Original Article | Peer Reviewed | Open Access

International Journal of Experimental Research and Review (IJERR)
© Copyright by International Academic Publishing House (IAPH)
ISSN: 2455-4855 (Online)
www.iaph.in

Enhancing Academic Integrity: An Analysis of Advanced Techniques for Plagiarism Detection using LESK, Word Sense Disambiguation, and SVM

Devesh Kumar Upadhyay[1]* and Keshav Sinha[2]

[1]Department of Computer Science and Engineering, Birla Institute of Technology, Mesra, Ranchi, India;
[2]School of Computer Science, UPES, Dehradun, India

E-mail/Orcid Id:

DV, deveshupadhyay3@gmail.com, https://orcid.org/0000-0002-2399-1850;
KS, Keshav.sinha@yandex.com, https://orcid.org/0000-0003-1053-3911

Article History:

Received: 07th Oct., 2023
Accepted: 19th May, 2024
Published: 30th May, 2024

Keywords:
Word Sense Disambiguation (WSD), LESK, Semantic Analysis, Support Vector Machine (SVM), Universal Language Translator (ULT)

How to cite this Article:
Devesh Kumar Upadhyay and Keshav Sinha (2024). Enhancing Academic Integrity: An Analysis of Advanced Techniques for Plagiarism Detection using LESK, Word Sense Disambiguation, and SVM. *International Journal of Experimental Research and Review*, *39*(spl.) 92-108.
DOI: https://doi.org/10.52756/ijerr.2024.v39spl.007

Abstract: Plagiarism is widespread in academia, from ancient literature to modern research, where scholars' work is copied and published without authorization. In the late 90s, researchers explored various methods to detect plagiarism, including Word Sense Disambiguation (WSD), LESK, and Support Vector Machine (SVM). However, these conventional techniques have shown limitations in aligning with contemporary writing styles. This paper proposes an improved LESK algorithm for word sense detection and Improved SVM for feature extraction, addressing the shortcomings of existing methods and offering enhanced accuracy and efficiency in identifying plagiarized content. The study evaluates the proposed system using three datasets from PAN 2012, PAN 2013, and PAN 2014 documents to assess its performance across different types of text plagiarism. Results demonstrate the system's superiority, achieving higher classification accuracy when trained on the Second Dataset. A comprehensive analysis of the feature's significance in the training database reveals the importance of discriminative sentence similarity. The proposed system contributes to combating academic dishonesty, ensuring the authenticity of digital content in various contexts. Future work will explore cross-lingual plagiarism detection and image duplicity identification using Word Sense Disambiguation techniques. Additionally, efforts will be made to optimize time complexity for faster execution.

Introduction

From the earliest days, plagiarism has been the most concerning writing. Plagiarism means copying someone else's work without informing the author and using it for personal purposes. The origin of the word "Plagiarism" is from Latin, which means "Kidnapper" or "to kidnap". Plagiarism detection plays an essential role in academics because students, research scholars and teachers are intended to produce original research work (Maurya and Madhusudhan, 2023). Plagiarism can be prevented by using the proper citation to ensure the original author gets credit for their original content (Prasanth and Rajshree, 2014; En et al., 2023). Detecting plagiarism will open many doors in research articles and thesis writing (Sedaghat, 2024). A mathematical document consists of formulas and results that do not need standard literature to understand those papers. A plagiarism detection tool is used to maintain the standard of the paper (Nguyen, 2023). The fast-growing internet provides lots of information in little time, and that information is used while writing papers. There is now a consensus regarding plagiarism, and we can say that plagiarism has become one of the biggest challenges in day-to-day life, and it also discourages the academic community from writing their papers.

Type of Plagiarism

Several techniques are available to detect plagiarized content. Plagiarism is categorized into (i) Intentional and (ii) Unintentional plagiarism. In Intentional plagiarism, the writer has intentionally copied the content from various sources (Banerjee & Pedersen (2002)). The copied part from dissimilar sources is detected during the plagiarism check. Intentional plagiarism is further divided into four parts:

Idea Plagiarism:

In this, any individual copyrights the original author's idea without crediting the original author. It is challenging to identify, and only authorized individuals can detect plagiarism.

Para Phrasing:

The copyrighted text is re-ordered or re-arranged, but the text's meaning remains the same.

Direct Plagiarism:

In this case, the individuals reuse the few words or text that the original user and plagiarism tools can quickly identify.

Patchwork Plagiarism:

It lifts some patches of content, and without crediting the original user, that portion of the original work is copyrighted by another user.

The writer does not intentionally plagiarize the content in unintentional plagiarism, but it may happen automatically due to a lack of vocabulary (Gipp et al., 2014). Unintentional plagiarism can be sub-divided into three types:

a. Citation Plagiarism:

A particular portion of the document is copied and claimed for another.

b. Insufficient acknowledgment of plagiarism:

The content is copied from internet and paper sources, which is not appropriately cited in the paper, causing plagiarism.

c. Mosaic Plagiarism:

By mistake or ignorance, a person left their content on the internet or any repository that may be plagiarized after a particular time. The original author may not know about the copied portion of his/her original work.

Plagiarism Detection Methods

Plagiarism detection consists of both paid and accessible formats. Tools like "PlagTracker" and "Turnitin" take the original text as input and apply some algorithms to extract the string. The extracted text string is then compared with the existing string database across the Network. The database used by this tool is dedicated, with strings used to match the original text (Slimani, 2013). If the content is plagiarized, it is identified from the database and marked throughout the document. The plagiarism detection algorithm works in the following three steps:

Knowledge-based Method:

It is a machine-readable repository where standard datasets match the content with the original text (Abdi et al., 2017). For comparison, there are different techniques used under this method, such as:

a. LESK Algorithm: It works on a data dictionary repository and checks the word's definition in a sentence. Similar words are collected in the dictionary according to a maximum number of matches.

b. Semantic Similarity: It finds the standard distance between similar words with the same meaning and sense.

c. Selection Preferences: It counts the number of pairs of words with identical meanings. The selection is made using the original document and paired Word document. The detected pair of words are separated to form a plagiarized dictionary used for plagiarism detection.

d. Heuristic Method: It evaluates many different linguistic properties to find the sense.

Supervised Method:

This machine learning technique applies the Word-Sense Disambiguation (WSD) system (Manning et al., 2008). In this method, the tags are created from the dictionary. There are several ways for supervised machine learning, such as:

a. Decision List: In this, the list is created using the "if and else" condition and used to calculate the score. The score is then used to generate the final decision tree, and the maximum score will indicate a high chance of matching sense between the content.

b. Decision Tree: It uses the classification rule to divide the data set into two parts. The test result and output are stored in the correct node, while the possible sense of the word is stored in the left node.

c. Naïve Bayes: It is a probabilistic classifier that compares the condition for finding the sentence feature. The mathematical model for a training set is given as:

$$S = \underset{S_i \in Sense_D(w)}{argmin} \; P(S_i|f_1, \dots, f_m) \quad (1)$$

$$S = \underset{S_i \in Sense_D(w)}{argmin} \; \frac{P(f_1, \dots, f_m|S_i P(S_i)}{P(f_1, \dots, f_m)} \quad (2)$$

$$S = \underset{S_i \in Sense_D(w)}{argmin} \; \mathrm{P}\,(S_i) \prod_{j=1}^{m} P(f_i|S_i) \quad (3)$$

Where, S= Sense, w = words, f = features, m = number of features, P(s) = probability of frequency in training set of sense, $P(f_i|S_i)$ = calculated feature present in the sense (Upadhyay et al., 2021).

Instance-Based Learning:

It is a memory-based learning algorithm used to compare new problem instances with training instances. A similar instance is stored in memory using the k-NN (k-nearest neighbor) algorithm to find the similarity between two words. Once a similarity is found, the Hamming distance is calculated using k-NN. The resultant familiarity between input and stored data is stored in the repository.

