Online-Journals.org

iJIM International Journal of Interactive Mobile Technologies

iJIM | eISSN: 1865-7923 | Vol. 19 No. 3 (2025) | OPEN ACCESS

https://doi.org/10.3991/ijim.v19i03.49897

PAPER

NLP-Enhanced Techniques for Cheating Detection in Virtual Exams: A Comparative Study of String and Semantic Similarity Measures with K-Shingling, Minhashing, LSH, and K-Means

Nabila El Rhezzali(✉), Imane Hilal, Meriem Hnida

ITQAN Team, LyRica Lab, Information Sciences School, Rabat, Morocco

nabila.el-rhezzali@esi.ac.ma

ABSTRACT

As online learning gains popularity, the issue of cheating becomes a topic of interest in discussions and scientific papers about virtual and distance education. The shift from in-person to online exams has raised many concerns about its potential to make cheating easier. The best way to detect cheating in online exams is to compute the closeness between answers. In this paper, we introduce a new approach based on similarity measure techniques from text mining to analyze answers in order to identify common patterns among academic responses. For this purpose, we have improved the K-Shingling-Minhashing-locality sensitive hashing (LSH) technique by incorporating the K-means clustering step, and then we applied our approach to a use case to efficiently find similar answers in a large dataset. Comparing our results with related works underscores the efficiency and applicability of our method in educational contexts, offering a novel contribution to the field of text similarity detection by highlighting the reliability and consistency of mathematical methods.

KEYWORDS

cheating detection, online exams, string-based similarity, semantic-based similarity, K-Shingling, winnowing, Minhashing, locality sensitive hashing (LSH), k-means

1 INTRODUCTION

E-learning refers to an online and distance education approach that leverages the internet and educational technologies such as smartphones, computers, and tablets. This mode of learning empowers students with the flexibility to access educational resources and engage in learning experiences at their convenience, regardless of their location. However, the challenge for many universities is to detect cheating in exams, especially

El Rhezzali, N., Hilal, I., Hnida, M. (2025). NLP-Enhanced Techniques for Cheating Detection in Virtual Exams: A Comparative Study of String and Semantic Similarity Measures with K-Shingling, Minhashing, LSH, and K-Means. *International Journal of Interactive Mobile Technologies (iJIM)*, 19(3), pp. 56–72. https://doi.org/10.3991/ijim.v19i03.49897

Article submitted 2024-04-29. Revision uploaded 2024-10-31. Final acceptance 2024-11-01.

when conducted remotely. One method for detecting cheating involves comparing the similarity between students' answers. The concept of similarity has been extensively discussed within the domains of linguistics, philosophy, and computer science. Two primary ways in which words can exhibit similarity are lexical and semantic. Lexical similarity pertains to words that share a resemblance in their character sequences [1]. Various string-based algorithms measure lexical similarity by analyzing sequences of characters and strings, helping to quantify the likeness or dissimilarity between text strings. These metrics are particularly useful for tasks such as string comparison or matching [2]. Semantic similarity, on the other hand, encompasses the equivalence of meaning between words. Words are considered semantically similar when they share the same meaning and are used in similar contexts. Two primary approaches for measuring semantic similarity include corpus-based and knowledge-based methods. Corpus-based similarity leverages information from extensive corpora to identify similarities between words [1]. This approach is underpinned by distributional semantics, which posits that words occurring in similar contexts tend to have similar meanings. Techniques such as latent semantic analysis (LSA) [3] and Word2Vec [4] have been widely used in corpus-based approaches to assess word similarity. Knowledge-based similarity relies on information from semantic networks, such as WordNet, to identify similarities among words [5]. This approach utilizes structured knowledge about word meanings and their relationships within a network to measure semantic similarity. Algorithms such as the Lesk algorithm and the Wu-Palmer measure exemplify knowledge-based techniques for computing semantic similarity.

Several studies have explored methods for detecting cheating in online exams using similarity measures. For instance, the authors in [6] investigated the effectiveness of string-based similarity metrics in detecting collusion among students in programming exams. Similarly, the authors in [7] examined the application of semantic similarity measures to identify instances of plagiarism in student essays. By employing these advanced methodologies, educational institutions can analyze students' answers to identify the closeness between responses, thus enhancing the ability to detect cheating in online exams. In summary, understanding both lexical and semantic similarity is crucial for developing robust methods to detect cheating in e-learning environments. By employing these advanced methodologies, we can analyze students' answers in order to identify the closeness between answers and then detect cheating in online exams.

In online learning, students can study whenever and wherever they want. E-learning is now essential for teachers all over the world. In the past, not everyone could access knowledge due to their geographical locations. Borders made it hard for teachers and students to travel. Nowadays, online education allows for consistent learning, better teamwork, and access to education worldwide for both students and teachers. As e-learning becomes more common, and as the shift from traditional classroom-administered exams to online assessments has repeatedly occurred because of COVID-19. The cheating issues have become more prevalent and difficult to detect.

A significant challenge faced by numerous educational institutions pertains to the identification of cheating, particularly in the context of remote examinations. Some students cheat individually, while others may collaborate. In an online setting, it's challenging to detect and stop students from working together. Certain students may gradually attempt to collaborate during their online exams, whether they are in the same location or different places, using the internet or social networks for communication. Our objective in this study is to identify cheating in online exams through the use of similarity techniques to measure the resemblance between student responses.

