, a document pair's divergence from

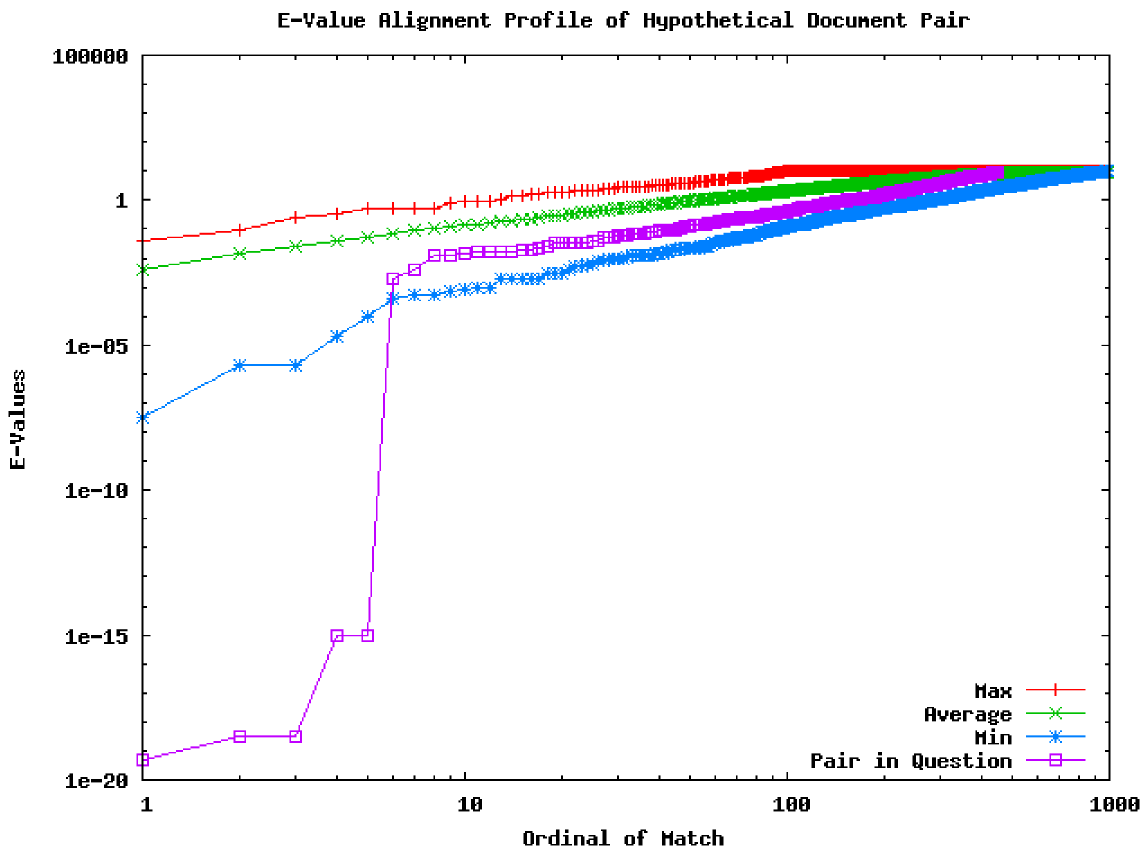


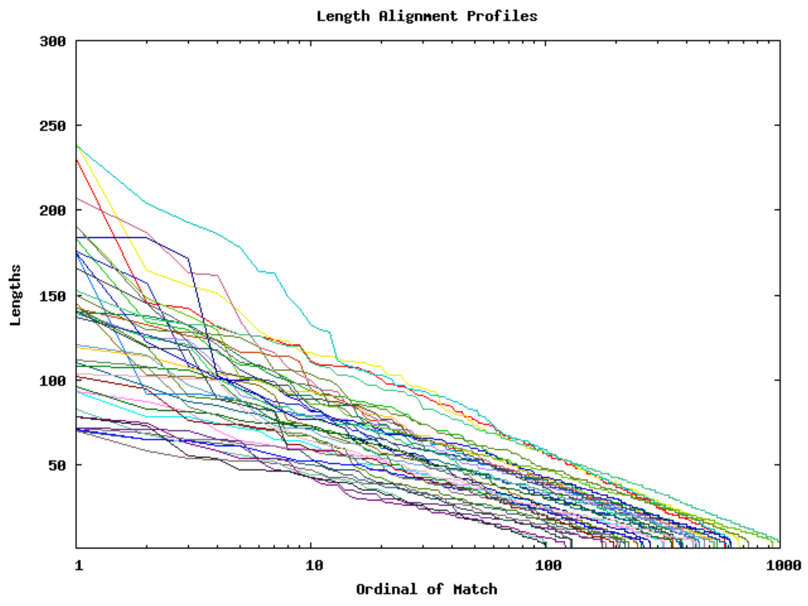
Figure 3.2: E-Value Alignment Profile of Hypothetical Document Pair

background similarity levels will be quantified into a *similarity score*.[2] This score will be calculated as the positive difference in the logarithms of the minimum bound and the document pair's alignment profile curve at each match ordinal for which the alignment profile lies below the normal bounds, plus the negative difference in the logarithms of the maximum bound and the document pair's alignment profile curve at each match ordinal for which the alignment profile lies above the normal bounds. Using this metric, a document pair that contains greater that normal similarity should receive a positive similarity score, and document that contains less than normal similarity should receive a negative one. A document whose e-value alignment profile falls entirely within the established normal bounds should receive a similarity score of zero. For reference, the similarity score of the hypothetical document pair discussed above is 135.2 using this metric.

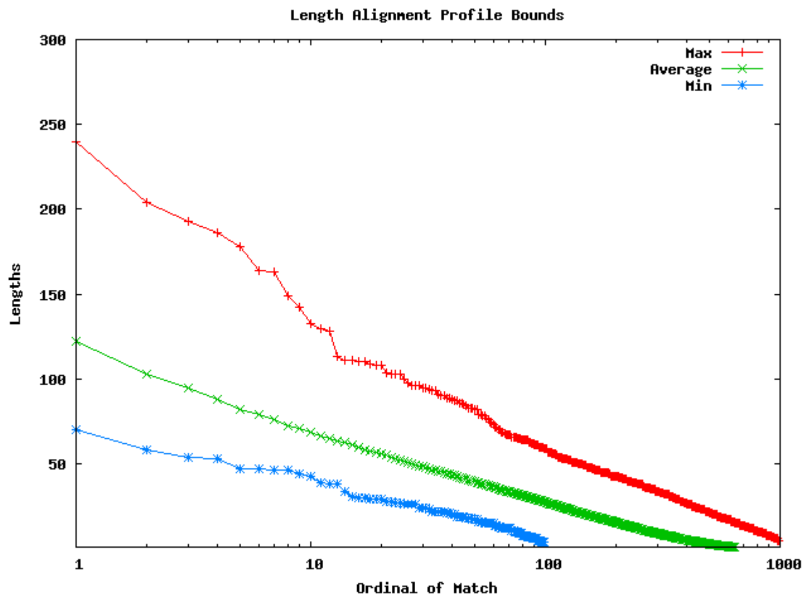
It should be noted that although the e-value alignment profile is probably the best parameter on which to base such a metric, it is not the only conceivable one. For example, e-value was only one of three parameters used to filter the results discussed in the previous chapter; alignment length and percent identity were used as well. One should, then, consider the alignment length and percent identity alignment profiles as bases for other possible metrics. Figure 3.3 presents the same background similarity analysis for alignment length as was performed for e-value in Figure 3.1. Again, Figure 3.3a presents the raw alignment profiles for each of the 45 document pairs, and Figure 3.3b displays the maximum, mean, and minimum of this distribution at each match ordinal. The abscissa is plotted using a logarithmic scale in order to accentuate the low end, where (as was also the case with the e-value alignment profile) a deviation is most likely to occur.