Unsupervised Method:

It depends on readable dictionaries or a sense-annotated data set. It does not assign meaning to the word. Instead, it divides the words based on the information. It consists of different techniques such as:

a. **Context Clustering:** The context vectors with the same meaning are grouped to form clusters. The word space, vector space, and dimension parameters are clustering parameters. The word within the context is treated as a vector, and the similarity is calculated using the co-occurrence matrix. Google uses the n-gram (i.e., n=5), which is considered a compressed summary. There are 9.7 billion sentences that 5-grams extract. All of them are tagged with POS (part-of-speech), and the resulting clustering is used for WSD utilization (Mahdavi et al., 2014).

b. **Word Clustering:** Here, similar and identical contexts are clustered. A list of the words is taken, and then the similarity is found among them, following which an ordered similarity tree is created for matched words. The common word is treated as an initial node, and the sense of the word is placed in the sub-node of the tree.

c. **Co-occurrence Graph:** A graph-based detection method wherein a word is plotted in vertices 'X,' and the corresponding ID of the word is plotted on edges 'E'. The distance between two words is calculated using the Markov clustering method, where every edge has a weight, which is the co-occurring frequency of the words. Weight for the edge {m, n} is given by the formula,

$$W_{mn} = 1\text{- max } \{P\,(W_m|W_n\,),\ P\,(W_n\,|W_m)\} \qquad (4)$$

Here $P\,(W_m|W_n\,)$ = is the $freq_{mn}/freq_n$, $freq_{mn}$ is the co-occurrence frequency of words W_m and W_n, and $freq_n$ is the occurrence frequency of W_n.

Different researchers use different methods and techniques for plagiarism detection. Some conventional methods, such as the Spanning tree-based approach, are used to identify the set of senses. Joshi et al. (2013) introduced the Graph Dependence (PGD) method for graph dependence analysis. PGD works on large data documents and files. It works on manual detection, taking more time while comparing the data with stored data. Pal et al. (2013) implemented the WSD technique, which works on Indian languages, mainly for Bengali. Due to automatic detection, it works for an extensive data set with less time complexity. Alzahrani et al. (2012) implemented a monolingual language for plagiarism detection, which works on intrinsic, extrinsic, and cross-lingual plagiarism detection. It works on copying text but fails to detect intelligent plagiarism. It is speedy and precise for small document files but will not work for large documents. Hiremath et al. (2014) implemented the day's plagiarism for text-based and shape-based plagiarism detection. It is reliable only for text-based searches. Basile et al. (2014) implemented the LESK and Word Sense Disambiguation (WSD) technique to find the overlap between words with absolute meanings. Mozgovoy (2011) implemented a natural language processing technique that works on a similar function for tree matching due to its manual detection taking less time while processing. Agarwal et al. (2013) implemented a semantic similarity for a group of words (text file) to find a proper relation. This method is very fast and automatically detects plagiarism. Mentari et al. (2022) used a Winnowing-based system to detect cross-language plagiarism, achieving 84.7% accuracy. While promising, challenges like computational complexity and tool reliance persist. Nonetheless, it represents a crucial step in preserving academic integrity across diverse linguistic landscapes, demanding further refinement for sustained efficacy. Kumari and Kumar (2023) introduce an extended Lesk and Conceptual Density approach for Word Sense Disambiguation (WSD), which is crucial in natural language processing. By leveraging overlap density and evaluation through BLUE, promising results are offered, particularly for morphologically rich languages like Hindi. Ayetiran and Agbele (2016) propose an optimized variant of Lesk-based algorithms for Word Sense Disambiguation (WSD), addressing computational complexity through topic composition. Leveraging English WordNet enriched with Wikipedia and Semcor corpus, the algorithm demonstrates superior efficiency and effectiveness across general and domain-specific datasets, particularly in knowledge-based techniques. El-Rashidy et al. (2023) propose an advanced plagiarism detection system that uses SVM and Chi-square techniques to leverage selective sentence similarity features and hyperplane equations. With three key phases, including document pre-processing and hyperplane computation, it outperforms recent systems, achieving top scores on PAN 2013 and PAN 2014 datasets. Kumar et al. (2020) introduce an Adapted Lesk algorithm-based Word Sense Disambiguation (WSD) system, employing a knowledge-based approach with WordNet. The system consists of three units: Input query, Pre-Processing, and WSD classifier. By leveraging context information and the lexical database, the WSD classifier accurately identifies the sense of polysemous words. Vrbanec and Meštrović (2020) overview paraphrase detection techniques, focusing on corpus-based models, particularly deep learning (DL) models. Evaluating eight models on three public datasets, including LSI, TF-IDF, Word2Vec, Doc2Vec, GloVe, FastText, ELMO, and USE, it explores text pre-processing, hyperparameters, sub-model selection, and similarity thresholds. Results indicate DL models' competitiveness with traditional approaches, suggesting further development potential. Plagiarism detection software, often labeled as prevention tools, faces limitations in capturing the breadth of plagiarism, including translations and intent. Despite being marketed as such, these tools primarily function as text-matching tools among over 25 available options. Concerns like false positives and negatives persist, while recent legal rulings reinforce universities' authority to revoke doctorates based on their criteria (Altheneyan & Menai, 2020). Plagiarism detection software, often labeled as prevention tools, faces limitations in capturing the breadth of plagiarism, including translations and intent. Despite being marketed as such, these tools primarily function as text-matching tools among over 25 available options. Concerns like false positives and negatives persist, while recent legal rulings reinforce Universities' authority to revoke doctorates based on their criteria (Weber-Wulff, 2018). The motivation of this work is to detect plagiarism in the least amount of time with higher accuracy. Previous techniques have found similarities between the two documents. It provides high accuracy in text-matching, whereas it fails when it detects image plagiarism. This paper uses semantic similarity to find the distance and relationship between the content. This paper is organized into four significant fragments. Section 1 will discuss the introduction, followed by the description of the research methodology in Section 2. In section 3, we discuss the result. Finally, in section 4, the conclusion of the paper is presented.

Research Methodology

After an exhaustive study of research articles, we proposed a modified LESK Algorithm using semantic similarity for plagiarism detection. For the completeness of this paper, we have used the simplified LESK Algorithm with brilliant default word sense (Vasilescu et al., 2004). The algorithm is used to find the sense between two words.

```
function SIMPLIFIED LESK(word, sentence) returns best sense of word
best-sense <- most frequent sense for word
max-overlap <- 0
context <- set of words in the sentence
for each sense in the senses of word do
signature <- set of words in the gloss and examples of sense
overlap <- COMPUTEOVERLAP (signature, context)
if overlap > max-overlap then
max-overlap <- overlap
best-sense <- sense
end return (best-sense)
```

In semantic similarity, it checks the distance and relationship between the content. In semantic similarity, we have used WSD (Word Sense Disambiguation), which checks the sense of the word. The proposed framework for detection is represented in *table 1*. The step-by-step procedure for the proposed plagiarism detection strategy is discussed as follows:

Step 1:

For the initial simulation, the paper dictionary is set. The two types of dictionary location are used: (i) Local and (ii) Global. The college database (DB) creates the local dictionary, and the universal web repository is shared globally.

Step 2:

For our proposed framework, the data is collected from the World Wide Web (WWW) and is placed in the local DB.

