OPEN ACCESS Freely available online

Mathematica Eterna

ISSN: 1314-3344

Research Article

Exploring Mathematical Models and Algorithms for Plagiarism Detection in Text Documents: A Proof of Concept

Tanvir Ahmed*

Department of Civil Engineering, Bangladesh University of Engineering and Technology, Dhaka, Bangladesh

ABSTRACT

The aim of this article is to explore the literature and develop a proof of concept application for detecting plagiarism in text documents. Plagiarism has become a crucial research area in recent years as various research, publishing and teaching institutions seek to drastically eliminate the reinvention of the wheel associated with many research studies. People have wondered about how to check a suspected document against millions of reference documents to find an instance of plagiarism. This study focuses on providing the audience with the necessary information about mathematical models for texts, text processing, similarity measures, algorithms and data structures for detecting plagiarism in text documents based on the type of texts.

Keywords: Plagiarism; Mathematical model; Algorithms

INTRODUCTION

This chapter makes an introduction to plagiarism, detection of plagiarism of text-based documents following purpose, motivation and methodology of our research. At the end organization of this research book is described.

Introduction to plagiarism

Plagiarism is defined as "theft of intellectual property" [1]. It is an unethical practice that involves taking someone else's work or ideas without giving proper credit and passing it off as one's own work. It is like stealing intellectual property of others. Plagiarism is illegal in almost all jurisdictions across countries in the world and subject to punishment by law. The significant improvements of computer and communication technology in recent years have given rise to ease in text representation and fast transportation following abundant amount of resources in the World Wide Web. The wide availability of the internet has made plagiarism even easier and being independent of geographical boundary. As a result, plagiarism has become widespread in academia, work and it is necessary to fight plagiarism to protect author's rights. Luckily, the enormous processing power of computer has enabled software developers to implement software systems to assist detecting plagiarism.

Some examples of plagiarism

The following are examples of plagiarism [2]:

- Unmodified partial or entire copy of contents such as texts from one or multiple sources and passing them off one's own by not giving the original source any credit.
- Modifying contents through similar item swapping such as paraphrasing a text document and not referring to the original source.
- Failing to identify borrowed content from other author's work exactly.
- Representing a content in a different media without referring source.
- Introducing deliberate misinformation and ambiguity in reference citation

Plagiarism detection of texts

Plagiarism detection of texts involves gathering reference texts, preprocessing texts for useful representation to computers, representing texts into computationally efficient formats such as numerical weights from statistical or semantic analysis of words, computing similarity between two text documents and detection of plagiarism based upon the similarity. There are different methods available for numerical weights, similarity measurement metrics each one performs slightly different than the other and

Correspondence to: Tanvir Ahmed, Department of Civil Engineering, Bangladesh University of Engineering and Technology, Dhaka, Bangladesh; E-mail: tanvir.mbstucs@gmail.com

Received: 29-Feb-2024, Manuscript No. ME-24-29875; **Editor assigned:** 05-March-2024, PreQC No. ME-24-29875 (PQ); **Reviewed:** 19-Mar-2024, QC No. ME-24-29875; **Revised:** 21-Mar-2025, Manuscript No. ME-24-29875 (R); **Published:** 28-Mar-2025, DOI: 10.35248/1314-3344.25.15.243

Citation: Ahmed T. Exploring Mathematical Models and Algorithms for Plagiarism Detection in Text Documents: A Proof of Concept. Math Eter. 15:243.

Copyright: © 2025 Ahmed T. This is an open-access article distributed under the terms of the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

performs slightly better or worse depending upon the type of texts, size of texts and dataset of weighting method etc. It is very important to determine the suitable method for any plagiarism detection system.

Purpose of research

The study aims at detecting plagiarism on text documents. It focuses on providing application developers with the necessary information about mathematical models for texts, text processing, similarity measures, algorithms and data structures based on the requirements and specification of the system for which text based plagiarism detection application is to be deployed. The study also aims at developing a general purpose proof of concept application for plagiarism detection of text based documents.

Motivation

People have wondered about how to check a suspected document against millions of reference document to find an instance of plagiarism. Despite the huge processing power of computers nowadays, pattern matching is computationally expensive and is not suitable for plagiarism detection involving huge amount of texts. The idea of weighting words into numeric values such as term frequency-inverse document frequency, representing documents as vectors of weights and computing correlation among those vectors has changed the shape of information retrieval, document indexing and ranking, semantic analysis, information mining, text summarization and plagiarism detection, text translation etc. Plagiarism detection involves finding similarity of meaning among documents which is a very important task in the field of information retrieval.

Methodology

The method used to conduct this study consisted of a comprehensive search for relevant literature *via* six online database repositories: IEEE Xplore, ACM Digital Library, ScienceDirect and Springer, Web of Science, Google.

We studied various available techniques for textual information representation to computer systems, mathematical model for document representation, similarity measurement of text documents, text processing to design a general purpose plagiarism detection system. Then we define a model for a general purpose plagiarism detection system and describes various elements of the model and makes a comparison among different elements based on performance. The model incorporates vector space model for representation of texts and similarity measurement first described by Salton et al. Finally, a general purpose software system for is developed for plagiarism detection of texts.