The remainder of this paper is organized as follows: Section 2 presents the background and an overview of common similarity measures, and Section 3 presents some related works. Our proposed methodology based on K-Shingling, Minhashinh,

locality sensitive hashing (LSH), and K-means is presented in Section 4 and discussed in Section 5. Finally, Section 6 concludes the present work and provides perspectives.

2 BACKGROUND

The concept of similarity has been extensively discussed. However, several methods and algorithms are used to measure similarity, such as string-based similarity and semantic-based-similarity.

2.1 String similarity

String similarity is a measure of how much two text strings have in common. Commonly utilized in natural language processing (NLP) activities such as spell checking, text classification, and information retrieval. String similarity algorithms can also be used to detect plagiarism and recommend similar articles to users. A string-based similarity measure is introduced in this paper through character-based similarity and term-based similarity.

Character-based similarity measures. Character-based similarity, or edit distance, is string-based similarity that takes two strings of characters and then computes the distance between them. Many measures are used to compute character-based similarity: Levenshtein distance, Jaro distance, Jaro–Winkler, the longest common substring, the longest common subsequence, Needleman-Wunsch, and Smith-Waterman. The following subsections present these measures in detail. Table 1 provides an overview of each term-based algorithm's metric, some properties, and common usage.

Table 1. Comparison of character-based similarity metric

Algorithm	Properties	Usage	Equation
Levenshtein	Measures edit operations (insertion, deletion, substitution) to transform one string into another.	– Spell checking – Text classification – Information retrieval – Plagiarism detection	$Lev_{a,b} = \begin{cases} \max(i,j) & \text{if } \min(i,j)=0, \\ \min\begin{cases} lev_{a,b}(i-1,j)+1 \\ lev_{a,b}(i,j-1)+1 \\ lev_{a,b}(i-1,j-1)+1_{(a_i \neq b_i)} \end{cases} & \text{otherwise} \end{cases}$
Jaro	Measures similarity between two strings based on the count of matching characters and transpositions.	– Text comparison – Record linkage	$Jaro\ dist = \begin{cases} 0 & \text{if } m=0 \\ \frac{1}{3}\left(\frac{m}{\lvert s1\rvert}+\frac{m}{\lvert s2\rvert}+\frac{m-t}{m}\right) & \text{if } m!=0 \end{cases}$ Where: – m represents the count of matching characters. – t denotes the count of transpositions, which is half the number of matching characters in the two strings but in a different order. – '$\lvert s1\rvert$' and '$\lvert s2\rvert$' represent the respective lengths of strings '$s1$' and '$s2$'."
Jaro-Winkler	Similar to Jaro distance but incorporates a prefix scale.	– Text comparison with consideration for common prefixes	$S_w = S_j + P*L*(1-S_j)$ Where: S_j represents the Jaro similarity. S_w represents the Jaro-Winkler similarity. P denotes the scaling factor, with a default value of 0.1. L signifies the length of the common prefix, limited to a maximum of 4 characters.

(Continued)

Table 1. Comparison of character-based similarity metric *(Continued)*

Algorithm	Properties	Usage	Equation
Longest Common Substring (LCS)	Finds the longest contiguous substring shared between two strings.	– Text comparison – Plagiarism detection	$LCS(i,j)=\begin{cases} LCS(i,j)=0 & if\ i=0\ ou\ j=0 \\ LCS(i,j)=LCS(i-1,j-1) & if\ X[i]=Y[j] \\ LCS(i,j)=0 & if\ X[i]\neq Y[j] \end{cases}$
Longest Common Subsequence (LCS)	Finds the longest subsequence (not necessarily contiguous) shared between two strings.	– DNA sequencing – Text comparison – Plagiarism detection	$LCS(i,j)=\begin{cases} f(i,j)=1+f(i-1,j-1) & if\ A_i=B_j \\ f(i,j)=\max(f(i-1,j),f(i,j-1)) & if\ A_i\neq B_j \end{cases}$
Needleman-Wunsch	Dynamic programming algorithm for global sequence alignment.	– DNA sequencing – Pairwise string comparison – Spell correction	$M_{i,j}=Maximum\left[M_{i-1,j-1}+S_{i,j}, M_{i,j-1}+W, M_{i-1,j}+W\right]$ Where: $S_{i,j}=1$ if the two letters at the current position match. $S_{i,j}=-1$ if the two letters at the current position mismatch. W is the indel or the gap score is assumed as –1.
Smith-Waterman	Dynamic programming algorithm for local sequence alignment.	– DNA sequencing – Pairwise string comparison – Biological sequence analysis	$H_{ij}=max\begin{cases} H_{i-1,j-1}+s(a_i,b_j), \\ max_{k\geq 1}\{H_{i-k,j}-W_k\}, \\ max_{l\geq 1}\{H_{i,j-1}-W_l\}, \\ 0 \end{cases}$ $(1\leq i\leq n,\ \ 1\leq j\leq m)$ Where: $H_{i-1,j-1}+s(a_i,b_j)$ represents the score of aligning a_i and b_j. $H_{i-k,j}-w_k$ denotes the score when a_i is positioned at the end of a gap of length k, $H_{i,j-1}-W_l$ denotes the score when b_j is positioned at the end of a gap of length l, 0 indicates no similarity up to a_i and b_j.