Figure 3.4 presents the length alignment profile of the hypothetical document pair overlaid on these normal bounds. Even though this document pair contains substantial similarity, its alignment profile does not deviate substantially from the normal range representative of background similarity. In fact, the similarity score for the document pair under this metric (calculated using the same formula as proposed above for the e-value metric, with the exception that absolute difference is used instead

[2] A number of algorithms discussed in the literature compute some manner of *similarity score* (although it may not always be called that) to quantify the similarity of a pair of documents. The *similarity score* used in this discussion is peculiar to the BLAST-based algorithm in question, and cannot be compared to any other similarity metrics discussed in other papers.



(a) Raw Alignment Profiles



(b) Alignment Profile Bounds

Figure 3.3: Length Alignment Profiles Within a Corpus

of difference of logarithms, since the data here are plotted with a linear ordinate) is actually −14. This score tends to indicate that the hypothetical document pair actually has less than background similarity, leading to a conclusion that the document pair contains no instances of plagiarism. Thus, it would seem that the alignment length profile is not a particularly effective metric. It is fairly easy to imagine, too, that the percent identity alignment profile would produce similarly poor results. As a candidate alignment could easily have perfect identity but diminutive length, a large number of high percent identity alignments is likely a very poor measure of the actual non-background similarity within a document pair. By a similar argument, none of the other parameters that describe the candidate alignments lend themselves to make an effective metric. It would seem that one is left with only the e-value alignment profile for this purpose.

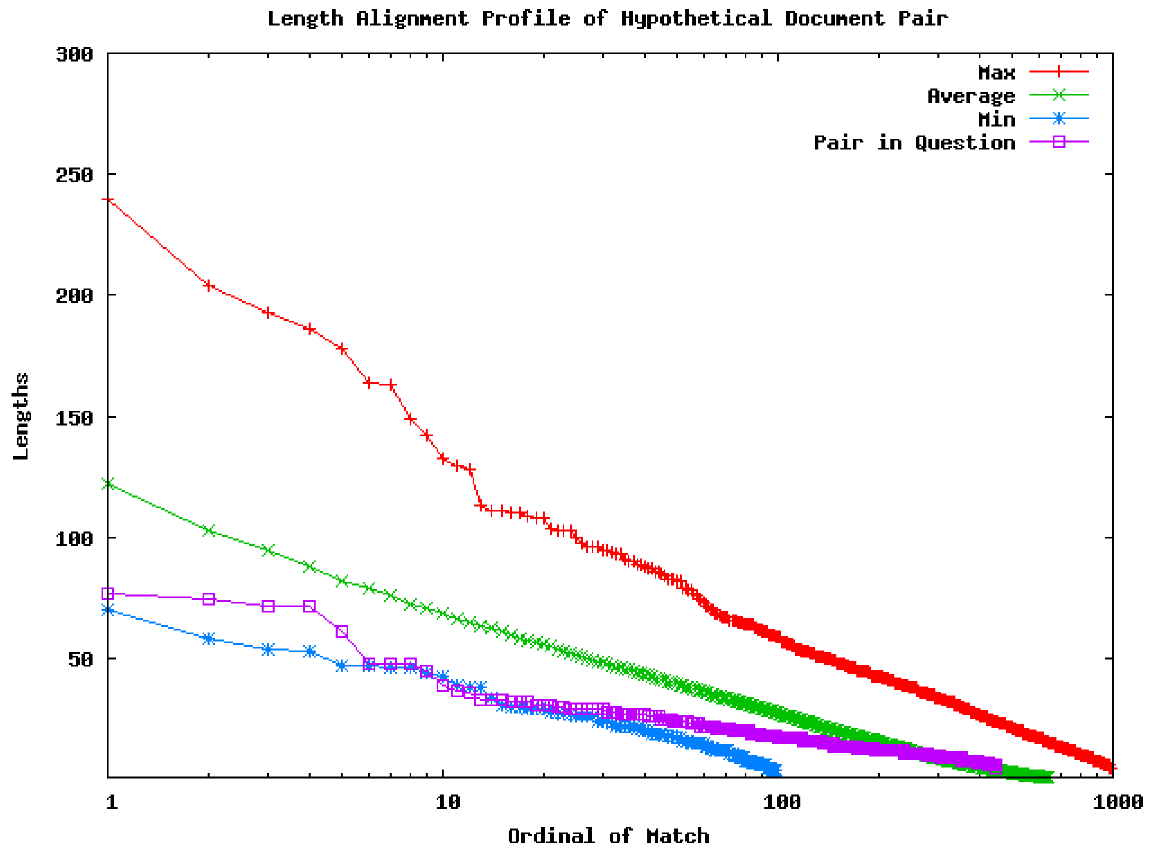


Figure 3.4: Length Alignment Profile of Hypothetical Document Pair

It should be noted that no threshold has been set for this metric to differentiate between document pairs believed to contain plagiarism and those not. It takes little imagination to believe that a similarity score of ≤ 0 is clearly an indicator of the absence of plagiarism, and a similarity score as high as 135 (as in the hypothetical case discussed above) is an indicator of its presence. Where exactly to draw the line between is not particularly well defined. Ultimately, the determination of this threshold must be up to the operator, who needs to make a decision on the tradeoff between false negatives and false positives. Since this investigation does not wish to make a judgement in this matter, no specific threshold will be selected, and the analysis in the following section will focus primarily on the behavior of the similarity score parameter as the amount and level of obfuscation of plagiarism in the document pair vary.

3.2 Results

Ultimately, the purpose of this discussion is to determine how effective the BLAST-based algorithm is at detecting plagiarism. To accomplish this, it is necessary to assess its accuracy at detecting plagiarism in document pairs where the degree of plagiarism is known. In the previous section, a metric has been established (the similarity score) that allows the algorithm to make its own determination about the likely plagiarism content of a document pair, without the intervention of a human operator. This was necessary to focus the assessment on the performance of the algorithm, rather than the skill of the particular human operator. Now, all that remains is to apply this algorithm, using that metric, to a number of document pairs with known plagiarism content, and evaluate the results.

To obtain a collection of document pairs with known levels of plagiarism, one can start with a document pair containing no more than baseline similarity and introduce artificial plagiarism in a regular, parameterized manner. For this discussion, the first two documents of the baseline corpus were chosen to create this initial document pair (they are both reasonably long,[3] about the same length, and contain normal, baseline levels of similarity). To introduce artificial plagiarism into this document pair, a substring of length n was copied from one document and inserted into the middle

[3]on the order of $10,000$ characters each. More details can be found in Appendix B.

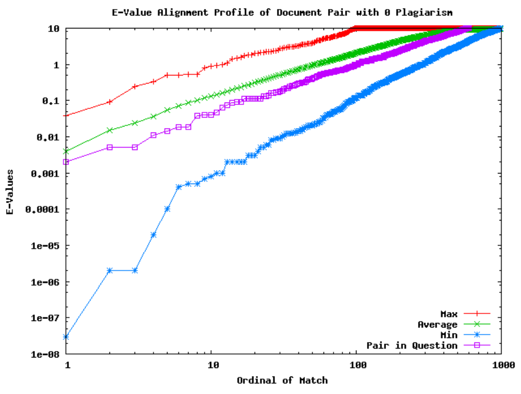
of the other document. By varying the parameter n, the amount of exact-copy plagiarism introduced into the document pair can be controlled and quantified. (Note, at this stage of the discussion only exact-copy plagiarism is being considered. Obfuscated plagiarism will have its turn in short order.)

The parameter n was varied from 0 to 200, in increments of 10. Figure 3.5 shows the e-value alignment profiles of several of the document pairs along this range. Figure 3.5a shows the alignment profile for the original document pair containing no artificial plagiarism. Figure 3.5b shows the alignment profile for the document pair infused with a 100 character string of plagiarized text, and Figure 3.5c shows that alignment profile for the document pair infused with the full 200 characters string of plagiarized text.