Organization

Chapter 2 reviews previous research related to plagiarism detection and text similarity. In chapter 3, we describe the specifications and processes of our plagiarism detection system. The preprocessing of texts is described in chapter 4, while chapter 5 explains various weighting methods for words and chapter 6 discusses various similarity measures. A comparison among various weights and similarity metrics is described in chapter 7. Chapters 8 and 9 detail the implementation of our plagiarism detection application, developed in the Java programming language, and its performance. Finally, chapter 10 outlines future work and concludes the study.

MATERIALS AND METHODS

Research questions

To understand the efficiency and limitations of existing methods for plagiarism detection and to identify areas for further research in order to complement the performance of existing techniques. To achieve this aim three research questions were formulated:

- What are the advantages and limitations of existing plagiarism detection techniques?
- What factors necessitate further development of existing plagiarism detection techniques?
- What features are used to detect plagiarism?

Plagiarism has become a crucial research area in recent years as various research,
publishing and teaching institutions seek to drastically eliminate the re-invention of the wheel
associated with many research studies. Six electronic database resources were used to extract data for synchronization in this research: IEEE Xplore, ACM Digital Library, Science Direct, Web of Science, Springer and Google.

Bag of words models-based techniques: Some techniques detect exact copy plagiarism using bag of words models. Fingerprint-based techniques detect plagiarism by matching strings in documents based on common fingerprint proportions, which are sequences of characters contained in the entire document. Features based text similarity detection integrates the fingerprinting technique with four basic features: Keyword, first sentence similarity, query phase and least common subsequence.

Semantic plagiarism detection-based techniques: These are techniques that detect plagiarism by taking the meanings of words into consideration. Stop-word n-grams employ content terms to represent documents using a small list of stop-words to detect plagiarism based on the syntactic composition and similarities of texts in various documents. The check method examines the proportion of keywords' structural characteristics for texts in a document. It was developed based on an indexed structure used to parse documents to build the structural characteristics of text.

Graph-based representation: The category of detection technique converts text into a graph to detect plagiarism by considering the nodes and edges in the graph. The nodes house the sentences while the attributes of the sentences are represented by the edges. The output of an edge corresponds to the contents of a node, which is calculated using the Jaccard measure. Each node is linked with a unique identity formed by considering the concept of sentence terms. This enables the identification of all the synonyms between the original and suspicious documents using WordNet.

Tree-based representation: The IPLAG (Intelligent Plagiarism Reasoner in Scientific Publications) technique intelligently detects plagiarism by dividing the suspected documents into various structural components where the degree of similarity is assigned numerical weights and the weights are computed to Downloaded by Florida International University GL810 At 06:42 13 May 2015 (PT) determine the percentage of the similarity scores of the suspected document. The suspected documents are segmented into logical tree-structured representation using a procedure called DSEGMENT. In the multilayer self-organizing map based approach documents are modelled with a rich tree-structured representation, which hierarchically includes document features such as pages and paragraphs to detect plagiarism. An algorithm was developed to efficiently match similar texts for comparison.

Ontology-base representation: Techniques in this category are ontology based which means they cater for both text and idea plagiarism. These techniques employ ontology and WordNet to detect paraphrased sentences or words in documents. The offline ontology technique was used to detect plagiarized ideas but was limited in accuracy. Cross-language or bilingual plagiarism detection deals with the automatic identification of plagiarized texts from different languages. This is achieved by building a bilingual dictionary of terms or words in different languages and a translator capable of converting the queried language to the target language to detect plagiarism in a multilingual setting.

System specification and plagiarism detection process

System specification: Plagiarism detection in texts is closely associated with information retrieval, term analysis, filtering, weighting, similarity measurement. This study focuses on external plagiarism detection systems, having a collection of source documents and a set of suspected documents, the task is to find all plagiarized sections in both set of documents. There are various types of texts such as general texts, source codes, special syntactic languages etc. We specify our system for general text of English language. There is no limitation to the size of text but larger documents usually require more computational resources.

Plagiarism detection process

In general, our system follows the following steps:

Lexical analysis: The document is scanned from begin to end, lexemes are identified and converted to tokens. Tokens or terms are meaningful units of a document.

Stemming: Stemming involves converting derived words to their word stem or morphological root. As a result, related words are stemmed to same root.

Filtering: Depending on the context of the application, irrelevant words are removed.

Scoring or weighting: Words, phrases or sentences are mapped into numeric values to represent a document into a vector. There are various methods for scoring such a term frequency, inverse document frequency etc.

Vector representation: Documents are represented into vectors using the scoring of terms, number of terms representing the dimension of vector.

Similarity measure: Similarity between documents are computed from the vector representation using correlation computation of vector weights.

Evaluation and verdict: A threshold value of similarity is defined depending on the type of document and verdict is returned accordingly.

A block diagram of steps to be performed in the system is depicted below (Figure 1).