Term-based similarity measures. Term-based similarity is string similarity that takes two vectors and computes similarity between them. The metrics used to compute term-based similarity are presented in Table 2. It provides an overview of each term-based algorithm's metric, some properties, and common usage.

Table 2. Comparison of term-based similarity metrics

Algorithm	Properties	Usage	Equation
Euclidean Distance	– Measures the geometric distance between two vectors.	– Document clustering – Text similarity measurement	$D(a,b)=\sqrt{\sum_{i=1}^{n}(a_i-b_i)^2}$
Cosine Similarity	– Measures the cosine of the angle between two vectors.	– Information retrieval – Text classification – Document similarity	$D(a,b)=\cos(\theta)=\frac{a.b}{\|a\|\,\|b\|}$
Jaccard Similarity	– Measures the ratio of intersection to the union of sets.	– Document clustering – Recommendation systems – Text similarity measurement	$D(a,b)=1-\frac{\|a\cap b\|}{\|a\cup b\|}$
Dice's Coefficient	– Measures the ratio of twice the intersection to the sum of set sizes.	– Document clustering – Text similarity measurement	$s=\frac{2\|a\cap b\|}{\|a\|+\|b\|}$

(Continued)

Table 2. Comparison of term-based similarity metrics *(Continued)*

Algorithm	Properties	Usage	Equation
Matching Coefficient	– Measures the ratio of matching attributes to the total number of attributes.	– Data analysis – Similarity measurement in binary data	$SMC = \frac{number\ of\ matching\ attributes}{sum\ of\ all\ attributes}$
Overlap Coefficient	– Measures the ratio of intersection to the minimum of set sizes.	– Information retrieval – Text similarity measurement – Document clustering	$overlap(a,b) = \frac{\lvert a \cap b \rvert}{\min\left(\lvert a \rvert, \lvert b \rvert\right)}$

2.2 Semantic-based similarity

Semantic similarity is a measure of how closely two concepts are related in meaning. It measures the correlation of two concepts based on their basic semantics. It is used in NLP and artificial intelligence (AI) to determine how related one concept is to another.

Corpus-based similarity. A corpus-based similarity is a form of semantic similarity that calculates similarity by utilizing information extracted from extensive corpora. Techniques used to compute corpus-based similarity include:

a) **Hyperspace analogue to language:** Hyperspace analogue to language (HAL) is a semantic model that determines statistical matching between words based on their common occurrences in surrounding text windows [8]. The square co-occurrence matrix of text, where all unique words n is represented as a row and a column. So, we have a matrix of size *nxn*. For every word x, we compute the number of times each second word y occurs next to a. Counting is actually done using weighted co-occurrences. For example, if y occurs near x, it has a weight of 5. If y is separated from x by a word, it has a weight of 4, and so on until a neighbor with a distance of 5 has a weight of 1. This is known as size 5 simple sliding window parsing.

b) **Latent semantic analysis:** Latent semantic analysis is an automated method grounded in mathematical and statistical techniques for uncovering the anticipated relationships between words within textual passages. It distinguishes itself from conventional NLP programs by receiving a collection of texts as input, breaking them down into distinct words defined as unique strings that exhibit discernible patterns, such as sentences or paragraphs.

The initial step in the LSA process involves representing the text as a matrix. In this matrix, each row corresponds to an individual word, while each column represents a portion of text or another contextual element. The cell within this matrix stores the count of times the word in its respective row appears in the specific paragraph or context indicated by its column.

Subsequently, the entries within these cells undergo an initial transformation. During this transformation, the frequency of each cell is weighted using a function that assesses the word's significance within a particular passage and its capacity to convey information within the given discourse domain [9].

c) **Generalized latent semantic analysis:** Generalized latent semantic analysis (GLSA) is a model designed for the computation of semantic concepts and document vectors. In contrast to dimensionality reduction techniques applied to documents, such as LSA, GLSA places its emphasis on the generation of document vectors, which are essentially linear combinations of concept vectors.

This approach extends the conventional LSA methodology, as it shifts the focus towards concept vectors rather than dual-document concept representations. The core requirement for GLSA is the presence of some level of semantic association among terms, coupled with the utilization of a dimensionality reduction method. GLSA methods offer the flexibility to integrate various similarity measures within the concept space in conjunction with an appropriate dimensionality reduction technique [10].

d) **Explicit semantic analysis:** The concept explicit semantic analysis (ESA) has been used to compute the semantic similarity of words or texts. This method has demonstrated high effectiveness across a range of applications, including information retrieval and text classification. The ESA representation of a word comprises a vector. The entries in this vector signify the word's associations with all documents contained within a collection, known as an indexed collection. Texts are represented as centroids of these word vectors, and semantic relationships between texts are evaluated using cosine similarity. Gabrilovich and Markovitch employ Wikipedia articles (concepts) as index documents and employ the standard Tf-Idf weighting scheme for vector entries [11].