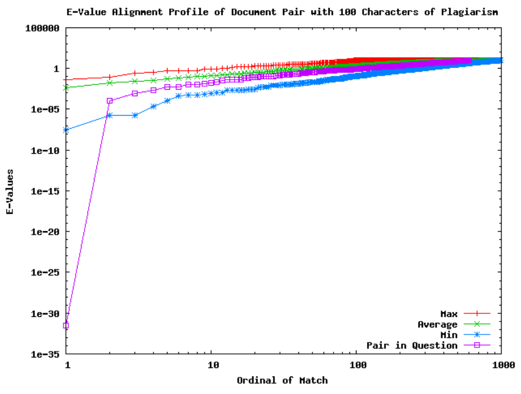
At the low end, the document pair depicted by Figure 3.5a has a similarity score of 0. The entirety of the alignment profile curve falls within the normal bounds. At the other end of the spectrum, Figure 3.5c represents a similarity score of 105.5. For reference, Figure 3.5b, at the middle of the range, represents a similarity score of 59.5. The question, then, is two-fold: how does the similarity score increase across the range, and at what threshold value does it become a meaningful indicator of plagiarism?

Figure 3.6 should answer both of these questions. This figure depicts the similarity score of each of these document pairs as the plagiarism length parameter varies along its range. It is apparent that the similarity score tends to increase in a linear fashion relative to the plagiarism length parameter, with the shortest amount of plagiarism triggering a response at all being a string as short as 30 characters. A similarity score of about 20 corresponds with an instance of plagiarism consisting of an exact copy of text about 50 characters long. As such a textual similarity is extremely unlikely to occur randomly in two independent pieces of writing, this level of similarity in real world documents has a high likelihood of representing actual plagiarism. Therefore, it seems reasonable to conclude that 20 represents an appropriate similarity score threshold for making a final determination about content of plagiarism. (In fact, the threshold could probably be even lower, but why be greedy?)

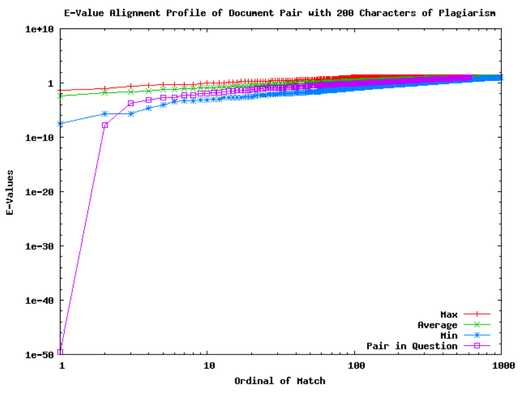
Thus far, this analysis has focused on the BLAST-based algorithm's ability to detect exact-copy plagiarism. The results have been promising (document pairs with plagiarized passages as short at



(a) With No Added Plagiarism



(b) With 100 Characters Copied



(c) With 200 Characters Copied

Figure 3.5: Alignment Profiles with Varying Amounts of Plagiarism

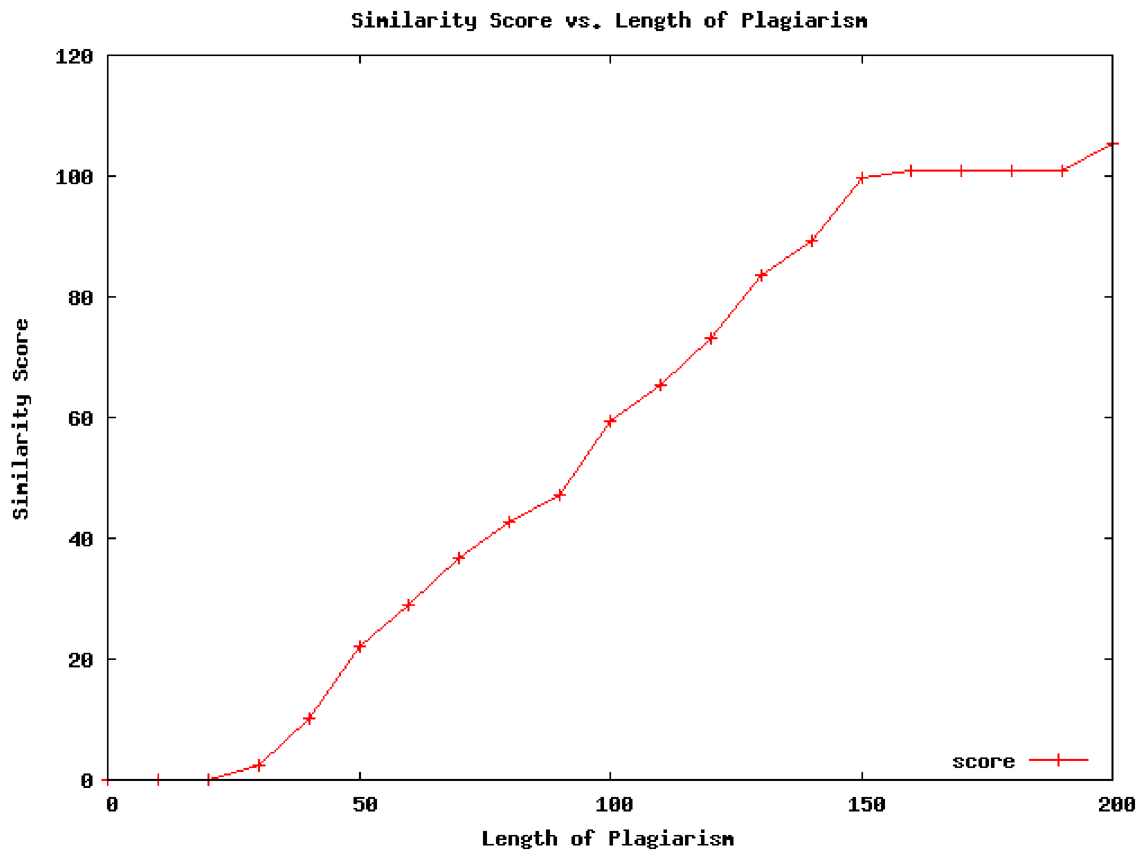


Figure 3.6: Similarity Score with Varying Amounts of Plagiarism

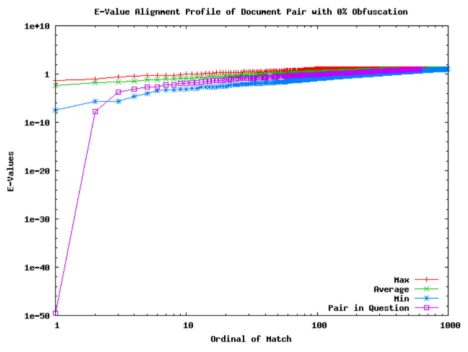
30 characters have been identified as having greater-than-normal similarity). However, that part of the analysis has carefully avoided the meat of the problem: detecting plagiarism that has been obfuscated. Ultimately, the purpose of this discussion is to determine how effective the BLAST-based algorithm is at detecting plagiarism that has been obfuscated.

To measure this ability, one requires a set of document pairs containing known levels of plagiarism that have been obfuscated to a known degree. In a similar manner as above, such a set of document pairs can be generated from a pair of documents containing only baseline similarity. Again, a substring (of length n) of one document is copied and inserted into the other document to create an instance of exact-copy plagiarism parameterized by n. Now, this process will be taken one step further. This copied passage can be artificially obfuscated in a regular manner to a degree varying by a parameter m to produce a document pair with obfuscated plagiarism varying by the parameters n and m. To effect this artificial obfuscation, approximately $m\%$ of the characters in the plagiarized passage are replaced by a randomly selected character. This produces a plagiarized string of length n with $m\%$ character identity.