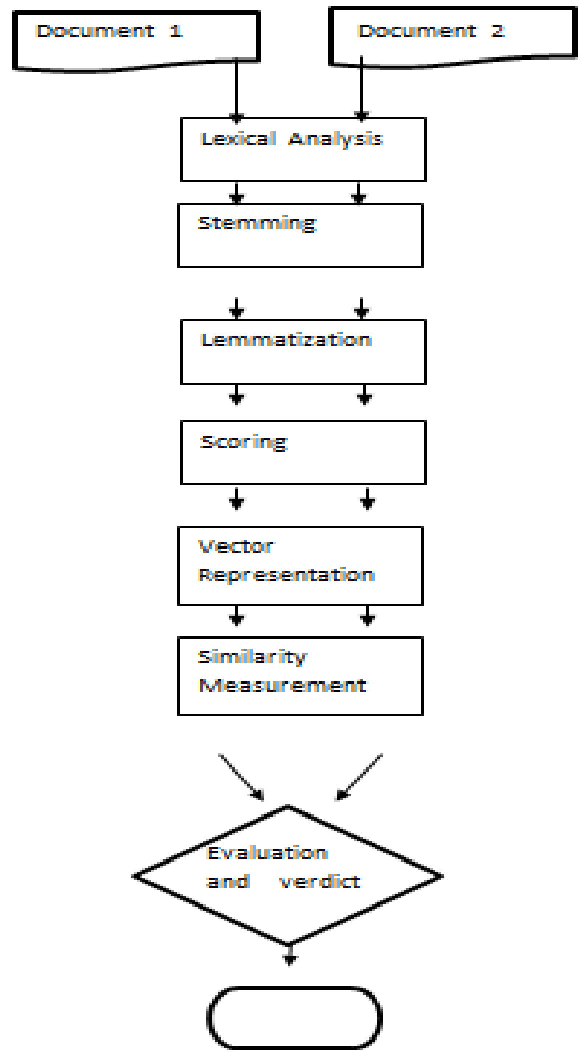

Figure 1: Plagiarism detection process.

Depending on the type of system filtering, stemming can be skipped. An improved version of stemming is lemmatization where part of speech of words are determined and stemming is performed depending on the part of speech. There are many alternative options are available for weighting and similarity measures and choosing the right measure is crucial for system's performance. We will take a deep look at them in later chapters.

Text preprocessing

Text preprocessing excludes useless contents from text and makes it suitable for efficient weighting and vector representation. Some preprocessing operations such as tokenization, filtering, lemmatization, stemming are described in this chapter.

Tokenization: Tokens from general texts could be extracted by looking at term boundary punctuation marks such as space, comma, full-stop, special character etc. The time complexity would be linear. Regular expression can be used for sophisticated patterns to be used as delimiter. The regular expression approach builds a finite state machine to match patterns. K. Thompson's regular expression algorithm [3] can build a Finite Automata for texts with characters of size n at asymptotic time complexity $O(n^2)$ and matching against a document length of m characters takes O(mn) if backtracking is not used.

N-gram fingerprinting: An n-gram is a sequence of n words from a text document. In this method, first proposed by Manber et al. [8] and subsequent work of Broader et al. [9] and Brin et al. [10] a portion of n-grams are hashed to integer numbers called minutiae by using a mathematical function for both the corpus and the query document and stored as part of an index of the respective document. Those numbers are called fingerprints and represents the content of documents. A portion of n-grams are taken into fingerprinting because of the time and space complexity of selecting n-grams and hashing. The number of overlapping minutiae between the query fingerprint and each document's fingerprint in corpus determines the similarity score of the corresponding document. The complexity of query is O(mn) where n is the number of documents and m is the number of minutiae, if number of minutiae is constant then the complexity is O(n). So, the performance of the query depends on choosing the number of minutiae. Because of using fingerprints, this method can detect similarity of texts if ordering of sentences are changed. This method is practical for large corpus if a small amount of n-grams are chosen for computing minutiae. Generating n-gram fingerprints are performed along with tokenization.

Stemming and filtering: In languages like English, there are many forms of a base word with almost similar meaning for grammatical reasons. Considering all derived forms of word will waste memory in vector space. Stemming solves this problem by converting derived forms of words into its base form or pseudo base form called stem. Some stemming algorithms such as Porter stemming do not use dictionary lookup and follows a predefined rules of stripping prefixes or suffixes. As a result a word stem may not be found in the dictionary in that approach. Filtering involves cutting off irrelevant tokens. Stemming and filtering involve similar process and typically performed together. Some popular stemming and filtering techniques are briefly described below.

Lookup table: In a lookup table, all possible derived forms of a word root is stored and upon query returns the root word for any of derived words. Some stemmers use production rules to generate probable forms of a word from a root. Lookup table is typically implemented by hash techniques.

Rule based stemming: Rule base stemming algorithms uses some predefined rules to convert word derivatives to their root. Rules are language specific and often involve prefix or suffix trimming. Rules are easy to implement and maintain. Rule based stemming requires knowledge about linguistics and morphology. Some examples of rule based stemming are Porter stemmer, Snowball stemmer etc.

Lemmatization: Lemmatization is an advanced approach to rule based stemming. Stemming usually refers to a crude heuristic process that chops off the ends of words in the hope of achieving this goal correctly most of the time and often includes the removal of derivational affixes. The idea behind lemmatization is the more information about a word would lead the rule based system to select a more appropriate root. Depending upon requirement tense, voice, part of speech of a word is first determined and the rule based stemming is applied based on part of speech. It is more efficient in preserving semantic structure of a document. Lemmatization is very correct in deriving base form of a word from derived forms because of vocabulary and morphological analysis of words. Lemmatization can be challenging for longer sentences and sentences with complex structures. Lemmatization is currently an open area of research.