Knowledge-based similarity. Knowledge-based similarity is a method of evaluating semantic similarity by utilizing information obtained from a semantic network to assess the degree of similarity between words [12]. WordNet, a comprehensive English language database, is widely recognized as one of the most prominent semantic networks. It categorizes nouns, verbs, adverbs, and adjectives into clusters known as "synsets," which are linked by concept semantics and lexical relations [13]. Table 3 provides a comparison of various semantic similarity metrics, their descriptions, and equations. It can aid in understanding and selecting appropriate techniques for your approach.

Table 3. Comparison of semantic similarity metrics

Metric	Description	Equation
Leacock & Chodorow (LCH)	Semantic similarity based on the length of the shortest path and maximum depth of the taxonomy	$Sim\ lch = -log\frac{length}{2*D}$ Where '*length*' denotes the most direct path connecting two concepts, determined through node-counting, while '*D*' represents the maximum depth of the taxonomy [29].
Wu & Palmer (WUP)	Semantic similarity based on the depth of the Least Common Subsumer (LCS)	$Sim\ wup = \frac{2*depth(LCS)}{depth(concept1)+depth(concept2)}$
Resnik	Semantic similarity based on the information content (IC) of the LCS	$Sim\ res = IC(LCS)$ where *IC* is determined as: $IC(c) = -log\ P(c)$
Lin	Semantic similarity incorporating IC and normalization factor	$Sim\ lin = \frac{2*IC(LCS)}{IC(cncept1)+IC(concept2)}$
Jiang & Conrath (JCN)	Semantic similarity based on information content and depth	$SIM\ jcn = \frac{1}{IC(cncept1)+IC(concept2)-2*IC(LCS)}$

3 RELATED WORKS

To address the issue of cheating in online exams, certain authors have suggested in-exam approaches, such as continuous authentication and real-time online monitoring [14], [15], [16]. Continuous authentication methods, such as biometric scans or keystroke dynamics, aim to verify the student's identity throughout the exam. Real-time online monitoring involves using proctoring software to observe students during the exam to ensure they are not engaging in dishonest behaviors. However, these methods have significant limitations. They can be intrusive, raising privacy concerns among students who may feel uncomfortable with biometric scans and constant observation. They are also resource-intensive, requiring significant investments in human proctors or sophisticated AI systems to monitor multiple students simultaneously, which is costly and challenging for institutions with large student populations. Additionally, these methods are prone to technical issues such as connectivity problems, software malfunctions, or hardware failures, which can disrupt the exam and increase student stress. Moreover, despite these measures, students can still find ways to cheat, such as using hidden devices or receiving help from someone off-camera, making these methods not foolproof and vulnerable to circumvention by determined cheaters. While alternative solutions involve post-examination cheating detection methods, which entail comparing students' responses and identifying similarities [17]. These methods analyze the completed exams to find patterns or similarities that suggest collusion or plagiarism.

In our study, we focus on identifying cheating through the use of post-examination methods that compare students' answers and find similarities. Our proposed approach represents an improvement over current methods in several ways. It offers a more accurate, scalable, objective, and non-intrusive solution to the problem of cheating in online education. By presenting different approaches and techniques refined for similarity measures, we aim to enhance the integrity and fairness of online assessments.

Most traditional similarity computing techniques can be classified into two categories: string-based techniques and semantic-based techniques. The authors in [18] highlight the role of string similarity measures through numerous practical functions, such as data cleaning, data integration, and redundancy detection. Given two sets of strings, the string similarity problem consists in finding all similar pairs in both sets. The existing similarity measures can roughly be categorized into two groups: character-based similarity measures and token-based similarity measures.

Identifying the problem of near-duplicate records in a database represents a crucial stage in the data cleansing and integration process. The authors of [19] point out that most existing methods rely on generic or hand-tuned distance measures to estimate the similarity of potential duplicates, and as string similarity estimates can vary depending on the domain considered, traditional similarity measures may not correctly estimate string similarity. At the token level, some words may be informative when comparing the equivalence of two strings, while others may be ignored.

Textual similarity metrics have been utilized extensively in the field of NLP applications and related domains. One of the initial applications of textual similarity can be traced back to vector models employed in information retrieval. In this context, the identification of the most pertinent documents for a query is achieved through the process of ranking documents within a collection, ordering them in reverse according to their similarity to the given query [20].

According to the findings presented in [21], the conventional approach for measuring similarity between two textual segments relies on a lexical comparison technique. It quantifies the similarity by considering the number of shared lexical elements within the given text fragments. While this method offers some degree of effectiveness, it falls

short in recognizing semantic resemblances between texts. The remedy for this limitation lies in LSA, which constitutes an enhancement over the previously advocated utilization of semantic similarity measures in information retrieval, as highlighted in reference [9]. The authors in [22] introduced an approach that combines semantic content and word order data to assess the similarity between sentences. Initially, semantic similarity is determined by drawing upon lexical knowledge bases and corpus data. Subsequently, the method devised in their study considers the influence of word order on sentence semantics. The similarity metric based on word order assesses the count of distinct words and word pairs in various sequential arrangements.

The authors in [23] proposed a measure called semantic text similarity (STS). This process integrates both syntactic and semantic methodologies to determine the similarity of two texts.

The authors [24] propose an approach that combines corpus-based semantic similarity scores and knowledge-based semantic similarity scores. In machine learning models such as linear regression and clustering models, all the scores are incorporated as features to derive a singular score that quantifies the level of similarity between phrases. This methodology yields notable enhancements in the computation of semantic similarity between sentences. It accomplishes this by merging knowledge-based similarity measures with corpus-based relatedness measures, surpassing the effectiveness of relying solely on corpus-based measures. Subsequently, the subsequent section will provide a more comprehensive overview of the similarity metrics.