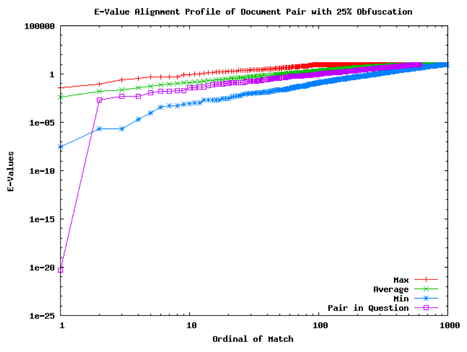
This technique of artificially generating obfuscation seems to be a reasonable approximation of the obfuscation techniques discussed in Chapter 1. Some of these techniques (for example, changing number or tense) alter a small number of characters strewn throughout a sentence. Others, (for example, replacing a word with a synonym) alter a cluster of characters, but leave the rest of the sentence unchanged. These two technique represent the vast majority of ways in which exact-copy plagiarism is likely to be obfuscated in the real world.[4] As the characters altered by the artificial obfuscation technique are randomly distributed throughout the plagiarized passage, there is a high likelihood that some of them will clump together and simulate synonym replacement, and some of them will spread out and simulate tense or number changes.

The parameter m was varied from 0 to 100 (as one would expect of a percentage), in increments of 5 (the establish the same number of data points as in the previous analysis). In this case, the parameter n was fixed at 200, so that only obfuscation percentage is a factor in resulting data.

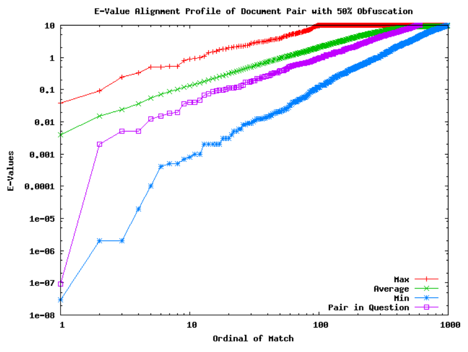
[4]The third major obfuscation technique mentioned in Chapter 1, altering the structure of a sentence, is extremely unlikely to affect the performance of BLAST, as large phrases remain syntactically unaltered. Thus, it was deemed unnecessary to combine this obfuscation technique into the artificial approximation discussed herein.



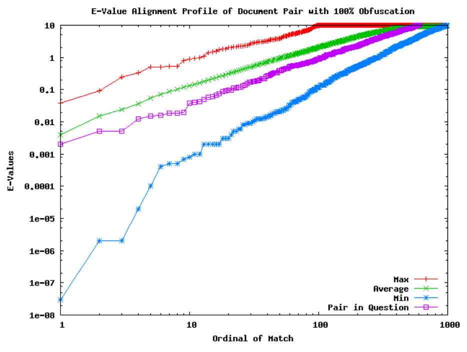
(a) 0% Obfuscation



(b) 25% Obfuscation



(c) 50% Obfuscation



(d) 100% Obfuscation

Figure 3.7: Alignment Profiles with Varying Amounts of Obfuscation

Figure 3.7 shows the e-value alignment profiles of several of the document pairs along this range. Figure 3.7a shows the alignment profile for the document pair containing 200 characters of artificial plagiarism with no obfuscation. Figure 3.7b shows the alignment profile for the same document pair after 25% of the plagiarized text has been obfuscated, Figure 3.7c shows the same with 50% obfuscation, and Figure 3.7d shows the same with 100% obfuscation (i.e. no characters remain from the original instance of plagiarism). At the low end, the document pair depicted by Figure 3.7a has a similarity score of 105.5. The document pair with 25% obfuscation has a similarity score of 33.6, and the document pair with 50% obfuscation has a similarity score of 3.1. Above 50% obfuscation, the similarity score falls to 0 and remains there.

Figure 3.8 addresses the behavior of the similarity score parameter as the obfuscation percentage increases. It appears from this plot that the similarity score decreases roughly linearly as obfuscation percentage increases. In general, this would tend to indicate the the BLAST-based algorithm is reasonably robust in the face of obfuscation. A constant additional amount of obfuscation will only reduce the similarity score by a constant amount, regardless of where it already falls. In this case, and this is likely a typical response, a positive similarity score is registered up to 50% obfuscation. However, this does not immediately mean that the BLAST-based algorithm can detect plagiarism that has been 50% obfuscated. Recall that the similarity score of the 50% document pair is only 3.1. In the previous phase of this analysis, 20 was established as a reasonable threshold to confidently return an assessment of plagiarism, being associated with a 50-character-long instance of exact-copy plagiarism. A similarity score of 3.1 is more commensurate with a 30-character-long instance, much more likely to occur by random chance than one of length 50. This is not to say that 3.1 is a meaningless result for a similarity score, just that it is not a particularly confidence-inspiring one. A similarity threshold of 20 applied to Figure 3.8 yields a cutoff of 40% obfuscation. This is still a significant result. In this light, it is reasonable to conclude that the BLAST-based approach is capable of confidently detecting an instance of plagiarism even when up to 40% of it has been obfuscated.

This analysis is the crux of this discussion. Figure 3.6 demonstrates that the BLAST-based algorithm is capable of detecting instances of exact copy plagiarism as short as 30 characters in length,