Term weighting and vector space model

Scoring or weighting methods for terms: Scoring involves assigning each term a representative numeric value of document. Scoring often called weighting and weight of each term is used in vector representation of a document to perform various useful operations like clustering, similarity measure etc. Some scoring methods are:

Term frequency: Term frequency normalizes the occurrence of terms with the size of document. Term frequency is also called the probability of a term in a document. The term frequency of a term t in a document d is defined as the frequency of the term f(t), divided by total number of terms in the document, N:

$$\mathrm{TF}(t) = \frac{f(t)}{N} \tag{1}$$

Inverse document frequency: In term frequency, each term is considered equally important. As a result high frequency terms common to all documents like 'the', 'a', 'an', 'to', 'of' etc. would be sent to vector representation and may takeover more important terms. To eliminate the effects of common terms, inverse document frequency is used which is defined as binary logarithm of total number of documents, D divided by number of documents containing the term t, F_d (t).

$$\mathrm{IDF}\,(t) = \log_2 \frac{D}{F_d(t)} \qquad (2)$$

If a term t occurs in all the documents, IDF (t) computed would be 0.

Term frequency-inverse document frequency: Term Frequency-Inverse Document Frequency (TF-IDF) of a term t is defined as:

$$\mathrm{TF\text{-}IDF}\,(t) = \mathrm{TF}\,(t) * \mathrm{IDF}\,(t) \qquad (3)$$

TF-IDF gives high value for a term t of a document if that term occurs often in that particular document and very rarely anywhere else.

Kullbak-Liebler divergence: Kullbak-Liebler (KL) Divergence is the measure of difference between two probability distribution of terms. Let, A and B be two distinct probability distribution of terms in a document. The KL divergence of B from A over the term t is defined as.

$$D_{KL}\,(A \,||\, B) = \textstyle\sum_t A(t) * \log \frac{A(t)}{B(t)} \qquad (4)$$

Where A (t) and B (t) are probabilities of term t in A and B.

Topic signature: Lin et al. [12] introduced the concept of "topic signature" which is combining a sequence of correlated words into a word or phrase and described as:

Topic Signature={topic, signature}

$=\{topic, <(t_1, w_1), (t_2, w_2) \ldots. (t_n, w_n)>\}$ (5)

where topic is the key concept and signature is a vector of related term *i.e.* word t_i is correlated with topic having an associated weight w_i. Lin et al. [12] described a method to compute topic signature using likelihood ratio [13]. Topic signature is useful for measuring semantic similarity.

Program measures: Parker et al. described various building blocks of source codes as a measure. One approach for defining program metric they mentioned uses Halstead's theory, which computes volume and mental effort of a program by the following formula,

n_1= number of unique operators

n_2=number of unique operands

N_1=number of operator occurrences

N_2=number of operand occurrences

Volume of a program, $V=(N_1+N_2) \log_2 (n_1+n_2)$

Effort to write the program, $E=[n_1 \; N_2 \; (N_1+N_2) \; \log_2(n_1+n_2)]/(2n_2)$

Parker et al. described some more sophisticated metrics that involves tokenization, parsing and context free grammar which could be used as metric.

Vector space model

Vector space model is a mathematical model for representing text documents in a multidimensional vector space. Salton et al. [20] first introduced the concept where a document D1 represents a vector $V_n=\{t_1, t_2, t_3, \ldots. t_n\}$ of term scores. Each term t_i represents the algebraic score of term in the document. The following diagram depicts the vector space of three documents with at most three terms (Figure 2).

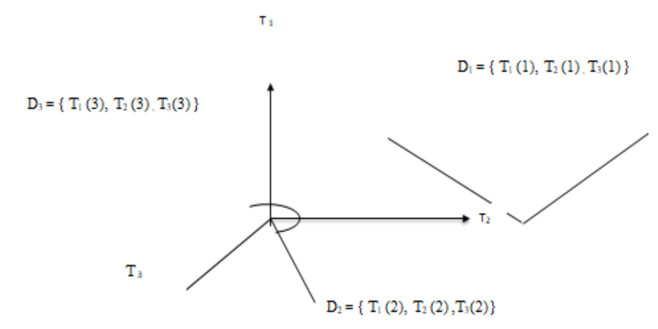
T1
D1 = { T1 (1), T2 (1), T3(1) }
D3 = { T1 (3), T2 (3), T3(3) }
T2
T3
D2 = { T1 (2), T2 (2), T3(2) }

Figure 2: Vector representation of document space.

Salton et al. proposed to use TF-IDF (Term Frequency-Inverse Document Frequency) as scoring of terms in a document.

The definition of term depends on the application. Typically terms are single words, keywords or longer phrases. If words are chosen to be the terms, the dimensionality of the vector is the number of words in the vocabulary.

Advantages

The vector space model has the following advantages over the Standard Boolean model:

- Simple model based on linear algebra.
- Term weights not binary.
- Allows computing a continuous degree of similarity between queries and documents.
- Allows ranking documents according to their possible relevance.
- Allows partial matching.