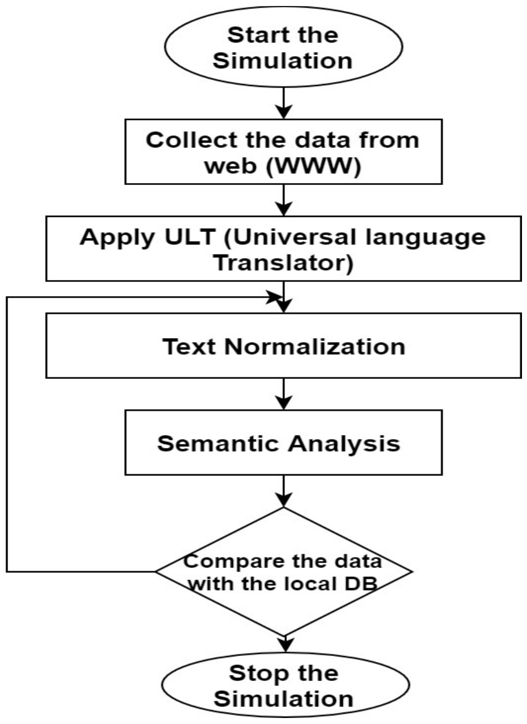
Start the Simulation
Collect the data from web (WWW)
Apply ULT (Universal language Translator)
Text Normalization
Semantic Analysis
Compare the data with the local DB
Stop the Simulation

Figure 1. Framework for Proposed Plagiarism Detection.

Step 3:

The Universal Language Translator (ULT) is applied, where it is possible to change the exact word by changing the language and publishing it in national journals, articles, and magazines. Using this, it is changed to a unique language (English) before checking for plagiarism.

Sample 1: French Sentence
Dans les siècles qui ont aucune trace ces îles étaient le foyer de millions d'oiseaux heureux, la station balnéaire de cent fois plus de millions de poissons, des lions de mer, et d'autres créatures dont les noms ne sont pas si communs ; la résidence marine, en fait, d'innombrables créatures prédestinées à partir de la création du monde à mettre en place un magasin de richesse pour l'agriculteur britannique, et un.
ULT Result Conversion: English Language
In ages without record, these islands were the home of millions of happy birds, the resort of a hundred times more millions of fishes, sea lions, and other creatures whose names are not so common.

It can be observed that the number of characters and words is different. For this reason, we use a universal language translator in our work.

Step 4:

The text normalization has been applied to the result of a ULT. Text normalization converts documented text into plain text by removing all the non-alphanumeric characters, regular expressions, word suffixes, and whitespace characters into a single space. The canonicalization process converts the ULT text data into standard, shared, or canonical form for delegated text normalization.

Sample 2: Before Normalization
Last Thursday, G. Gordon Liddy had the so-called confidential witness live on his radio show. CW, who discovered foster's body in Fort Marcy Park, Va., just across the Potomac River from Washington, at 5:45 p.m. on July 20, 1993, said several times with emphasis that he told the FBI that foster's hands were palms up, thumbs out and there was no gun in either hand.
After Normalized:
Last Thursday, G Gordon Liddy had the so-called confidential witness live on his radio show. CW, who discovered foster's body in Fort Marcy Park, Va, just across the Potomac River from Washington, at five forty-five pm on July twentieth, nineteen ninety-three, said several times with emphasis that he told the FBI that foster's hands were palms up, thumbs out and there was no gun in either hand.

If we observe, before normalization, the text had unwanted characters. Normalization removed all the unwanted characters and removed the numbers.

Step 5:

Semantic Analysis is challenging for natural language processing. It is used to analyze the relationship between two sets of documents. The text must check the domain space and determine whether two words are similar. While using the WSD to find the sense in the plain text and the improved LESK algorithm to get an accurate result in less time, we already know word sense disambiguate in the introduction, types of WSD, and its applications. In this semantic analysis, we divided the document into three phases to get a better result: (i) heading, (ii) body, and (iii) references.

(i). The document's heading is used to check with another document (cluster) if it is related to each other or not. For example, if the document heading is "Network Security and its applications in modern society," the document that forms a cluster is stored in the Network related clusters in serial order (N_1). It is easier to check the documents only with these clusters.

(ii). The document's body can only be checked with other related documents, i.e., related clusters using the LESK algorithm, which will have been explained in detail.

(iii). References to the document are not considered because most authors and research scholars may have a standard reference, which is neglected during plagiarism detection.

Improved LESK Algorithm

Based on Vasilescu et al. (2004), the Improved LESK algorithm is proposed using a dynamic window. For that, we separated the text document (all the words), gave them a sense, and saved them in an array. As the algorithm moved forward, the context of the window also increased. The algorithm also took care of the missing target words. Hence, the number of ambiguities is calculated by counting the senses.

(i). Word <— senses (calculate how many words are in a sense) (ii).Sense_Count <— number of senses (calculate the sense count of the word)

(iii).Instance_Count <— Context window of size n, (whereas n = determined dynamically, calculate the output for every target word. The context window is dynamic. If, in any case, the given sense overlaps the target sense, then Instance_Count + = 1. The context window is the number of left and right words in the target words.

(iv).For every target word, the context vector is generated.

(v).Determine plagiarism rate.

Step 6:

The result is compared with the local database, and the simulation is stopped.

Modified SVM Classifier

The proposed system utilizes two paths to determine sentence similarity. The first path relies on traditional word-level comparison, as discussed earlier. The second path involves the use of a modified SVM classifier (Haloi et al., 2023). However, the first path's initial "word-level comparison" did not yield satisfactory results in identifying text similarity. Instead, it involves constructing an SVM classifier that can detect several types of lexical, syntactic, and semantic similarities. The development process of the SVM classifier consists of four main stages: extraction of negative and positive instances, computation of sentence similarity features, selection of relevant features, and construction the classifier. In the initial stage, the system extracts "Non-plagiarized" and "Plagiarized" cases from training documents to create a supervised training database. The second stage system calculates sentence similarity features for each case, encompassing several lexical, syntactic, and semantic text similarities. These features are then recorded along with their corresponding class labels. In the third stage, the system employs a filter feature selection technique using the Chi-square algorithm to rank the features and select the most discriminative ones. It detects efficient text plagiarism, covering diverse lexical, syntactic, and semantic plagiarism types. Finally, in the fourth stage, the system constructs the hyperplane equation using the modified SVM classification algorithm. It eliminates the need for extensive experimentation to find the optimal weighting coefficients for incorporating the features effectively. By adopting this systematic approach, the proposed system aims to detect text plagiarism precisely, emphasizing the importance of feature selection and SVM classification.

Positive Feature Extraction

The algorithm is designed to extract features from positive sentence pairs, where 'P_z' is the plagiarized passage and 'P_s' is the source passage. It seems to involve pre-processing steps and extracting features from each sentence pair. The pseudo-code representation of the algorithm is as follows:

Algorithm 1

```
function extract_features($P_z$, $P_s$):
# Segment pz into sentences
  sentences_ $P_z$ = segment_into sentences($P_z$)
# Segment ps into sentences
  sentences_ $P_s$ = segment_into_sentences($P_s$)
# Pre-process source sentences
  processed_sentences_ $P_s$ = preprocess(sentences_$P_s$)
# Initialize feature sets
  feature_set = []
# Loop through each sentence in sentences_pz
  for sentence_ $P_z$ in sentences_ $P_z$:
# Pre-process plagiarized sentence
    processed_sentence_ $P_z$ = preprocess(sentence_ $P_z$)
# Calculate shared unigrams
    shared_unigrams = calculate_shared_unigrams(processed_sentence_ $P_z$, processed_sentences_ $P_s$)
# Calculate Meteor score
    meteor_score = calculate_meteor_score(processed_sentence_ $P_z$, processed_sentences_ $P_s$)
# Add the feature set for this sentence pair to the feature_set list
    feature_set.append({'shared_unigrams': shared_unigrams, 'meteor_score': meteor_score})
# Return the feature set for all positive sentence pairs
  return feature_set
```

- **Plagiarized_sentence:** The actual plagiarized sentence extracted from the plagiarized passage 'P_z'.
- **source_sentence:** The corresponding source sentence extracted from the source passage 'P_s'.
- **shared_unigrams:** A list of unigrams that are shared between the plagiarized sentence and the corresponding source sentence.
- **meteor_score:** The similarity score between the plagiarized sentence and the corresponding source sentence calculated using the Meteor metric or any other chosen similarity metric.