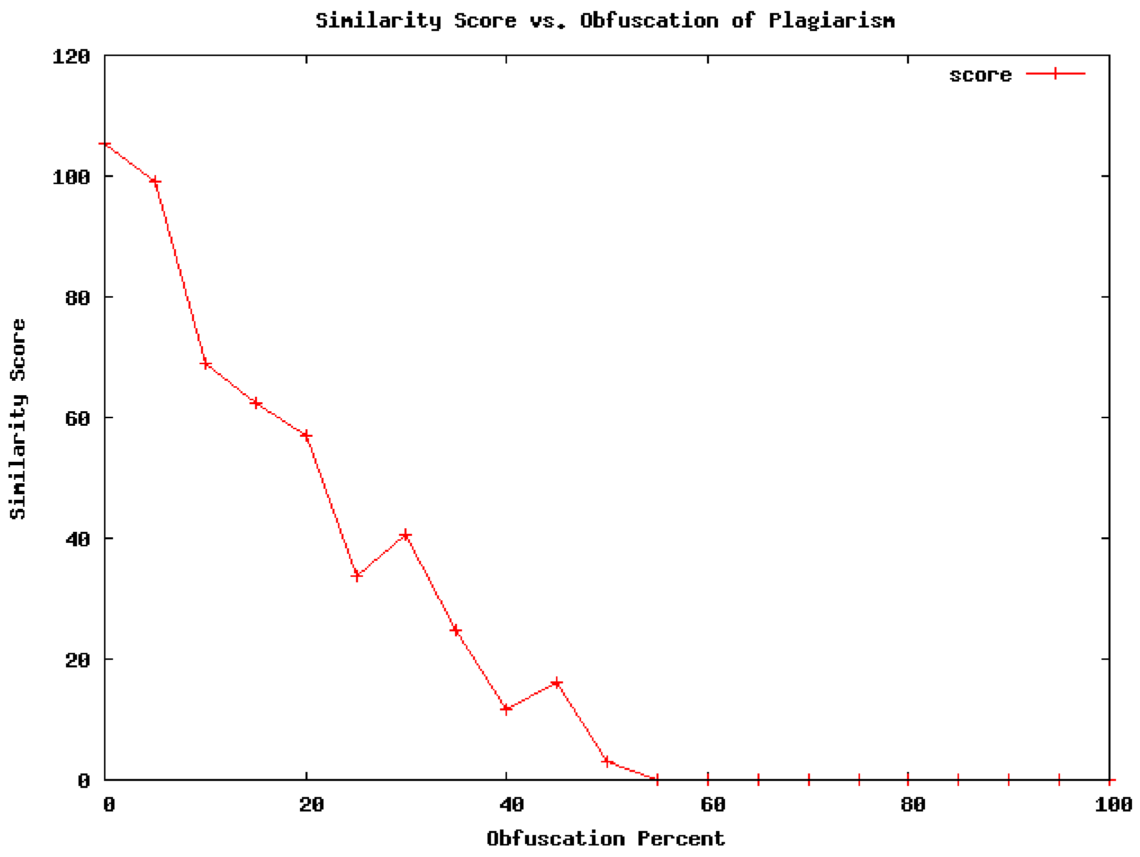


Figure 3.8: Similarity Score with Varying Amounts of Obfuscation

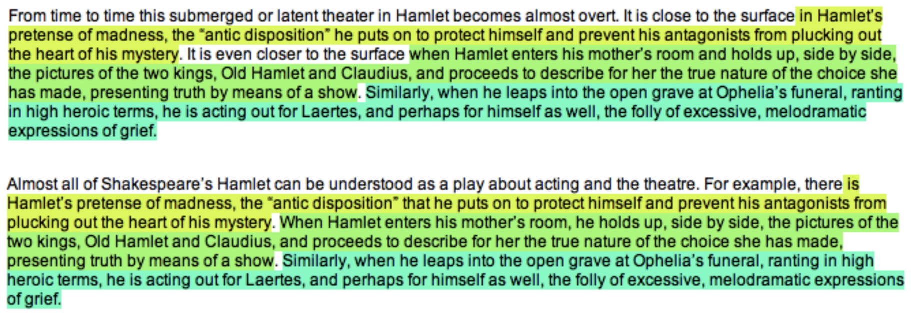
and that its confidence increases linearly with the length of the plagiarism. Figure 3.8 demonstrates that the BLAST-based algorithm is also capable of detecting instances of obfuscated plagiarism that have been altered to the point where it retains only 50% character similarity to the original passage, and that it is confidently able to do so in situations with at least 60% character similarity. This seems to be an acceptable level of efficiency to demonstrate to merit of the BLAST-based approach.

3.3 Applications

This section will examine the application of the BLAST-based algorithm to potential, real-world instances of plagiarism in actual text. The theoretical analysis above is quite alright for developing an insight into the theoretical behavior and limits of this method. However, when the BLAST-based algorithm is confronted with real examples of potential plagiarism is truly where the rubber meets the road. In addition to displaying great potential in the theoretical analysis above, it is important for this technique to produce useful, meaningful results in the real world.

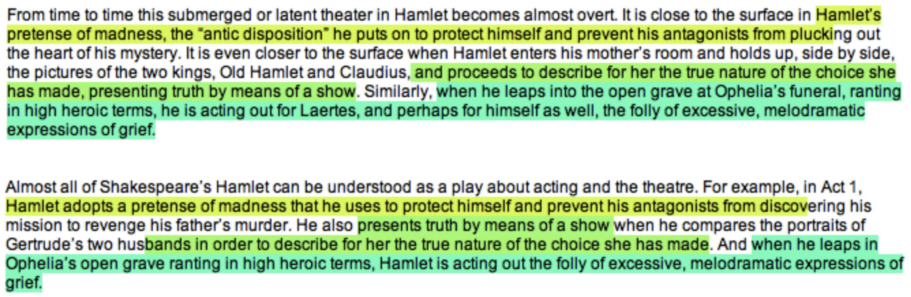
Virtually every college and university worth its salt seems to have published some manner of plagiarism-avoidance guide. Many of these provide short examples of improperly-used sources. Princeton University's publication on appropriate conduct contains a large discussion of academic dishonesty in general, and plagiarism in particular. Specifically, this publication includes several short (paragraph-length) examples of plagiarism in varying forms - ranging from direct, blatant copying to elegant paraphrasing [18].

The examples provided in Princeton's publication consist of an original passage about *Hamlet*, followed by three plagiarized versions. The similarities between the original and each of these versions are displayed in Figure 3.9. The first example (Figure 3.9a contains large sections of text directly copied from the original source. The second example (Figure 3.9b) contains somewhat more paraphrasing that the first example, although it retains a number of shorter copied phrases along with the basic structure of the original. This example is likely indicative of an instance of exact-copy plagiarism (like in the first example) that has been obfuscated by some small degree of paraphrasing. The third example (Figure 3.9c) is entirely rewritten, and maintains little textual similarity to the original, although the ideas and examples used in the original source have been used with little modification [18].


From time to time this submerged or latent theater in Hamlet becomes almost overt. It is close to the surface in Hamlet's pretense of madness, the "antic disposition" he puts on to protect himself and prevent his antagonists from plucking out the heart of his mystery. It is even closer to the surface when Hamlet enters his mother's room and holds up, side by side, the pictures of the two kings, Old Hamlet and Claudius, and proceeds to describe for her the true nature of the choice she has made, presenting truth by means of a show. Similarly, when he leaps into the open grave at Ophelia's funeral, ranting in high heroic terms, he is acting out for Laertes, and perhaps for himself as well, the folly of excessive, melodramatic expressions of grief.

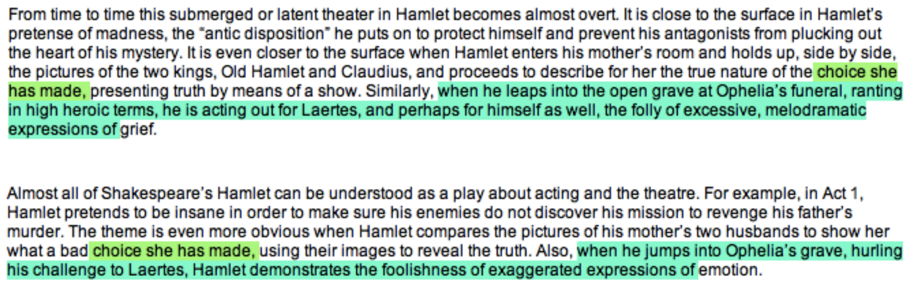
Almost all of Shakespeare's Hamlet can be understood as a play about acting and the theatre. For example, there is Hamlet's pretense of madness, the "antic disposition" that he puts on to protect himself and prevent his antagonists from plucking out the heart of his mystery. When Hamlet enters his mother's room, he holds up, side by side, the pictures of the two kings, Old Hamlet and Claudius, and proceeds to describe for her the true nature of the choice she has made, presenting truth by means of a show. Similarly, when he leaps into the open grave at Ophelia's funeral, ranting in high heroic terms, he is acting out for Laertes, and perhaps for himself as well, the folly of excessive, melodramatic expressions of grief.

(a) Original vs. Example 1


From time to time this submerged or latent theater in Hamlet becomes almost overt. It is close to the surface in Hamlet's pretense of madness, the "antic disposition" he puts on to protect himself and prevent his antagonists from plucking out the heart of his mystery. It is even closer to the surface when Hamlet enters his mother's room and holds up, side by side, the pictures of the two kings, Old Hamlet and Claudius, and proceeds to describe for her the true nature of the choice she has made, presenting truth by means of a show. Similarly, when he leaps into the open grave at Ophelia's funeral, ranting in high heroic terms, he is acting out for Laertes, and perhaps for himself as well, the folly of excessive, melodramatic expressions of grief.

Almost all of Shakespeare's Hamlet can be understood as a play about acting and the theatre. For example, in Act 1, Hamlet adopts a pretense of madness that he uses to protect himself and prevent his antagonists from discovering his mission to revenge his father's murder. He also presents truth by means of a show when he compares the portraits of Gertrude's two husbands in order to describe for her the true nature of the choice she has made. And when he leaps in Ophelia's open grave ranting in high heroic terms, Hamlet is acting out the folly of excessive, melodramatic expressions of grief.

(b) Original vs Example 2

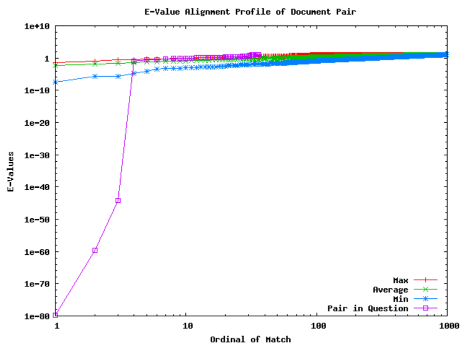

From time to time this submerged or latent theater in Hamlet becomes almost overt. It is close to the surface in Hamlet's pretense of madness, the "antic disposition" he puts on to protect himself and prevent his antagonists from plucking out the heart of his mystery. It is even closer to the surface when Hamlet enters his mother's room and holds up, side by side, the pictures of the two kings, Old Hamlet and Claudius, and proceeds to describe for her the true nature of the choice she has made, presenting truth by means of a show. Similarly, when he leaps into the open grave at Ophelia's funeral, ranting in high heroic terms, he is acting out for Laertes, and perhaps for himself as well, the folly of excessive, melodramatic expressions of grief.