Limitations

- Long documents are poorly represented because they have poor similarity values (a small scalar product and a large dimensionality).
- Search keywords must precisely match document terms; word substrings might result in a "false positive match".
- Semantic sensitivity; documents with similar context but different term vocabulary won't be associated, resulting in a "false negative match".

Similarity measures in vector space model

Cosine similarity: The cosine of angle between two documents represented by vectors in vector space gives similarity value in range [0, 1], the greater the value, the more similar two documents are. This is called cosine similarity and a popular measure because of ease in computing cosine of angle. Given two vectors $\vec{P}$ and $\vec{Q}$ of dimension n, the cosine similarity is defines as:

$$\text{Cosine Similarity} = \frac{\vec{P}\cdot\vec{Q}}{\|\vec{P}\|\|\vec{Q}\|}$$

$$= \frac{\sum_{j=1}^{n} P_j \cdot Q_j}{\left(\sqrt{\sum_{j=1}^{n} P_j^2}\right)\left(\sqrt{\sum_{j=1}^{n} Q_j^2}\right)} \qquad (7)$$

Where {P_1, P_2, ... P_n} and {Q_1, Q_2, ... Q_n} are scores of terms in each document. Cosine similarity is independent of the length of vectors, so documents with different size having identical content will be treated identical.

Jaccard index and Dice's coefficient: Jaccard index or Jaccard coefficient measures similarity of two vectors by dividing the intersection of vectors by their union. Given two vectors →P and →Q of dimension n, the Jaccard coefficient is defined as:

$$Jaccard_{(\vec{P},\vec{Q})} = \frac{\vec{P}\cdot\vec{Q}}{|\vec{P}|^2 + |\vec{Q}|^2 - (\vec{P}\cdot\vec{Q})} \qquad (8)$$

Which gives a value in range [0, 1] like cosine measure. Jaccard distance which is computed by subtracting Jaccard coefficient from 1

$$JD = 1 - Jaccard_{(\vec{P},\vec{Q})} \cdot$$

is a measure of dissimilarity.

A similar measure Dice's coefficient is defined as:

$$Dice's\ Coefficient_{(\vec{P},\vec{Q})} = \frac{2|\vec{P}\cdot\vec{Q}|}{|\vec{P}|^2 + |\vec{Q}|^2} \qquad (9)$$

Pearson correlation coefficient: Given two vectors →P and →Q of dimension n, Pearson coefficient is defined as:

$$Pearson_{(\vec{P},\vec{Q})} = \frac{\sum_{j=1}^{n}(P_j - \bar{P})(Q_j - \bar{Q})}{\left(\sqrt{\sum_{j=1}^{n}(P_j-\bar{P})^2}\right)\left(\sqrt{\sum_{j=1}^{n}(Q_j-\bar{Q})^2}\right)} \qquad (10)$$

$$\text{Where } \bar{P} = \frac{1}{n}\sum_{j=1}^{n} P_j \text{ and } \bar{Q} = \frac{1}{n}\sum_{j=1}^{n} Q_j.$$

Unlike the previous two measures, similarity value of Pearson coefficient range is [-1, 1], where two identical vectors yield 1. The dissimilarity measure is,

$$PD = 1 - Pearson_{(\vec{P},\vec{Q})}$$

Averaged Kullbak-Liebler divergence: The Kullbak-Liebler divergence described in the previous section is not symmetric, so D_{KL} (A || B) ≠ DKL (B || A), as a result not suitable a similarity measure. Averaged KL divergence is symmetric and computed as:

$$D_{KL\_AVERAGE}(A \,||\, B) = C_1 * D_{KL}(A \,||\, E) + C_2 * D_{KL}(B \,||\, E) \qquad (11)$$

$$\text{Where, } E = C_1A + C_2B,\ C_1 = \frac{A}{A+B} \text{ and } C_2 = \frac{B}{A+B}.$$

Bhattacharyya coefficient: The Bhattacharyya coefficient of two vectors →P and →Q of dimension n is defined as:

$$BC_{(\vec{P},\vec{Q})} = \sum_{j=1}^{n}\sqrt{P_j Q_j} \qquad (11)$$

Clustering and cluster similarity: Clustering is the process of grouping identical terms in a document into a set of homogenous groups called clusters. Measuring the similarity of clusters between two documents can be used to detect certain plagiarism. One computationally suitable algorithm for clustering text documents is standard K-means algorithm. Given a data set S and a cluster length k, K-means algorithm works by initially selecting random k terms for k clusters, each being the centroid of the cluster and the remaining terms are assigned to nearest similar cluster using some distance measures. Then new centroid are computed for each cluster and all terms are reallocated to nearest cluster. This process stops when new centroid yield the same cluster as in previous step. The distance measures could be Euclidean distance, Jaccard distance, Pearson distance etc. Initial guess of k cluster could be done from scoring methods such as term frequency, TF-IDF, KL-divergence etc. Similarity of clusters could be computed from the methods of vector space model.

Other similarity measures

String matching approach: String matching is the simplest method that looks for text overlaps between two documents. One popular approach to matching string is building a finite automata. K. Thompson's regular expression algorithm [3] can build a Finite Automata for texts with characters of size n at asymptotic time complexity O(n^2) and matching against a document length of m characters takes O(mn) if backtracking is not used [14]. So, this approach is only practical for small text strings where m, n<10000.