The techniques described before encompass various approaches to measuring text similarity, primarily focusing on string-based and semantic-based methodologies. While string-based techniques may overlook semantic resemblances, the limitations predominantly arise from semantic similarity measures. These measures, often reliant on lexical knowledge bases and corpus data, struggle to capture the nuanced meanings and context-dependent variations in text. Traditional semantic measures may also fail to adequately account for syntactic structures and word order, further hindering their effectiveness. Even when combined with corpus-based or knowledge-based scores, semantic similarity estimation remains challenged by the complexity of language and domain-specific semantics. Consequently, these limitations underscore the need for more nuanced approaches to text similarity assessment that can better capture the richness of semantic relationships and contextual nuances.

Given the limitations of traditional semantic similarity measures, new approaches based on techniques such as K-Shingling, Minhashing, and LSH offer promising avenues for improving similarity detection. K-Shingling involves representing documents as sets of shingles (contiguous subsequences of words), which can capture more nuanced textual patterns beyond individual words or characters. Minhashing enables efficient estimation of Jaccard similarity between sets by hashing shingles into shorter signatures, reducing computational complexity while preserving similarity relationships. LSH builds upon Minhashing to efficiently identify similar pairs of documents by grouping similar signatures using hash functions, thereby enabling scalable similarity detection across large datasets. These approaches, by focusing on capturing contextual information and reducing computational overhead, address some of the limitations of traditional semantic similarity measures, offering enhanced accuracy and scalability for similarity detection tasks.

3.1 Current approach in similarity and plagiarism detection

Natural language processing is considered a fundamental method for extracting information from text documents. It refers to the language that people use in

communicating with each other (French, English, etc.). NLP allows computers to process natural language. Usually, the process has several steps during which the import files is processed. In the first stage, terms are extracted from source documents. They are presented in a certain way and thus analyzed using different classification methods. Extracting information from source code is different than extracting information from text documents. Fortunately, there are good tokenization methods for programming languages such as:

a) **K-gram:** A K-gram is a technique used in NLP to analyze sequences of text. This method is used to extract a sequence of k contiguous characters from a text [25].
b) **K-Shingling:** K-Shingling is an algorithm for detecting document clones based on tokens. It is a way to convert documents into tokenized collections that can perform sentence similarity comparisons. First, documents are divided into shingles of size k. A shingle is a contiguous subsequence of characters or words in a document [26].
c) **Winnowing:** Winnowing is an algorithm for processing document fingerprints. The winnowing algorithm computes the hash value for each k-gram, identifies the hash values, and employs a hash rolling function. These hash values constitute the window. Within each window, the algorithm selects the minimum hash value, and in case of multiple hashes having the smallest value, it chooses the rightmost hash. These selected hashes collectively form the document's fingerprint, utilized for similarity detection between input texts [25].
d) **Minhashing:** MinHash is a method developed by Broder [27] to estimate the similarity between sentences that are presented as shingles. It calculates the Jaccard similarity between shingles, which is a widely used and effective similarity measure. This calculation is inefficient when the amount is large. Instead, MinHash compute a signature matrix for sets, which relies on the computation of K Minhash values, which maps elements of an arbitrary set S to different integer values using a different hash function [27], [28].
e) **LSH:** The general theory of the LSH approach is that elements must be hashed against each other multiple times so that identical elements that are very similar to each other will hash into a common bucket. MinHash signatures of sentences allow the development of efficient LSH methods to index sentences such that sentence pairs exceeding a certain similarity threshold are likely to be stored in similarity buckets [29].

An example of a combination of previous methods is presented in Figure 1.

Fig. 1. A combination of K-shingling, Minhashing, and LSH methods

4 PROPOSED METHODOLOGY

The objective of our study is to address the following question given multiple exam sheets; our objective is how to detect the near-duplicate pairs to effectively identify potential instances of cheating.

In this work we aim to construct and optimize string similarity measures by providing efficient ways to compare large sets of documents or answers. We used techniques to compare and match strings such as shingling, Minhashing, and LSH.

Figure 2 depicts our proposed approach for documents similarity detection in online exams.