Data structure 1

```
# Data structure for feature set of positive sentence pairs
feature_set = [
  {
    'plagiarized_sentence': '…',        # The actual plagiarized sentence
    'source_sentence': '…',             # The corresponding source sentence
    'shared_unigrams': […],             # List of shared unigrams between the sentences
    'meteor_score': 0.0                 # Meteor similarity score between the sentences
  },
  {
    'plagiarized_sentence': '…',
    'source_sentence': '…',
    'shared_unigrams': […],
    'meteor_score': 0.0
  },
  # Add more dictionaries for other positive sentence pairs as needed
]
```

The process's second step is constructing semantic matrices using semantic similarity between words. It calculates the semantic similarity between words using seven WordNet similarity metrics. The dimension of the semantic matrix is equal to the number of words in the joint matrix, and each cell in the semantic matrix corresponds to a word in the joint matrix. The semantic similarity metrics mentioned earlier (such as Path Length, Wu-Palmer Similarity, Leacock-Chodorow Similarity, Resnik Similarity, and Jiang-Conrath Similarity) can be used to quantify the similarity between any two words in the dataset based on their meanings as represented by the WordNet synsets. A high-level explanation: Path Similarity Metric (PSM) is based on the shortest path length between two synsets in the WordNet hierarchy. The shorter the path length, the more closely related the synsets are in meaning. The path similarity metric is a normalized measure that ranges from 0 to 1, where 0 indicates no similarity, and 1 indicates identical meanings (i.e., the same synset).

$$\text{PSM(A, B)} = \frac{1}{d+1} \tag{5}$$

Here, 'd' is the shortest path length between synsets A and B in the WordNet hierarchy. Wu-Palmer Similarity (WUP) is based on the depth of the Least Common Subsumer (LCS) and the depth of the two synsets being compared.

$$WUPSM\ (A,B) = \frac{2*depth(LCS\ (A,B))}{depth(A)+depth(B)} \tag{6}$$

Leacock-Chodorow Similarity (LCH) is based on the shortest path length between two synsets and the depth of the WordNet hierarchy.

$$LCHSM\ (A,B) = -log\frac{distance+1}{2*depth(A,B)} \tag{7}$$

Resnik Similarity (RES) is based on the information content of the Least Common Subsumer (LCS).

$$RESSM\ (A,B) = IC\ (LCS\ (A,B) \tag{8}$$

$$IC(c) = \log\frac{1}{P(c)} \tag{9}$$

$$P(c) = \frac{appearances(t)}{v} \tag{10}$$

Where P(c) probability of concept 'c' in the corpus, the corpus that can inferred by 'c' and 'v' is the total number of corpus terms, Jiang-Conrath Similarity (JCN) is based on the information content of the two synsets being compared, and the information content of their Least Common Subsumer (LCS).

$$JCNSM(A,B) = \frac{1}{IC(A)+IC(B)-2*IC(LCS(A,B))} \tag{11}$$

These metrics are used to measure the similarity between two synsets in WordNet, and they are commonly used in natural language processing tasks that require assessing the semantic relatedness between words or

Algorithm 2

```
function detect_plagiarism(suspicious_document, source_document):
    # Step 1: Identify filter seeds for potential plagiarism segments
    filter_seeds = generate_filter_seeds()

    # Step 2: Apply filter seeds to identify potential plagiarism segments
    suspicious_segments = filter_plagiarism_segments(suspicious_document, filter_seeds)
    source_segments = filter_plagiarism_segments(source_document, filter_seeds)

    # Step 3: Merge adjacent detected seeds to form larger segments
    suspicious_segments = merge_adjacent_segments(suspicious_segments)
    source_segments = merge_adjacent_segments(source_segments)

    # Step 4: Apply adaptive behavior (adjust parameters if required)
    adjust_parameters()

    # Step 5: Filter segments to remove false positives
    filtered_suspicious_segments = filter_segments(suspicious_segments)
    filtered_source_segments = filter_segments(source_segments)

    # Step 6: Apply extension techniques to improve recall and granularity
    extended_suspicious_segments = extend_segments(filtered_suspicious_segments,
suspicious_document)
    extended_source_segments = extend_segments(filtered_source_segments, source_document)

    # Step 7: Compare extended segments to find the best-plagiarized segments
    best_plagiarized_segments = compare_segments(extended_suspicious_segments,
extended_source_segments)

    # Step 8: Return the best-plagiarized segments
    return best_plagiarized_segments
```

concepts. In the third step of the process, the algorithm computes semantic similarity measurements between two sets of words, namely A and B. It utilizes three measures of semantic similarity: cosine measure, dice measure, and Jaccard measure. Cosine similarity measures the cosine of the angle between two vectors in a high-dimensional space. It ranges from -1 (completely dissimilar) to 1 (completely similar). In semantic similarity, the vectors represent the word embeddings or semantic representations of the sets A and B. The Dice similarity coefficient measures the similarity between two sets. It is the ratio of the size of the intersection of the two sets to the average of their sizes. It ranges from 0 (completely dissimilar) to 1 (completely similar). For semantic similarity, this measure can be calculated using the semantic matrix. The Jaccard similarity coefficient is also used to measure the similarity between two sets. It is the ratio of the size of the intersection of the two sets to the size of their union. Like the Dice measure, it ranges from 0 (completely dissimilar) to 1 (completely similar). The Jaccard measure can also be calculated using the semantic matrix. The semantic similarity evaluates the degree of similarity between the word sets A and B based on the semantic matrix, providing insights into their semantic relationship in the context of plagiarism detection or any other relevant natural language processing task.

List of dictionaries to represent the detected plagiarism segments and their properties. Each dictionary in the list will represent a segment and contain information such as the start and end positions of the segment in the document and the segment’s similarity score.

Data Structure 2

```
# Data structure for detected plagiarism segments
plagiarism_segments = [
    {
        'start_position': 0,       # Start position of the segment in the document
        'end_position': 50,        # End position of the segment in the document
        'similarity_score': 0.85   # Similarity score of the segment (e.g., cosine similarity)
    },
    {
        'start_position': 100,
        'end_position': 130,
        'similarity_score': 0.91
    },
    # Add more dictionaries for other detected segments as needed
]
```

Step 1: Feature Extraction

```
function extract_features(document):
    # Extract relevant features from the document
    features = []

    # … Feature extraction process as discussed earlier …
    return features
```

Step 2: Feature Selection

```
function select_features(features):
    # Select the most effective subset of features
    selected_features = []

    # … Feature selection process as discussed earlier …
return selected_features
```

Step 3: SVM Training

```
function train_svm_classifier(train_data, labels, selected_features):
    # Train the SVM classifier using the selected features
    svm_model = SVM.train(train_data[:, selected_features], labels)

    return svm_model
```

Step 4: Hyperplane Equation

```
function get_hyperplane_equation(svm_model):
    # Obtain the hyperplane equation from the trained SVM model
    hyperplane_equation = svm_model.coefficients

    return hyperplane_equation
```

Step 5: Plagiarism Detection

```
function detect_plagiarism(suspicious_document, source_document, svm_model, selected_features,
threshold):
    # Extract features from suspicious and source documents
    suspicious_features = extract_features(suspicious_document)
    source_features = extract_features(source_document)
```

```
Calculate similarity scores using the SVM classifier and hyperplane equation
  suspicious_score = SVM.predict(suspicious_selected_features, svm_model)
  source_score = SVM.predict(source_selected_features, svm_model)

Determine if the documents are plagiarized or not based on similarity scores
  if suspicious_score > threshold and source_score > threshold:
      return "Plagiarized"
  else:
      return "Not Plagiarized"
```

Result and Discussion

The proposed algorithm has worked for all types of English letter language papers. The simulation is run under the Python and Oracle 11g environments. The result is evaluated using simple text files in different cases. However, during the testing phase, we noticed that Plagiarism detection tools were not giving the exact result. While evaluating the plain text, we modified it by different senses and references, but it does not give the exact result. In the testing phase, we checked sentence to sentence to make it easier to detect the plagiarism rate.