One advantage of using automata is flexibility as it allows various operations such as: Restricting certain characters, matching subsequence, matching one or more times of a text pattern. The flexibility comes at the cost of computation expense. Linear string matching algorithms such as proposed by Knuth et al. [15], Boyer et al. [16], Karp et al. [17], Navarro et al. [18] can be used if no flexibility in matching is not required. The string matching methods detects only exact matches and a slight modification in sentence structure will make detection unsuccessful.

Suffix tree: Another approach is building a suffix tree introduced by Weiner [4], where a suffix tree of words is built for entire corpus and input document is queried for maximum subsequence of words. Another word matching algorithm is necessary for comparing word equality while building the suffix tree. Boyer and Moore's algorithm [6] takes O(pq) time at worst case for words with length p and q but performs much better on average which is near O(p+q). Farach et al. [5] described that suffix tree can be built in Θ (n) time where n is the total number of words in the corpus in our case. Gusfield and Dan at [7] showed that query in suffix tree is linear, so for a document with n words the time complexity of query in suffix tree is O(n). The overall time complexity of building the suffix tree of words would be O(np+nq). This method is computationally expensive for larger n.

Automatic summarization

An improvement of fingerprinting approach is summarizing the text based on important information and selecting n-grams from the summary. Automatic text summarization was initiated by Luhn et al. [11], He proposed word frequency to extract key concepts from a text ignoring most common high frequency words. There are two approaches to automatic text summarization.

Extractive summarization: Key concepts are extracted from text using non-semantic approaches such as topic models, word frequency models, probability distribution models etc.

Abstractive summarization: Semantic analysis and natural language processing methods are used to generate a concise summary of the text.

Hovy et al. developed SUMMARIST system for text summarization that follows three steps:

• Intermediate representation of given texts by identifying salient concepts and keywords.

• Interpretation of intermediate representation by scoring keywords or sentences into numeric weights for computation efficiency.

• Select a summary taking sentences with best scores from step 2.

A plagiarism detection system can follow the steps above and apply correlation measurement with association rule as having same keyword score in two document at step 2 to compute the similarity between two documents. Scoring of a document could be mapped into a multidimensional vector for efficient computation of correlation [19]. The any similarity measure of vector space model described can be applied.

RESULTS

Performance analysis of different term weights and similarity measures

Experiment on weighting and similarity measures: Thompson et al. made a comparison among different similarity measure methods using “PAN@Clef 2012 text alignment corpus, comprises of 6500 documents; of which 3000 are suspicious documents”. Following four tables shows their experiment results where similarity measure scores are computed for similar, reviewed (paraphrased, partially copied etc.) and distinct documents. Results from the experiment of Thompson et al. are presented in Tables 1 to 4.

Table 1: Recall for highly similar texts.

Similarity measures	**Term weighting**	**N-grams**	**Number of retrieved documents (highest ranking documents**	
			1	**5**
Cosine similarity	Binary/TF/TFIDF	10	0.97	1
KLD	Binary/TFIDF/TF	12	0.97	1
Dice coefficient	Binary/TF	12	0.96	1
Jack -index	Binary/TF	12	0.96	1
Bhayttacharyan	Binary/TF/TFIDF	12	0.95	1
PCC(R)	TF/Binary	10	0.94	1
JSD	Binary/TF/TFIDF	10	0.83	0.897
Euclidean distance	TF/Binary	8	0.68	0.73

Table 2: Recall for lightly reviewed similar texts.

Similarity measures	**Term weighting**	**N-grams**	**Number of retrieved documents (highest ranking documents)**

Table 2: Recall for lightly reviewed similar texts.

Similarity measures	Term weighting	N-grams	Number of retrieved documents (highest ranking documents)						
			1	5	10	20	30	40	50
Bhayttacharyan	TF	3	0.913	0.933	0.94	0.947	0.96	0.96	0.96
Cosine similarity	TFIDF	3	0.893	0.933	0.94	0.94	0.953	0.953	0.96
Dice coefficient	Binary	3	0.893	0.927	0.933	0.94	0.947	0.947	0.947
Jaccard index	Binary	3	0.893	0.927	0.933	0.94	947	0.947	0.947
KLD	TF	3	0.853	0.873	0.913	0.933	0.933	0.947	0.947
PCC (R)	TFIDF	1	0.66	0.727	0.807	0.827	0.86	0.873	0.9
JSD	TF	3	0.56	0.633	0.667	0.69	0.697	0.72	0.74
Euclidean distance	TF/IDF	3	0.51	0.613	0.62	0.627	0.633	0.633	0.63

Table 3: Recall for heavily reviewed texts.

Similarity measures	Term weighting	N-grams	Number of retrieved documents (highest ranking documents)						
			1	5	10	20	30	40	50
Cosine similarity	TFIDF	3	0.527	0.6	0.653	0.66	0.687	0.707	0.733
PCC (R)	TFIDF	1	0.5	0.567	0.65	0.72	0.753	0.78	0.827
Dice coefficient	Binary	2	0.513	0.6	0.647	0.693	0.713	0.727	0.753
Jaccard_index	Binary	2	0.513	0.6	0.647	0.693	0.713	0.727	0.753
Bhayttacharyan	TF	2	0.5	0.567	0.613	0.633	0.66	0.713	0.747
KLD	TF	3	0.48	0.513	0.607	0.627	0.66	0.673	0.72
JSD	TF	3	0.38	0.447	0.487	0.507	0.52	0.54	0.573
Euclidean distance	TFIDF	1	0.313	0.373	0.447	0.493	0.527	0.533	0.55

Table 4: Recall for highly dissimilar texts.