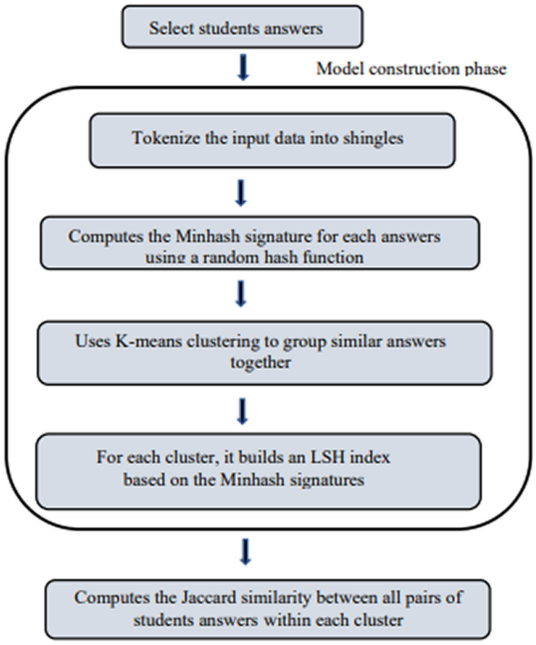

Fig. 2. The proposed approach for documents similarity detection

Related to our research topic, our main objective is to apply these techniques to detect cheating by clustering students answers and compute the value of answers similarity in each cluster. For that purpose, we tried to improve the K-Shingling-Minhashing-LSH approach by incorporating K-means clustering. The K-means step helps to group similar answers together, which can improve the effectiveness of LSH by reducing the number of false positives. The representative signatures computed for each cluster also help to reduce the size of the signatures used in LSH, which can improve the efficiency of this approach. In this study we are focusing on the type of examination that requires writing a paragraph that must differ from student to another. Figure 2 depicts our proposed approach for document similarity detection. A schema of the proposed approach is presented in Figure 3.

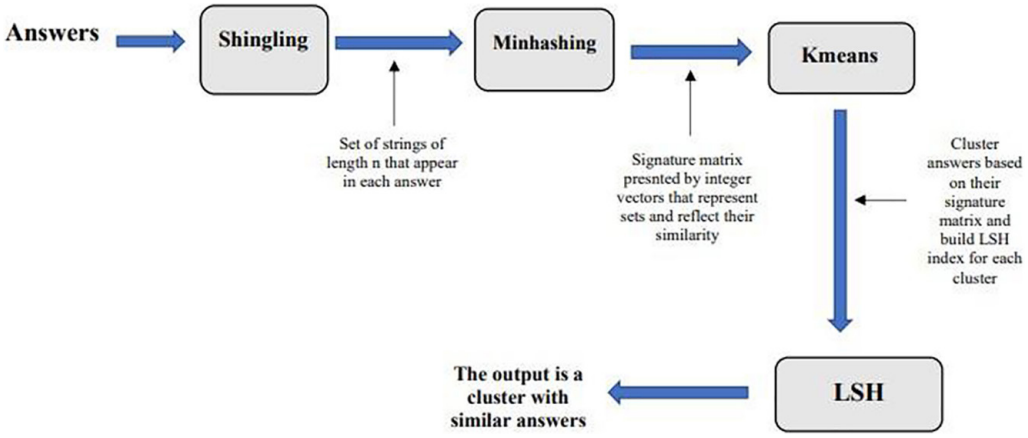

Fig. 3. A schema of proposed approach

Before delving into the implementation details, let us first provide a high-level overview of the algorithm's structure and logic.

4.1 Algorithm

Algorithm 1: The Proposed Implementation Algorithm

```
Begin
  n = length of shingles
  Generate random vectors a and b
  For each answer Ai:
    Extract shingles of length n from Ai
    Compute hash value for each shingle using (a * hash(shingle) + b) % max_shingle_id
    Generate Minhash signature for the shingles using a random hash function
  Set number_of_clusters and LSH_threshold
  Cluster answers using K-means based on their signatures
  Initialize empty LSH index for each cluster
  For each cluster:
    Compute representative signature by taking centroid of all signatures in the cluster
    Use LSH to hash the representative signatures into buckets
  For each cluster:
    Query LSH index to retrieve candidate pairs of answers
    For each candidate pair of answers:
      Compute Jaccard similarity between the pair
      Print the Jaccard similarity
End
```

An overview of the process is presented in Figure 4.

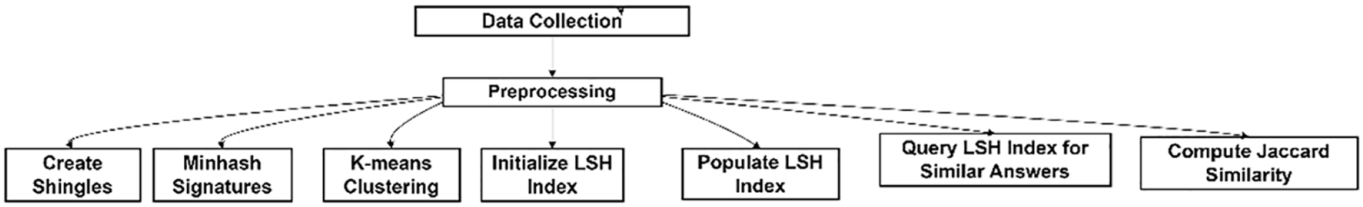

Fig. 4. Overview of the process

First, we tokenize the input data into shingles, which are contiguous sequences of words. It then computes the Minhash signature for each answer using a random hash function and uses K-means clustering to group similar answers together. Finally, for each cluster, it builds an LSH index based on the Minhash signatures of the answers in that cluster and uses this index to efficiently query for similar answers given a query answer.

The create shingles function takes in a text string and an n parameter and returns a set of shingles of length n extracted from the text. This function first tokenizes the text into words and then extracts all consecutive sequences of n characters from each word.

The Minhash function takes a set of shingles and an n_hashes parameter and computes the Minhash signature of the shingles using a random hash function. This function first generates two random vectors, a and b, and then computes the hash value for each shingle using the formula (a * hash(shingle) + b) % max_shingle_id, where max_shingle_id is a large prime number. It then keeps track of the minimum

hash value for each of the n_hashes random vectors and returns these minimum values as the Minhash signature.