Almost all of Shakespeare's Hamlet can be understood as a play about acting and the theatre. For example, in Act 1, Hamlet pretends to be insane in order to make sure his enemies do not discover his mission to revenge his father's murder. The theme is even more obvious when Hamlet compares the pictures of his mother's two husbands to show her what a bad choice she has made, using their images to reveal the truth. Also, when he jumps into Ophelia's grave, hurling his challenge to Laertes, Hamlet demonstrates the foolishness of exaggerated expressions of emotion.

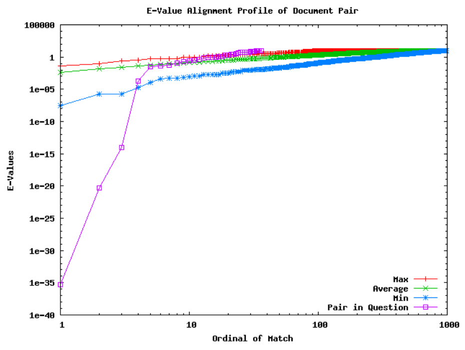
(c) Original vs Example 3

Figure 3.9: Similarities Within Several Examples of Plagiarism

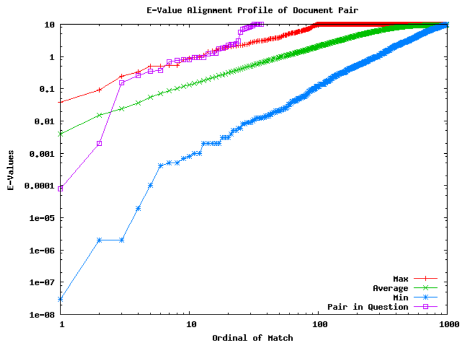
Figure 3.10 presents the alignment profiles of these three examples with the original source. In Figure 3.10a, the example that has been copied wholesale from the original demonstrates marked divergence from background similarity at the low end. This document pair has a similarity score of 368.7, and would clearly be classified as plagiarism. In Figure 3.10b, the example that contains some



(a) Original vs. Example 1



(b) Original vs Example 2



(c) Original vs Example 3

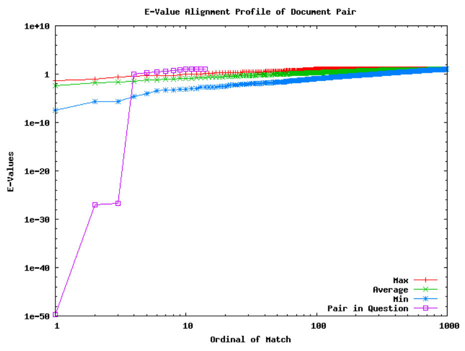
Figure 3.10: Alignment Profiles for Several Examples of Plagiarism

paraphrasing and some direct copying still clearly differs from the baseline norms at the low end, but to a lesser extent that does the first example. The second example received a similarity score of 103.8 in comparison with the original document. This would also likely be classified as an instance of plagiarism. However, in Figure 3.10c, the third example does not diverge from the baseline at the low end. This document pair received a similarity score of -15.5 and thus would certainly not be classified as plagiarism by the BLAST-based algorithm.

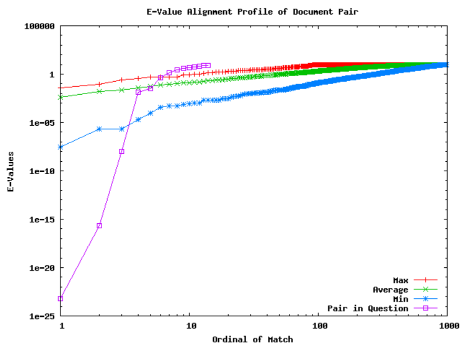
The BLAST-based algorithm has correctly classified the first two examples with reasonable confidence. Its failure to correctly classify the third document is most likely not indicative of a defect in the algorithm, but rather is simply an example of well-obfuscated plagiarism. This example has been completely rewritten and maintains only semantic similarity to the original document. Because the BLAST-based algorithm is unable to comprehend the semantic meaning of a passage, it can only make comparisons on the basic of syntactic, textual similarity. In this case, the plagiarized passage has been obfuscated / rewritten to a sufficient degree that no more than negligible textual similarity remains. As no computer algorithm at this time can actually understand the meaning of language, it is a fair assessment that other plagiarism detection solutions (such as those discussed in Chapter 1) will run into the same problems. Detecting the plagiarism in this third example is likely a problem to which a solution is currently unattainable.

Princeton University is not the only academic institution to proffer examples of plagiarism as warnings of what not to do. Dartmouth College has also published an extensive discussion on the proper use of sources in academic work, which contains a similar discussion of plagiarism, accompanied by examples [19]. These examples follow the same general form as those from Princeton. There is one original source, again about a paragraph in length, from which three other paragraphs have been plagiarized. The first example consists most of exact copies, interspersed with a few original words. The second example consists of general paraphrasing, with a decent number of phrases copied from the original appearing throughout. The third example consists of a completely reworded passage with little textual similarity to the original (although it does directly copy of the original source's ideas).

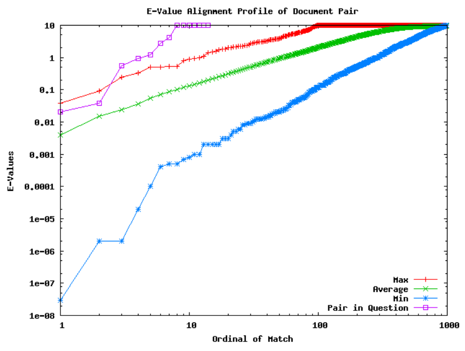
Figure 3.11 presents the alignment profiles of these three examples with the original source. In



(a) Original vs. Example 1



(b) Original vs Example 2



(c) Original vs Example 3

Figure 3.11: Alignment Profiles for Several Additional Examples of Plagiarism

Figure 3.11a, the example consisting almost entirely of exact copies displays significant divergence from the baseline similarity at the low end (as does the first example document pair in the Princeton set). This document pair has a similarity score of 180.1, and should again be classified as plagiarism. In Figure 3.11b, the example with some paraphrasing and some direct copying again differs from the baseline, but again to a smaller degree. This document pair scores on 58.5. Although this score is lower than the others already discussed, it is still likely high enough to allow for a judgement of plagiarism with reasonable confidence. In the third example (displayed in Figure 3.11c), as is the case above, no noticeable similarity was detected. This document pair's alignment profile does not fall below the baseline curve at all, and in fact received a similarity score of -21.4. Again, this would not be considered plagiarism by any means.

Tests using both of these sets of plagiarism examples demonstrate that the BLAST-based algorithm is reasonably effective at detecting such examples of real-world plagiarism. However, in order to demonstrate that this approach is not overly prone to judge a document pair as plagiarism, one must also consider the reverse case. This is the case where two documents, written on the same subject and likely including a number of similar terms, do not contain any genuine instances of plagiarism. Ideally, the BLAST-based algorithm should not detect substantially more than the previously established baseline levels of similarity in such a case.

Unfortunately, it is substantially easier to find examples of document pairs that do contain plagiarism than it is to find examples of document pairs on the same topic that do not. It is important to note that the lack of plagiarism sought in this case is not just proper use of citations, but rather a complete lack of copied text regardless of citations. Since the BLAST-based approach developed above does not take citations into account in any respect, it will classify properly cited instances of textual similarity as potential plagiarism, leaving the task of verifying the correctness of the citations to the human operator.