TEST CASE 1

Here, we copied the text from sentence to sentence by copying it exactly from another text document. So, we took plain text with 80% copied and 20% not copied. Then, we noticed that it gave an 80% plagiarism detection rate, summarized in *Table 1*.

Table 1. Plagiarism detection rate at copied and self-analysis text.

Text Data	Detection Rate
Copied text	80%
Self-Analysis text	20%
Plagiarizes detection rate	80%

Figure 2 represents a copied and self-analysis text. The higher percentage of the copied text shows that the plagiarism rate is much higher.

TEST CASE 2:

Here, we copied the text, sentence to sentence, by copying exactly from another text document. In this case, we have taken 30% of the copied text, and the original text changes 50% of the sentence, but the sense remained constant, and 20% of the text is not copied, i.e., the self-analyzed text. *Table 2* represents the 60% plagiarism detection rate, which means it does not give an accurate plagiarism rate.

Figure 3 represents the comparison of copied text and changed text sense, where changed text sense has a higher percentage of plagiarism.

TEST CASE 3:

In this case, we have taken 10% of the copied text, and the original text changes 70% of the sentence, but the

Table 2. Plagiarism detection rate at copied, changed text and self-analysis text.

Text Data	Detection Rate
Copied Text	30%
Changed text, but the sense is the same	50%
Self-Analyzed text	20%
Plagiarism rate	60%

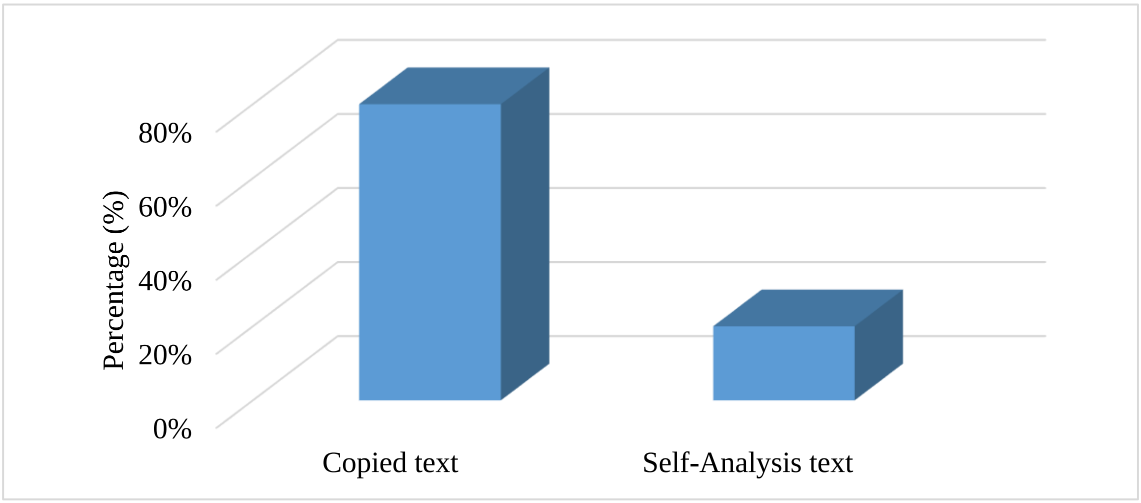

Figure 2. The plagiarism detection rate of copied and self-analysis.

sense remained constant, and 20% of the text is not copied, i.e., self-analyzed text. Then we noticed that it gives nearly a 55% plagiarized detection rate, which means ‘t’ does not give an accurate plagiarism rate. From this case, we can say that plagiarism can also calculate the senses with the help of WSD, which is summarized in *Table 3*.

Figure 4 explains the exact meaning of getting a high plagiarism rate. At the same time, copied and self-analyzed text gets minor plagiarism during the detection. To find the accuracy of the proposed method, we compared the result with a different research paper. As we saw in test case 1 for self-analyzed text, the accuracy for plagiarism detection was 80%.

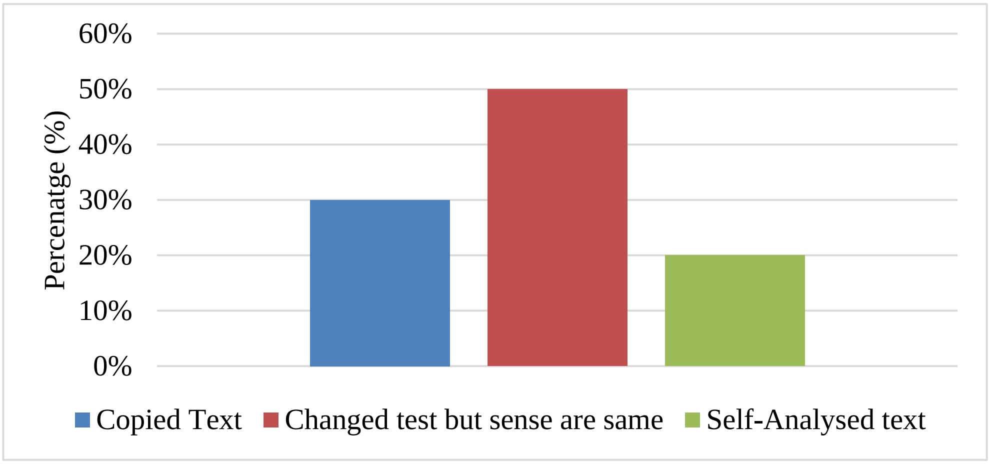

Figure 3. The plagiarism detection rate of copied, changed text and self-analysis text.

Table 3. The plagiarism detection rate of the original text but senses is changed.

Text Data	Detection Rate
Copied Text	10%
Changed text, but the sense is the same	70%
Self-Analyzed text	20%
Plagiarism rate	~55%

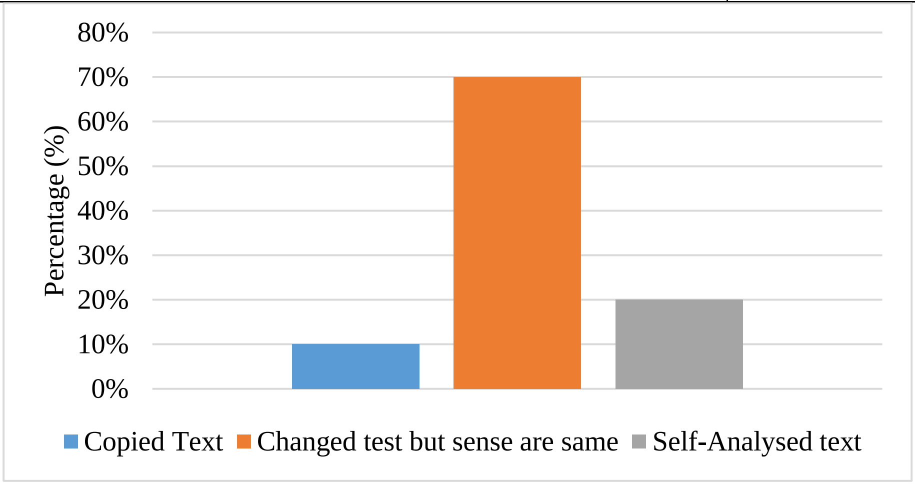

Figure 4. Graphical Representation of Test Case 3.

Table 4. Comparison of different research papers with the proposed method.

Work	Accuracy (%)
Sánchez-Vega et al. (2013)	75.9 %
Chong et al. (2010)	70.53 %
Vani & Gupta (2017)	83.16 %
Proposed System	80 %

Table 4 represents the comparison of the proposed model with Vani & Gupta (2017), Chong et al. (2010), and Sánchez-Vega et al. (2013). The proposed model uses the self-analyzed text, which gave 80% accuracy higher than Chong et al. (2010) and Sánchez-Vega et al. (2013). While comparing with Vani & Gupta (2017), the proposed system will provide nearly the same accuracy rate.