The LSH function takes in a list of Minhash signatures and a threshold parameter and returns an LSH index built on these signatures. This function first initializes an empty LSH index with the given threshold and then inserts each Minhash signature into the index with a unique key. The query function takes an LSH index and a Minhash signature and queries the index to find similar answers to the one represented by the Minhash signature. This function first calls the query method of the LSH index with the Minhash signature, which returns a set of keys of the answers that have a Jaccard similarity greater than the threshold of the LSH index. It then converts these keys back to integers and returns them as the list of similar answers. Then we initialize two variables, the number of clusters and the LSH threshold. It then tokenizes the input data into shingles and computes the Minhash signature for each answer. It then clusters the answers using K-means and initializes an empty LSH index for each cluster. It populates these indexes by inserting the Minhash signatures of the answers that belong to each cluster. We query each LSH index for similar answers to a given query answer, which is also tokenized into shingles and converted to a Minhash signature using the same process as before. For each cluster, it prints the list of similar documents returned by the LSH query.

Finally, we compute the Jaccard similarity between all pairs of answers within each cluster and prints the results.

5 RESULTS AND DISCUSSION

We suppose we have six students who take the exam; the question of the examination is what the definition of NLP is the answers of students are as follows:

Table 4. Students' answers

Doc_Index	Students	Answers
0	Student 0	Natural language processing refers to the branch of computer science—and more specifically, the branch of artificial intelligence or AI—concerned with giving computers the ability to understand text and spoken words in much the same way human beings can.'
1	Student 1	"Natural language processing represents a field within computer science to enable computers to comprehend both written text and spoken language in a manner similar to human understanding."
2	Student 2	"Natural language processing is a subfield of artificial intelligence dedicated to empowering computers with the capability to comprehend human speech."
3	Student 3	"Natural language processing is a field within computer science that aims to enable computers to understand written text and spoken language in a way that is similar to human comprehension."
4	Student 4	"Natural language processing represents a field within computer science to enable computers to comprehend both written text and spoken language in a manner similar to human understanding."
5	Student 5	"Natural language processing represents a field within computer science to enable computers to comprehend both written text and spoken language in a manner similar to human understanding."

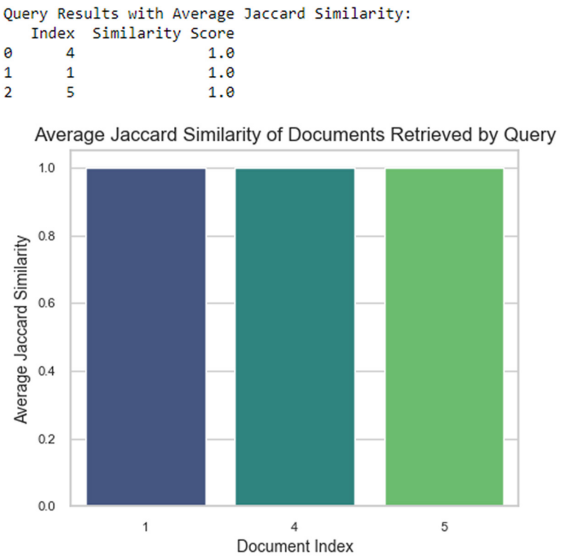

Fig. 5. Average Jaccard similarity of documents retrieved by query

Our goal is to detect the students who have the same answer **as student 1,** after the implementation of our approach with **Python**, we choose a threshold = 0.9, the output was a cluster: [4, 5, 1], where all answers similar to the query are gathered.

As a result, we have **Student 1**, **Student 4**, and **Student 5** having identical answers.

The average Jaccard similarity within the cluster is indicative of how closely related the documents are within a given cluster. A high average Jaccard similarity suggests that the clustering has accurately grouped similar documents together. Figure 5 illustrates the average Jaccard similarity of documents retrieved by query.

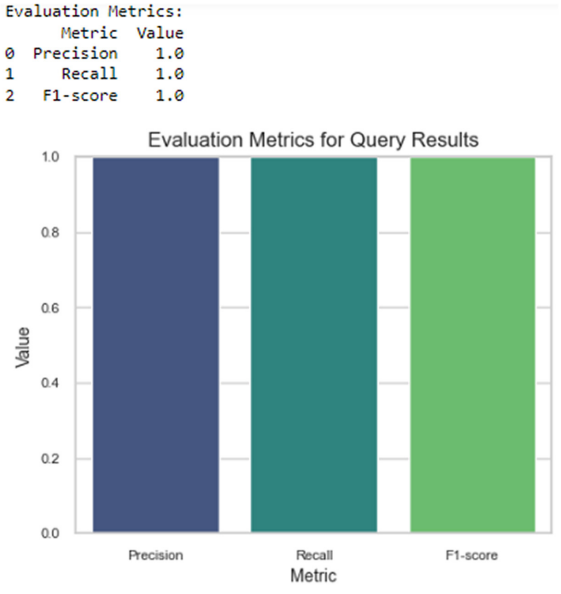

Fig. 6. Evaluation metrics for query results

The bar plot in Figure 6 displays three key evaluation metrics: Precision, Recall, and F1-Score, each represented by a bar with values of 1.0. Precision is the ratio of true positives to the sum of true positives and false positives, indicating that all

retrieved documents were relevant. Recall is the ratio of true positives to the sum of true positives and false negatives, showing that all relevant documents were retrieved. The F1-Score, the harmonic mean of Precision and Recall, reflects perfect performance. The bar plot in Figure 6 illustrates that our approach achieves perfect precision, recall, and F1-Score, indicating ideal performance in identifying similar documents. This result is expected due to the mathematical nature of our approach, which provides exact similarity calculations. The evaluation metrics are used to verify and showcase the effectiveness of your method, reinforcing confidence in its accuracy and reliability.