To some extent, the third example documents discussed above, which contain paraphrased text that has not been properly attributed, constitute an example of a document pair not containing plagiarism for the purposes of this particular inquiry. Since adding proper citations would effectively remove the plagiarism from these documents, and the newly-added citations will not be considered by the BLAST-based algorithm, it is immaterial to this discussion whether or not the documents contain appropriate citations. These documents (the third examples) can thus be considered a reasonable case for the correctness of the BLAST-based algorithm's ability to judge a document pair as not containing plagiarism.

The examples presented in this section represent instances of plagiarism (or, potentially, instances of not-plagiarism) that have been held up by several leading academic institutions as guidelines for what to do and what not to do where plagiarism is concerned. The BLAST-based algorithm developed above has been able to assess the likely plagiarism content of each of these example pairs with a reasonable degree of confidence, expressed via the magnitude of the similarity score. As one would expect, this confidence has decreased as the examples have become increasingly obfuscated. Overall, the BLAST-based algorithm has performed as expected against real-world scenarios.

Conclusion

Plagiarism is the act of stealing another's ideas and claiming them as one's own.[5] In a written document, this will generally involve stealing some amount of another author's words. If two documents (roughly) share the same words, in the same order, to a sufficient degree, they are vulnerable to detection by computer algorithms that analyze the text of the two documents. Most algorithms that currently exist to perform this comparison seem susceptible to small perturbations in the document text - changes that are easy to make with little intellectual effort, yet stymie the computer's ability to see the similarity between the documents in question. Developing an algorithm that can better withstand these tricks is the primary goal of this discussion.

It seems that little progress has been made in the last decade to the particular problem of detecting plagiarism, while huge advances have been made in other fields of computer science. In light of this, it seems reasonable that a technique that has been developed for some other application in some other field could also prove useful for plagiarism detection. The Basic Local Alignment Search Tool, a tool for identifying similarity between DNA and protein sequences and one of the most widely-used algorithms in the booming field of bioinformatics, suggests itself as a possible candidate for such an application. This algorithm can readily be adapted to analyze sequences of English text, and with slightly more effort, its output can be processed and visualized in a manner amenable to comparing two such documents and analyzing their potential plagiarism content.

The effectiveness of this proposed algorithm can be analyzed in several ways. Most simply, it can

[5]In many cases, the line between plagiarism and legitimate use is defined by the proper attribution of borrowed material through citations. In this discussion, assessing the propriety of citations is left as an exercise for the reader, and the algorithms discussed herein focus solely on identifying instances of similarity between documents that might conceivably require citation.

be run on artificially-constructed examples of inter-document similarity. As the amount of similarity introduced into these documents can be precisely controlled, theoretical responses of the algorithm can be readily determined. It would appear from this analysis that this algorithm is quite responsive to even low-level amounts of similarity between documents. Even when this artificially similarity is intentionally disguised (by replaced a percentage of the similar characters with random ones) this algorithm is able to detect the latent similarity with confidence when as little as 60% of the original characters remains. Additionally, when run on several real-world examples of plagiarism (examples used by colleges and universities in discussions of academic honesty) the algorithm correctly identifies the existence of plagiarism in all situations except where no vestiges of the original text remain. In this light, it would appear that the algorithm developed herein displays great potential for use in the detection of plagiarism, and that, at very least, further research and refinement is warranted.

Bibliography

[1] E. Babbie. “Plagiarism,” [Online Document], 2005, [cited 2008 April 6], Available HTTP: `http://www1.chapman.edu/~babbie/plag00.html`

[2] E. Dionne, Jr. “Biden Admits Plagiarism in School But Says It Was Not ‘Malevolent,’” in New York Times, [Online Document], 1987 September 18, [cited 2008 April 6], Available HTTP: `http://query.nytimes.com/gst/fullpage.html?res=9B0DE3DB143FF93BA2575AC0A961948260`

[3] A. Smith. “Plagiarism in Journalism,” in The Daily Collegian, [Online Document], 2007 March 16, [cited 2008 April 29], Available HTTP: `http://media.www.dailycollegian.com/media/storage/paper874/news/2007/03/16/EditorialOpinion/Plagiarism.In.Journalism-2777803.shtml`

[4] R. Lukashenko, V. Graudina, and J. Grundspenkis. “Computer-based plagiarism detection methods and tools: an overview,” in International Conference on Computer Systems and Technologies, 2007.

[5] Australian Universities Teaching Committee. “Plagiarism Detection Software: How Effective Is It?” [Online Document], 2002, [cited 2008 April 7], Available HTTP: `http://www.cshe.unimelb.edu.au/assessinglearning/03/Plag2.html`

[6] D. R. White and M. S. Joy. “Sentence-based natural language plagiarism detection,” in J. Educ. Resour. Comput., 2004, vol. 4, p. 2.

[7] T. Lancaster, F. Culwin. “A Visual Argument for Plagiarism Detection Using Word Pairs,” [Online Document], 2004 April, [cited 2008 April 7], Available HTTP: `http://www.jiscpas.ac.uk/documents/abstracts/2004abstract15.pdf`

[8] “Advanced Document Search and Analysis Software,” [Online Document], 2007, [cited 2008 April 7], Available HTTP: `http://www.copycatchgold.com/`

[9] S. Schleimer, D. S. Wilkerson, and A. Aiken. “Winnowing: local algorithms for document fingerprinting,” in SIGMOD ’03: Proceedings of the 2003 ACM SIGMOD international conference on Management of data, 2003.

[10] Rensselaer-Wadsworth Bioinformatics Center. “The Bioinformatics Center at Rensselaer and Wadsworth,” [Online Document], [cited 2008 April 29], Available HTTP: `http://www.bioinfo.rpi.edu/`

[11] J. Craig Venter Institute. “What’s a Genome?” [Online Document], 2003 January 15, [cited 2008 April 29], Available HTTP: `http://www.genomenewsnetwork.org/resources/whats_a_genome/Chp4_1.shtml`

[12] S. Altschul, et. al. “Basic Local Alignment Search Tool,” in J. Mol. Bio, 1990, vol. 215, pp. 403 – 410.

[13] T. F. Smith and M. S. Waterman. “Identification of Common Molecular Subsequences,” in J. Mol. Bio, 1981, vol. 147, pp. 195 – 197.

[14] Internation Union of Pure and Applied Chemistry. “Sequence Manipulation Suite: IUPAC codes,” [Online Document], 2000, [cited 2008 April 29], Available HTTP: `http://www.bioinformatics.org/sms2/iupac.html`

[15] National Center for Biotechnology Information. “FASTA format description,” [Online Document], [cited 2008 April 29], Available HTTP: `http://www.ncbi.nlm.nih.gov/blast/fasta.shtml`

[16] Lorem Ipsum. “Lorem Ipsum,” [Online Document], [cited 2008 April 29], Available HTTP: `http://www.lipsum.com`

[17] Wikimedia Foundation. “Wikipedia,” [Online Document], [cited 2008 April 29], Available HTTP: `http://www.wikipedia.org/`

[18] Princeton University. “Rights, Rules, Responsibilites,” [Online Document], 2007, [cited 2008 April 30], Available HTTP: `http://www.princeton.edu/pr/pub/rrr/07/`

[19] Dartmouth College. “Sources: Their Use and Acknowledgement,” [Online Document], 1998, [cited 2008 April 30], Available HTTP: `http://www.dartmouth.edu/~sources/about/what.html#verbatim`

[20] D. Ford. “Ghetto Ass Bowl Pick’em,” [Online Document], 2007 December, [cited 2008 January], Available HTTP: `http://webscript.princeton.edu/~drford/gabp/`

[21] G. Snyder, et. al. “Spring Football,” [Online Document], 2008 March, Available HTTP: `http://springfootball.blogspot.com/`

Appendix A

Example Documents

A.1 Document A

Mauris pellentesque, odio sit amet accumsan laoreet, libero libero volutpat nulla, tempus rutrum lectus leo malesuada mi. Maecenas malesuada purus id tellus. Praesent nulla. Proin sed purus non leo fermentum iaculis. Nulla quis enim. Nullam elit nulla, pretium vel, rhoncus eget, sollicitudin sit amet, turpis. Cras sed lacus. Praesent scelerisque nibh eu nulla. Pellentesque sed massa. In hac habitasse platea dictumst. Proin ullamcorper tincidunt nisl. Sed posuere bibendum lacus. Morbi lacinia nisi et felis.