Table 5. Comparison results on PAN 2012, PAN 2013, and PAN 2014 complete corpus data.

Dataset	PlagDet (%)	F-measure (%)	Recall (%)	Precision (%)	Granularity
PAN 2012	88.44	88.54	86.79	90.54	1.00019
PAN 2013	89.54	89.83	86.79	93.12	1.00034
PAN 2014	92.84	92.52	90.32	95.54	1.00052

Evaluation Based on Dataset

The experiments described in the text were conducted on three datasets from the PAN Workshop series: PAN 2012, PAN 2013, and PAN 2014. Each dataset contains suspicious and source documents, and the authors applied various obfuscation strategies on paragraphs of different lengths of the source documents, which were then incorporated into the suspicious documents. Below is a summary of the datasets and their characteristics:

- **PAN 2012 Dataset:** Data Source: Books available at Project Gutenberg, Training Set: 1804 suspicious documents and 4210 source documents, and Test Set: 3000 suspicious documents and 3500 source documents.
- **PAN 2013 Dataset:** Data Source: ClueWeb 2009 corpus, Total Documents: 3653 suspicious documents and 4774 source documents.
- **PAN 2014 Dataset:** Data Source: ClueWeb 2009 corpus (same as PAN 2013), Training Set: Same as PAN 2013 (3653 suspicious documents and 4774 source documents), and Test Set: Introduced an additional test set.

The datasets were used for evaluating plagiarism detection algorithms, and they are publicly available as part of the PAN Workshop series. The training sets are used to train the plagiarism detection models, while the test sets are used to evaluate the performance of the models on unseen data.

Table 5 evaluates the effectiveness of the proposed system for text plagiarism detection. The experiments use three datasets created from the documents of PAN 2012, PAN 2013, and PAN 2014. These experiments aim to assess the proposed system's performance in detecting different types of text plagiarism. The First Dataset contains positive and negative cases extracted from the documents of PAN 2012. The purpose is to construct the proposed system for plagiarism detection. The proposed system is constructed and trained on this dataset. Dataset content extracting cases from the documents of PAN 2012 and PAN 2013. The purpose is to train the proposed system's Support Vector Machine (SVM). The SVM classifier of the proposed system is trained on this dataset to discover several types of text plagiarism. The system's performance is evaluated on test documents from the random obfuscation PAN 2013 sub-corpora, complete PAN 2013 corpus, and complete PAN 2014. The comparison of results indicates that the proposed system achieved the highest classification accuracy when trained on the dataset. It suggests that the proposed system is more effective in training the SVM classifier.

In summary, *Table 6* analyses the significance of each feature in the constructed training database for text plagiarism detection. The metrics, such as mean, standard deviation, and 95% confidence limits, help assess the features' ability to distinguish between positive (plagiarized) and negative (non-plagiarized) cases. The importance of similarity features in discriminative sentences lies in their ability to accurately differentiate positive and negative cases. Features with high intra-similarity (within the same class) and low inter-similarity (between different classes) are considered discriminative. The findings from *Table 6* show that many sentences similarity features have mean values close for both positive and negative cases.

Additionally, the 95% confidence limits of the positive cases are far from 1 and closer to the mean values of both positive and negative cases. Relying solely on these features for decision-making may lead to confusion and inaccurate classification. Combining multiple features, particularly those with higher intra-similarity and lower inter-similarity, is crucial to enhancing text plagiarism detection's accuracy and effectiveness. By considering various informative features, the system can better distinguish between plagiarized and non-plagiarized text segments, resulting in more reliable detection outcomes.

The results presented in *Table 7* show that most of the previous systems achieved varying ranks across different datasets. This variation in performance is attributed to differences in dataset structures and the types of plagiarism present in each dataset. On the other hand, the

Table 6. Statistical Analysis

Statistical analysis of the constructed dataset	The Extracted Cases	
Sentence Similarity Feature	**Negative N = 42983 Mean ± Standard Deviation**	**Positive N = 42983 Mean ± Standard Deviation**
Syntactic with word path similarity	0.5123 ± 0.1190	0.8414 ± 0.151
Dice Semantic with word path similarity	0.5321 ± 0.0785	0.7521 ± 0.202
Jaccard Semantic with word path similarity	0.4231 ± 0.0185	0.8502 ± 0.1414
Cosine Semantic with word path similarity	0.5123 ± 0.1185	0.9083 ± 0.1312
Syntactic with word depth estimation	0.3174 ± 0.1086	0.7327 ± 0.1959
Dice Semantic with word depth estimation	0.2124 ± 0.0145	0.7407 ± 0.1832
Jaccard Semantic with word depth estimation	0.6104 ± 0.1058	0.7673 ± 0.1844
Cosine Semantic with word depth estimation	0.3174 ± 0.0157	0.6574 ± 0.2374
Syntactic with combined word similarity	0.4144 ± 0.0722	0.7781 ± 0.1746
Dice Semantic with combined word similarity	0.3124 ± 0.0988	0.7227 ± 0.1956
Jaccard Semantic with combined word similarity	0.4134 ± 0.1105	0.6908 ± 0.2362
Cosine Semantic with combined word similarity	0.5211 ± 0.1975	0.5092 ± 0.2925
Syntactic with WUP word similarity	0.3123 ± 0.0125	0.7361 ± 0.1832
Dice Semantic with WUP word similarity	0.4214 ± 0.0315	0.4086 ± 0.4094
Jaccard Semantic with WUP word similarity	0.5141 ± 0.0765	0.3562 ± 0.3924
Cosine Semantic with WUP word similarity	0.8124 ± 0.0359	0.9005 ± 0.1089
Syntactic with LCH word similarity	0.3148 ± 0.0421	0.8342 ± 0.1563
Dice Semantic with LCH word similarity	0.5818 ± 0.0512	0.9073 ± 0.1655
Jaccard Semantic with LCH word similarity	0.3289 ± 0.0251	0.7391 ± 0.1834
Cosine Semantic with LCH word similarity	0.5221 ± 0.0321	0.7653 ± 0.1459
Syntactic with RES word similarity	0.4312 ± 0.0612	0.6438 ± 0.2059
Dice Semantic with RES word similarity	0.3276 ± 0.0562	0.7743 ± 0.1454
Jaccard Semantic with RES word similarity	0.3498 ± 0.0663	0.7348 ± 0.1844
Cosine Semantic with RES word similarity	0.3212 ± 0.0432	0.4859 ± 0.287
Syntactic with JCN word similarity	0.3211 ± 0.0123	0.8414 ± 0.151
Dice Semantic with JCN word similarity	0.4331 ± 0.0331	0.7521 ± 0.202
Jaccard Semantic with JCN word similarity	0.3765 ± 0.0447	0.8502 ± 0.1414
Cosine Semantic with JCN word similarity	0.4587 ± 0.0234	0.9083 ± 0.1312
Syntactic with LIN word similarity	0.4377 ± 0.0221	0.5772 ± 0.2796
Dice Semantic with LIN word similarity	0.3786 ± 0.0322	0.6999 ± 0.2294
Jaccard Semantic with LIN word similarity	0.3986 ± 0.0412	0.7409 ± 0.1831
Cosine Semantic with LIN word similarity	0.3753 ± 0.0187	0.7306 ± 0.2085
Hybrid similarity	0.3997 ± 0.0322	0.6173 ± 0.2581
Fuzzy Semantic	0.4689 ± 0.0132	0.7409 ± 0.1998

proposed system consistently maintained its rank and demonstrated superior performance across the different datasets. These results indicate that the proposed system exhibits efficiency and robustness in detecting various forms of text plagiarism. The success of the proposed system can be attributed to the support vector machine (SVM) algorithm's ability to find the hyperplane equation of the selected 32 features, enabling it to effectively detect different types of text similarities. The proposed system's adaptive behaviour has enhanced the Plagdet score in sub-corpus without negatively affecting the recall value in the no-obfuscation sub-corpus. These findings highlight the adaptability and effectiveness of the proposed system in tackling different challenges posed by several types of text plagiarism in diverse datasets.