Similarity measures	Term weighting	N-grams	Number of retrieved documents (highest ranking documents)								
			1	5	10	20	30	40	50	60	70
PCC (R)	TFIDF	1	0.367	0.433	0.50	0.58	0.62	0.66	0.70	0.733	0.787

Cosine similarity	TFIDF	1	0.313	0.40	0.46	0.513	0.58	0.64	0.66	0.673	0.733
Dice coefficient	Binary	2	0.273	0.34	0.46	0.54	0.573	0.61	0.653	0.68	0.707
Jaccard index	Binary	2	0.273	0.34	0.46	0.54	0.573	0.61	0.653	0.68	0.707
Bhayttacharyan	TF	2	0.333	0.301	0.467	0.513	0.547	0.567	0.613	0.667	0.667
KLD	TF	2	0.287	0.347	0.373	0.373	0.407	0.46	0.493	0.527	0.58
JSD	TF	2	0.27	0.32	0.34	0.367	0.38	0.40	0.44	0.487	0.54
Euclidean distance	TF	1	0.247	0.267	0.293	0.313	0.333	0.353	0.387	0.407	0.44

Findings

From the experiment of Thompson et al., we observed that KL-divergence and Bhayttacharyan measures shows the best results in case of both similar and dissimilar documents and TF-IDF weighting performs better than tf or bianry. On the other hand, Cosine, Pearson and other measures yield slightly diverse when it comes to large dissimilar documents. Cosine measure yields moderate results for all cases and suitable for general purpose systems. Jaccard index and Dice's coefficient are easy to compute and yield similar results like cosine, so we recommend them for computer systems with limited capacity.

We propose various techniques for preprocessing, scoring, vector representation, similarity measures depending upon the specifications for computer assisted plagiarism detection system based on our discussion and the experiment of Thompson et al. We assume the system can perform one hundred operations per second and define some parameters:

Document size: Documents having more than ten thousand words are considered big.

Exact clone: A document having some exact identical sections from other documents.

Partial copy: A document with mixture of exact and similar identical section from other documents.

Paraphrasing: A document with paraphrased sections have semantically closely related sections from other documents.

Ordering: A suspicious document keeps ordering from a source if it keeps syntactic or semantic sequence of words or sentences for a section.

The following table describes our proposals for different parameters (Table 5).

Table 5: Proposals for different parameters.

Document size		Document attribute approximation				Text category		Proposal						
Small	**Big**	**Exact clone**	**Partial copy**	**Paraph rasing**	**Ordering**	**General text**	**Source code**	**Tokenization**	**Stemming**	**Scoring**	**Vector representation**	**Substring matching**	**Suffix tree**	**Similarity measure**
√		High				√				∂		√		Matched word frequency
	√	High				√		√		TF-IDF, Boolean	√			Cosine, Pearson,

														Jaccard, Dice coefficient etc.
√	√		High		Distinct			√	Rule based	TF-IDF, TF				Averaged KL divergence, Bhayttacharyan
√	√			High	Distinct	√		√	Lemmatization	Topic Signature, KL divergence, TF-IDF				Averaged KL divergence, Bhayttacharyan
√	√			High	Uniform			√	Cluster	Topic Signature, KL divergence, TF-IDF	√			Averaged KL divergence, Cluster similarity
	√		High	Uniform				√	N-gram	TF-IDF	√			Cosine, Pearson
√			High					√	N-gram				√	N-gram matching frequency
√		High	High	Uniform			√	√				√		Matched keyword frequency
	√		High	Distinct			√	√	Parsing	Program measure	√			Averaged KL divergence, Cosine, Pearson etc.

Analysis of different metric and techniques show that some perform better than others depending on the type of document and resource of the system. One of the key challenge in plagiarism detection is lemmatization which is to derive appropriate meaning of terms from texts, a feat that statistical models and natural language processing do not perform as good as humans. Lemmatization is an open area of research and future improvements in this area would lead computer assisted plagiarism detection to new dimension of success. Nevertheless, current computer assisted plagiarism detection techniques could considerably help to find plagiarized work.

System design of plagiarism detection system

System design: Our plagiarism detection system is designed as an external plagiarism detection system. In external plagiarism detection, given a set of suspicious documents and a set of source documents, the function of a typical plagiarism detection system is to find all text passages in the suspicious documents which have been plagiarized and the corresponding text passages in the source documents. Our system is designed to calculate plagiarism between a suspected document and a reference document as percentage where one hundred percent mean plagiarism to the highest extent and zero percent mean no plagiarism at all. The system is designed to be implemented as a desktop application.

Use case: The user can add any number of reference text documents to check against a suspected document. There is only one option for suspected document, the user can either pick a suspected text document or put some text on the textbox. Upon query, the application will compute plagiarism in percentage and show the result to user for each combination of a reference document and the suspected text. The following is a diagram (Figure 3).