Nam sed erat sed augue volutpat commodo. Donec ut augue at lorem feugiat lobortis. Nulla dignissim pede quis sem. Suspendisse augue mauris, aliquet id, molestie quis, feugiat vitae, diam. Sed pretium elit in ligula. Vestibulum volutpat. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Ut consectetuer pretium lacus. Morbi at dui. Aliquam ut arcu sed ante pulvinar molestie. Nunc urna est, pulvinar consectetuer, malesuada eget, tempor ac, pede. Nunc malesuada, sapien eget fermentum scelerisque, lacus justo tempus orci, suscipit sodales neque enim nec erat. Nunc et ante at ligula hendrerit cursus. Fusce posuere. Morbi fermentum accumsan ipsum. Donec commodo tellus ut tellus. Sed interdum tellus id purus. Curabitur dignissim varius risus. Mauris vulputate tincidunt lorem. Nulla malesuada condimentum risus.

Vivamus sed nisi id nisl varius gravida. In turpis. Sed quis lectus nec lectus ultricies varius. Nulla non nunc. Morbi diam lectus, blandit nec, volutpat quis, gravida a, justo. Mauris in tortor. Suspendisse risus lectus, pulvinar ut, volutpat sed, tincidunt vel, felis. Praesent in neque eu tortor tempor mattis. In condimentum neque nec erat. Nam elementum mollis risus. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos.

Aenean at augue rhoncus tellus porttitor interdum. Cras nisi dui, pellentesque eget, posuere vitae, tincidunt eget, ipsum. Integer eget ipsum. Phasellus pellentesque libero quis nulla. Donec sed diam. Donec ligula nunc, vulputate non, dapibus eget, rutrum sit amet, dui. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Nam malesuada ipsum mollis lectus. Fusce blandit pretium sapien. Sed sollicitudin. Quisque ut mauris ac felis ultricies bibendum. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae;

Etiam convallis erat sit amet urna rutrum cursus. Vestibulum ac risus bibendum odio mollis cursus. Maecenas tellus urna, tincidunt sed, accumsan ut, faucibus dignissim, lectus. Phasellus mattis sapien ut dui. Praesent ullamcorper. Maecenas sed quam. Fusce interdum dolor a ante. Etiam pretium fringilla nisi. Suspendisse sit amet dui placerat tellus rutrum tincidunt. Mauris porta imperdiet sem. Cras imperdiet volutpat arcu. Duis non nibh. Cras convallis felis et mi. Pellentesque tempor convallis nulla. Suspendisse pharetra accumsan purus. Suspendisse consectetuer. Nullam a sem.

A.2 Document B

Aenean faucibus vestibulum odio. Vestibulum condimentum pretium pede. Aenean nibh mauris, ultrices a, venenatis in, tristique quis, purus. Vivamus luctus laoreet turpis. Phasellus leo. Nunc convallis. Morbi magna. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Ut rutrum, nisl sit amet eleifend placerat, neque massa vestibulum risus, sit amet porta odio dui eu pede. Morbi facilisis sapien et nisi. Duis mi urna, commodo eu, venenatis id, adipiscing eget, leo. Curabitur congue, nibh pulvinar hendrerit ornare, felis felis placerat velit, et malesuada enim odio molestie elit. Mauris eget leo. Praesent non risus sit amet ipsum iaculis venenatis. Donec tempus. Donec dignissim.

Nam scelerisque diam id nibh. Sed dui tellus, luctus ac, imperdiet vitae, molestie non, tortor. Integer sit amet elit. Nullam semper molestie lectus. Donec faucibus luctus ipsum. In cursus, quam eu ultrices dapibus, nisl lacus egestas sem, ut imperdiet urna nibh mattis ligula. Ut vulputate ullamcorper lectus. Phasellus cursus mi eget velit. Nulla cursus. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Nulla eget elit. Duis erat augue, lacinia quis, euismod eu, egestas id, nulla. Curabitur egestas imperdiet purus. Sed erat tortor, scelerisque in, rhoncus at, mollis sit amet, neque. Donec ligula. In sit amet risus quis erat eleifend volutpat. Praesent eget arcu sit amet mauris hendrerit tincidunt. Donec mattis est nec dui. Morbi feugiat sagittis ipsum. Aliquam pharetra metus id dolor.

Nunc pellentesque, mauris quis volutpat laoreet, lacus dui fringilla urna, sit amet fringilla sem orci quis felis. Morbi in pede. Sed non quam quis justo bibendum laoreet. Aenean eget risus. Vestibulum sem. Nunc imperdiet. Donec accumsan ultricies velit. Suspendisse potenti. Morbi pretium nulla sit amet nisi. Sed dolor. Vivamus eu nibh quis eros ultrices cursus.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Maecenas nisl. Maecenas interdum nisi at diam. Phasellus sit amet dolor. Morbi neque quam, hendrerit at, tempor quis, elementum eu, erat. Fusce sapien. Donec viverra, ipsum vitae venenatis tincidunt, nibh diam pellentesque arcu, nec rutrum massa metus vel tortor. Suspendisse ut quam. Phasellus nulla nisi, sagittis id, laoreet et, feugiat auctor, massa. Aenean ultricies sapien sed sapien. Ut metus. Quisque mi augue, cursus eu, rhoncus sit amet, iaculis quis, magna.

Fusce nec lorem nec sem tempus sollicitudin. Etiam ornare massa pulvinar quam. Aliquam consectetuer nunc at tellus. Nullam dapibus risus ut tortor. Vestibulum accumsan sollicitudin ipsum. Praesent a pede. Cras sit amet quam. Aliquam pharetra. Curabitur nibh arcu, facilisis ut, convallis eu, ultrices nec, velit. Nulla erat elit, pharetra at, mattis adipiscing, lacinia sit amet, diam. Suspendisse potenti.

Appendix B

Document Corpi

B.1 Corpus for Background Similarity Measure

The corpus of documents used to analyze baseline similarity levels in Chapter 3 was randomly selected from Wikipedia [17]. These documents are described below, to allow future researchers to construct the same corpus. All of these documents were accessed on 25 April, 2008, at approximately 6:30AM, GMT. If the document has been edited at some point thence, the Wikipedia revision for that date should yield the version used herein.

The Bronze Age Collapse

1,592 Words 10,062 Characters

`http://en.wikipedia.org/wiki/Bronze_age_collapse`

Eiger

1,798 Words 10,673 Characters

`http://en.wikipedia.org/wiki/Eiger`

Fantasy in the Sky

669 Words 4,140 Characters

`http://en.wikipedia.org/wiki/Fantasy_in_the_sky`

Freezing

695 Words	4,386 Characters

`http://en.wikipedia.org/wiki/Freezing`

General Order # 11, 1862

1,732 Words	10,481 Characters

`http://en.wikipedia.org/wiki/General_Order_%E2%84%96_11_%281862%29`

Katy Band

317 Words	1,827 Characters

`http://en.wikipedia.org/wiki/Katy_Band`

Mercator Cooper

480 Words	2,743 Characters

`http://en.wikipedia.org/wiki/Mercator_Cooper`

Ocean Sunfish

2,522 Words	15,437 Characters

`http://en.wikipedia.org/wiki/Ocean_sunfish`

Onager

438 Words	2,461 Characters

`http://en.wikipedia.org/wiki/Onager`

United States Presidential Election, 1888

921 Words	5,876 Characters

`http://en.wikipedia.org/wiki/United_States_presidential_election%2C_1888`