Discussion

The techniques proposed in this paper hold significant practical implications for real-world educational settings, particularly in Word Sense Disambiguation (WSD). By leveraging the Adapted Lesk algorithm and a knowledge-based approach using WordNet, these methods offer a promising avenue for enhancing language understanding and educational tools. Implementation of these techniques in educational settings can offer numerous benefits. Firstly, utilizing widely available resources such as WordNet facilitates accessibility and reduces implementation barriers.

Table 7. Comparison of Rank Score.

Comparison of the result of the proposed system on the complete PAN 2014 dataset					
Team	**Plagdet (%)**	**F-measure (%)**	**Recall (%)**	**Precision (%)**	**Granularity**
Gillam & Notley (2014)	44.08	44.07	29.66	85.74	1.0000
Abnar et al., (2014)	66.38	66.59	84.78	54.83	1.00455
Palkovskii and Belov (2014)	90.78	90.80	88.92	92.76	1.00027
Oberreuter et al., (2014)	89.27	89.30	91.54	87.17	1.00051
Sanchez-Perez et al., (2014)	89.20	89.21	91.98	86.61	1.00026
Glinos (2014)	88.77	89.89	84.51	96.01	1.01761
Shrestha et al., (2014)	86.81	87.05	89.84	84.42	1.00381
Gross & Modaresi, (2014)	85.50	86.84	81.82	92.52	1.02187
PlagLinSVM (2020)	90.01	90.15	90.55	89.75	1.00210
PlagRbfSVM (2020)	88.27	88.40	91.49	85.52	1.00209
El-Rashidy et al. (2023)	92.91	92.95	90.14	95.94	1.00053
Proposed System Using SVM	**92.86**	**92.50**	**90.84**	**95.82**	**1.00087**

Educational platforms can integrate these methods seamlessly into their existing infrastructure, allowing educators and students to benefit from more accurate and nuanced language analysis. The structured approach outlined in the manuscript, consisting of input query processing, pre-processing, and WSD classification units, provides a clear framework for implementation. The structured workflow ensures that the system can effectively handle unstructured queries from users, process them into structured queries with added linguistic features, and accurately classify the sense of polysemous words based on context and lexical data. Despite these advantages, several potential challenges may arise during the implementation process. One significant challenge is adapting the techniques to diverse educational contexts, each with unique linguistic and pedagogical requirements. Ensuring the methods are applicable across various subjects, languages, and educational levels may require additional customization and fine-tuning. Integrating these techniques into existing educational systems may pose technical challenges. Educational platforms often have diverse architectures and interfaces, requiring careful consideration to ensure compatibility and seamless integration. Providing adequate training and support for educators and students to use these tools effectively is essential for successful implementation. Scalability and computational requirements are important considerations, especially in resource-constrained educational environments. Ensuring that the methods can operate efficiently within the constraints of educational infrastructure, such as limited computing resources and network bandwidth, is crucial for widespread adoption.

Conclusion

In this research, we thoroughly evaluated the proposed system for text plagiarism detection using three datasets constructed from PAN 2012, PAN 2013, and PAN 2014 documents—the experiments aimed to assess the system's performance in detecting several types of text plagiarism. The First Dataset extracted positive and negative cases from the documents of PAN 2012 and served as the foundation for constructing the proposed plagiarism detection system. The Second Dataset included cases extracted from PAN 2012 and PAN 2013 and was instrumental in training the Support Vector Machine (SVM) classifier. The SVM classifier's training on this dataset facilitated the detection of multiple types of text plagiarism. We evaluated the system's performance on test documents from the random obfuscation PAN 2013 sub-corpora, complete PAN 2013 corpus, and complete PAN 2014. The results indicated that when the proposed system was trained on the Second Dataset, it achieved the highest classification accuracy. It highlights the system's effectiveness and superiority in training the SVM classifier. Metrics such as mean, standard deviation, and 95% confidence limits were instrumental in assessing the features' ability to distinguish between positive (plagiarized) and negative (non-plagiarized) cases. The importance of discriminative sentence similarity features, characterized by high intra-similarity within the same class and low inter-similarity between different classes, was evident. Many sentence similarity features were observed to have mean values close for both positive and negative cases, and the 95% confidence limits of positive cases were closer to the mean values of both positive and negative cases. It indicated that relying solely on these features for decision-making might lead to confusion and inaccurate classification. Thus, combining multiple features, particularly those with higher intra-similarity and lower inter-similarity, proved crucial in enhancing the accuracy and effectiveness of text plagiarism detection. Looking ahead, we plan to expand the system's training with a larger data set to evaluate its performance further. We also foresee potential applications of this work in detecting plagiarism in documents written in different languages by leveraging mathematical problem-solving techniques. We aim to explore the extension of this approach to detect duplicity in images using Word Sense Disambiguation (WSD) techniques. The proposed algorithm's results demonstrated superior performance to conventional techniques, validating its efficacy in text plagiarism detection. To enhance future executions, we aim to reduce time complexity by implementing parallel computing techniques. The research marks considerable progress in text plagiarism detection, with the proposed system showing promising results and potential for broader applications. As we continue to refine and extend our approach, we envision a significant contribution to ensuring the authenticity and originality of digital content in diverse contexts.

Conflicts of Interest

The authors declare no conflict of interest.

References

Abdi, A., Shamsuddin, S. M., Idris, N., Alguliyev, R. M., & Aliguliyev, R. M. (2017). A linguistic treatment for automatic external plagiarism detection. *Knowledge-Based Systems*, *135*, 135-146. https://doi.org/10.1016/j.knosys.2017.08.008

Abnar, S., Dehghani, M., Zamani, H., & Shakery, A. (2014). Expanded n-grams for semantic text alignment.In: CLEF (working notes) 1180:928-938. Available: http:// ceur- ws. org/ Vol- 1180/ CLEF2 014wn-Pan- Abnar Et2014. Pdf

Agarwal, J., Goudar, R. H., Kumar, P., Sharma, N., Parshav, V., Sharma, R., ... & Rao, S. (2013, August). Intelligent plagiarism detection mechanism using semantic technology: A different approach. *IEEE, In 2013 International Conference on Advances in Computing, Communications and Informatics* (ICACCI), pp. 779-783. https://doi.org/10.1109/ICACCI.2013.6637273

Altheneyan, A. S., & Menai, M. E. B. (2020). Automatic plagiarism detection in obfuscated text. *Pattern Analysis and Applications*, *23*, 1627-1650. https://doi.org/10.1007/s10044-020-00882-9

Alzahrani, S. M., Salim, N., & Abraham, A. (2012). Understanding Plagiarism Linguistic Patterns, Textual Features, and Detection Methods. *IEEE Transactions on Systems, Man, and Cybernetics, Part C* (Applications and Reviews), *42*(2), 133–149. https://doi.org/10.1109/tsmcc.2011.2134847

Ayetiran, E. F., & Agbele, K. (2016). An Optimized Lesk-Based Algorithm for Word Sense Disambiguation. *Open Computer Science*, *8*(1), 165–172. https://doi.org/10.1515/comp-2018-0015

Banerjee, S., & Pedersen, T. (2002, February). An adapted Lesk algorithm for word sense disambiguation using WordNet. Berlin, Heidelberg: Springer Berlin Heidelberg, *In International conference on intelligent text processing and computational linguistics*, pp. 136-145.

Basile, P., Caputo, A., & Semeraro, G. (2014, August). An enhanced lesk word sense disambiguation algorithm through a distributional semantic model. In Proceedings of COLING 2014, *the 25th International Conference on Computational Linguistics: Technical Papers*, pp. 1591-1600.

Chong, M., Specia, L., & Mitkov, R. (2010, June). Using natural language processing for automatic plagiarism detection. *In Proc. of 4th International Plagiarism Conference*, Northrumbia University Newcastle-upon-Tyne, UK.