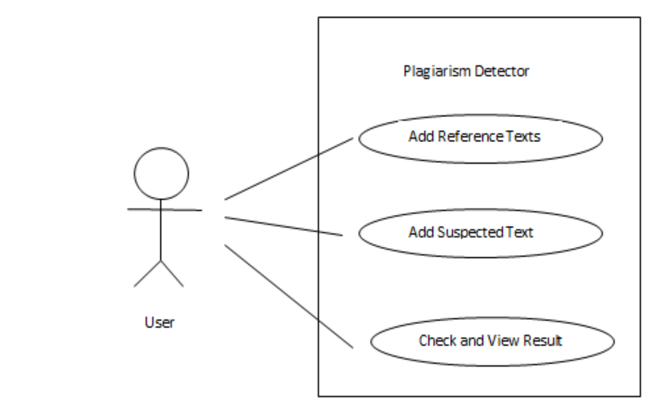

Figure 3: Plagiarism detector shows the results to user.

Future work

This study on focused on only general texts of English. Methods of this study can be used to upgrade from general texts to special texts such as source code using suitable tokenization and weighting methods. We discussed program measure weighting in chapter 5, which could be useful for weighting terms in source code. Our system could be applicable to all languages having proper tokenization and lemmatization specific to a particular language. Our system can be upgraded to check paraphrased plagiarism through semantic analysis. As natural languages are ambiguous and may have complex semantic structure because of synonymy and polysemy, machine learning approaches for semantic analysis such as deep neural networks could be used to score terms with same meaning into a uniform weight.

In future, we will focus to include options for Bangla language texts using proper processing of Bangla words. We can make collaboration with online research journals such as Springer to detect plagiarized papers using our application.

CONCLUSION

The increasing growth of the internet has made a huge amount of information available. It is difficult for humans to detect plagiarism from large amounts of text. Thus, there is an immense need for automatic plagiarism detection tools in this age of information overload. In this paper, we emphasized vector space model based approaches for single and multi-document plagiarism checking. We described some of the most extensively used methods such as text preprocessing and representation approaches, frequency-driven methods, similarity metrics etc. Although it is not feasible to explain all diverse algorithms and approaches comprehensively in this paper, we think it provides a good insight into recent trends and progresses in plagiarism detection methods and describes the state-of-the-art in this research area.

REFERENCES

1. Maurer HA, Kappe F, Zaka B. Plagiarism-A survey. J Univers Comput Sc. 2006;12(8):1050-1084.
2. Lancaster T, Culwin F. Classifications of plagiarism detection engines. Innov Teach Learn Inform Comput Sci. 2005;4(2):1-6.
3. Thompson K. Programming techniques: Regular expression search algorithm. Communications of the ACM. 1968;11(6):419-422.
4. Weiner P. Linear pattern matching algorithms. Annu Symp Switch Auto Theo. 1973;1-11.
5. Farach M. Optimal suffix tree construction with large alphabets. Annu Symp Found Comput Sci. 1997;137.
6. Boyer RS, Moore JS. A fast string searching algorithm. Commun ACM. 1977;20(10):762-772.
7. Gusfield D. Algorithms on stings, trees, and sequences: Computer science and computational biology. ACM SIGACT News. 1997;28(4):41-60.
8. Manber U. Finding similar files in a large file system. InProceedings of the USENIX Winter 1994 Technical Conference on USENIX Winter 1994 Technical Conference 1994;2-2.
9. Broder AZ, Glassman SC, Manasse MS, Zweig G. Syntactic clustering of the web. Comput Net ISDN Syst. 1997;29(8-13): 1157-1166.
10. Brin S, Davis J, Garcia-Molina H. Copy detection mechanisms for digital documents. SIGMOD Rec.1995;24(2):398-409.
11. Luhn HP. The automatic creation of literature abstracts. IBM J Res Dev. 1958;2(2):159-165.
12. Lin CY, Hovy E. The automated acquisition of topic signatures for text summarization. InProceedings of the 18th conference on Computational linguistics-Volume 1, 2000;495-501.
13. Dunning T. Accurate methods for the statistics of surprise and coincidence. Comput Linguist. 1994;19(1):61-74.
14. HOVY E. Automated Text Summarization in SUMMARIST. Proceedings of the Intelligent Scalable Text Summarization. 1997:18-24.
15. Knuth DE, Morris JR JH, Pratt VR. Fast pattern matching in strings. Computer Algorithms: String Pattern Matching Strategies. 1994;55:8.
16. Karp RM, Rabin MO. Efficient randomized pattern-matching algorithms. IBM J Res Dev. 1987;31(2):249-260.
17. Navarro G, Raffinot M. A bit-parallel approach to suffix automata: Fast extended string matching. Lect Notes Comput Sci. 1998:14-33.
18. Huang A. Similarity measures for text document clustering. InProceedings of the sixth New Zealand computer science research student conference (NZCSRSC2008), Christchurch, New Zealand 2008;4:9-56.
19. Salton G, Wong A, Yang CS. A vector space model for automatic indexing. Commun ACM. 1975;18(11):613-620.
20. Cutting DR, Karger DR, Pedersen JO, Tukey JW. Scatter/Gather: A cluster-based approach to browsing large document collections. InProceedings of the 15th annual international ACM SIGIR conference on Research and development in information retrieval 1992;318-329.