El-Rashidy, M. A., Mohamed, R. G., El-Fishawy, N. A., & Shouman, M. A. (2023). An effective text plagiarism detection system based on feature selection and SVM techniques. *Multimedia Tools and Applications*, *83*(1), 2609–2646. https://doi.org/10.1007/s11042-023-15703-4

En, A. C. M., Karim, A. A., Noor, N. M., & Majid, M. Z. A. (2023). Plagiarism Experience among Higher Education Students. *International Journal of Academic Research in Business and Social Sciences*, *13*(9), 1877–1883. http://dx.doi.org/10.6007/IJARBSS/v13-i9/18611

Gillam, L., & Notley, S. (2014, September). Evaluating Robustness for 'IPCRESS': Surrey's Text Alignment for Plagiarism Detection-Notebook for PAN at CLEF 2014. *In CLEF 2014 Evaluation Labs and Workshop—Working Notes Papers*, 15-18 September, Sheffield, UK (pp. 951-957). CEUR-WS. org.

Gipp, B., Meuschke, N., & Breitinger, C. (2014). Citation-based plagiarism detection: Practicability on a large-scale scientific corpus. *Journal of the Association for Information Science and Technology*, *65*(8), 1527–1540. https://doi.org/10.1002/asi.23228

Glinos, D. G. (2014, September). A Hybrid Architecture for Plagiarism Detection. *In CLEF* (working notes), pp. 958-965.

Gross, P., & Modaresi, P. (2014, September). Plagiarism Alignment Detection by Merging Context Seeds. *In CLEF* (working notes), pp. 966-972.

Haloi, R., Chanda, D., Hazarika, J., & Barman, A. (2023). Statistical feature-based EEG signals classification using ANN and SVM classifiers for Parkinson's disease detection. Int. J. Exp. Res. Rev., 31(Spl Volume), 141-149. https://doi.org/10.52756/10.52756/ijerr.2023.v31spl.014

Hiremath, S. A., & Otari, M. S. (2014). Plagiarism detection-different methods and their analysis. *International Journal of Innovative Research in Advanced Engineering, 1*(7), 41-47.

Joshi, M., & Khanna, K. (2013). Plagiarism Detection over the Web: Review. *International Journal of Computer Applications, 68*(15), 17–20. https://doi.org/10.5120/11655-7163

Kumar, M., Mukherjee, P., Hendre, M., Godse, M., & Chakraborty, B. (2020). Adapted Lesk Algorithm based Word Sense Disambiguation using the Context Information. International *Journal of Advanced Computer Science and Applications, 11*(3). https://doi.org/10.14569/ijacsa.2020.0110330

Kumari, L., & Kumar, S. (2023). Optimizing word sense disambiguation for Hindi language using extended Lesk and conceptual density. *8th International Conference on Computing in Engineering and Technology* (ICCET 2023). https://doi.org/10.1049/icp.2023.1493

Mahdavi, P., Siadati, Z., & Yaghmaee, F. (2014, October). Automatic external Persian plagiarism detection using vector space model. IEEE, *In 2014 4th International Conference on Computer and Knowledge Engineering* (ICCKE), pp. 697-702. https://doi.org/10.1109/ICCKE.2014.6993398

Manning, C. D., Raghavan, P., & Schütze, H. (2008). Text classification and naive bayes. *Introduction to Information Retrieval, 1*(6). https://doi.org/10.1017/CBO9780511809071.014

Maurya, A., & Madhusudhan, M. (2023). Plagiarism in Research: Problems and its Solutions. *Journal of Advancements in Library Sciences, 10*(1), 59–69. https://doi.org/10.37591/joals.v10i1.3688

Mentari, M., Rozi, I. F., & Rahayu, M. P. (2022). Cross-Language Text Document Plagiarism Detection System Using Winnowing Method. *Journal of Applied Intelligent System, 7*(1), 44–57. https://doi.org/10.33633/jais.v7i1.5950

Mozgovoy, M. (2011). Dependency-based rules for grammar checking with LanguageTool. IEEE, *In 2011 Federated Conference on Computer Science and Information Systems* (FedCSIS), pp. 209-212.

Nguyen, Q. H. (2023). AI and Plagiarism: Opinion from Teachers, Administrators and Policymakers. *Proceedings of the Asia CALL International Conference, 4*, 75–85. https://doi.org/10.54855/paic.2346

Oberreuter, G., Carrillo-Cisneros, D., Scherson, I. D., & Velásquez, J. D. (2014). Submission to the 4th international competition on plagiarism detection. In Proc. of 2014 Cross Language Evaluation Forum Conference, Working Notes Papers of the CLEF 2014 Evaluation Labs, CEUR Workshop Proceedings.

Palkovskii, Y., & Belov, A. (2014). Developing high-resolution universal multi-type n-gram plagiarism detector. Working Notes Papers of the CLEF 2014 Evaluation Labs, 984-989.

Prasanth, S., & Rajshree, R. (2014). A Survey on Plagiarism Detection. *International Journal of Computer Applications, 86*(19). https://doi.org/10.5120/15104-3428

Ranjan Pal, A., Kundu, A., Singh, A., Shekhar, R., & Sinha, K. (2013). Hybrid Approach to Word Sense Disambiguation Combining Supervised and Unsupervised Learning. *International Journal of Artificial Intelligence & Applications, 4*(4), 89–101. https://doi.org/10.5121/ijaia.2013.4409

Sanchez-Perez, M. A., Sidorov, G., & Gelbukh, A. F. (2014). A Winning Approach to Text Alignment for Text Reuse Detection at PAN 2014. CLEF (Working Notes), 2014, 1004-1011.

Sánchez-Vega, F., Villatoro-Tello, E., Montes-y-Gómez, M., Villaseñor-Pineda, L., & Rosso, P. (2013). Determining and characterizing the reused text for plagiarism detection. *Expert Systems with Applications, 40*(5), 1804-1813.

Sedaghat, S. (2024). Plagiarism and Wrong Content as Potential Challenges of Using Chatbots Like ChatGPT in Medical Research. *J. Acad. Ethics*, pp.1-3. https://doi.org/10.1007/s10805-024-09533-8

Shrestha, P., Maharjan, S., & Solorio, T. (2014). Machine Translation Evaluation Metric for Text Alignment. In CLEF (working notes), pp. 1012-1016.

Slimani, T. (2013). Description and evaluation of semantic similarity measures approaches. *arXiv preprint arXiv*, 1310.8059.

Upadhyay, D. K., Mohapatra, S., Singh, N. K., & Bakhla, A. K. (2021). Stacked SVM model for Dysthymia prediction in undergraduates students. IEEE, *In 2021 8th International Conference on Signal Processing and Integrated Networks* (SPIN), pp. 1148-1153.

Vani, K., & Gupta, D. (2017). Text plagiarism classification using syntax-based linguistic features. *Expert Systems with Applications*, *88*, 448-464. https://doi.org/10.1016/j.eswa.2017.07.006

Vasilescu, F., Langlais, P., & Lapalme, G. (2004, May). Evaluating Variants of the Lesk Approach for Disambiguating Words. In Lrec.

Vrbanec, T., & Meštrović, A. (2020). Corpus-Based Paraphrase Detection Experiments and Review. Information, 11(5), 241. https://doi.org/10.3390/info11050241

Weber-Wulff, D. (2018). Why does plagiarism detection software not find all plagiarism? *Student Plagiarism in Higher Education*, pp. 62–73. https://doi.org/10.4324/9781315166148-5

How to cite this Article:

Devesh Kumar Upadhyay and Keshav Sinha (2024). Enhancing Academic Integrity: An Analysis of Advanced Techniques for Plagiarism Detection using LESK, Word Sense Disambiguation, and SVM. *International Journal of Experimental Research and Review*, *39*(spl.) 92-108.

DOI: https://doi.org/10.52756/ijerr.2024.v39spl.007