Our approach leverages mathematical methods such as K-Shingling, Minhashing, and LSH, known for their deterministic nature, providing consistent and correct outputs, unlike many machine learning models that can produce variable results based on training data and model parameters. K-Shingling involves breaking text into overlapping substrings (shingles) of length k, ensuring a robust and reproducible similarity measure based on exact substring matches. Min-hashing generates a compact representation of sets of shingles, allowing for efficient and accurate computation of the Jaccard similarity between sets without relying on probabilistic models. LSH efficiently retrieves similar items by hashing inputs so that similar items are mapped to the same buckets with high probability, ensuring an efficient and reliable process that reduces computational overhead while maintaining accuracy.

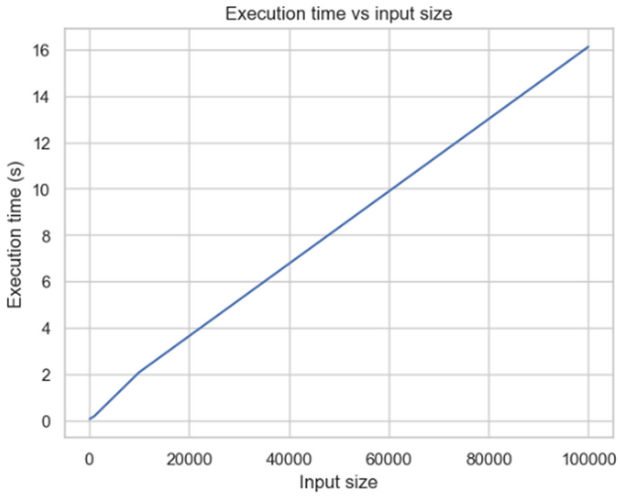

Fig. 7. Execution time for creating shingles of varying input sizes

We have also measured the execution time for creating shingles of varying input sizes. The time is increasing when the size of input increases too. The conclusion based on Figure 7 is that the execution time for creating shingles increases as the input size increases. This suggests a positive relationship between input size and execution time, which is typical for many text processing algorithms. In other words, the longer the text being processed, the more time it takes to create the shingles.

The novelty of our work lies in its application context, targeting educational assessments rather than traditional document similarity detection focused on large-scale web data. Our use of a similarity threshold for clustering identical answers provides a clear and intuitive way to group similar responses, easily interpretable by educators. By analyzing execution time concerning input size, we offer insights into the scalability of shingling methods for varying text lengths, valuable for optimizing

text processing workflows. Additionally, our approach emphasizes the reliability of mathematical methods such as K-Shingling, Minhashing, and LSH, ensuring accurate and reproducible results, in contrast to the variability often seen in machine learning models. Comparing our approach with related work is essential to highlight the novelty and improvements. In typical LSH-based document similarity tasks, the main challenge is to balance the trade-off between accuracy and computational efficiency. Our approach, which combines MinHash LSH with K-means clustering, aims to improve the clustering quality and retrieval accuracy. Related works often focus solely on LSH or K-means separately, which may lead to suboptimal results in terms of clustering similar documents. By integrating these methods, we leverage the strengths of both: LSH's ability to efficiently approximate Jaccard similarity and K-means' capability to cluster similar feature vectors.

One limitation of our approach is that it relies on the choice of parameters, such as the number of permutations in MinHash and the number of clusters in K-means. Future work could explore adaptive methods for selecting these parameters. Additionally, our evaluation is based on a small dataset; larger and more diverse datasets could provide more insights into the scalability and robustness of our method.

6 CONCLUSION

As online learning continues to grow in popularity, the need for effective measures to detect answer similarity becomes increasingly important. In this study, we have enhanced the K-Shingling-Minhashing-LSH approach by incorporating K-means clustering. Overall, the implementation demonstrates how to use K-Shnigling, MinHashing, and LSH with K-means to efficiently find similar answers in a large dataset.

However, while the LSH technique plays a significant role in identifying candidate pairs, it is important to acknowledge that the Jaccard similarity used in our study may not always be the optimal choice for comparing document similarity. Jaccard similarity does not take into account the frequency of shingles in each document, which may lead to limitations in certain scenarios.

To address this limitation and enhance the robustness of our approach, future work could explore alternative similarity measures such as cosine similarity or Euclidean distance.

7 REFERENCES

[1] P. Resnik, "Semantic similarity in a taxonomy: An Information-based measure and its application to problems of ambiguity in natural language," *Journal of Artificial Intelligence Research*, vol. 11, pp. 95–130, 1999. https://doi.org/10.1613/jair.514

[2] W. W. Cohen, P. Ravikumar, and S. E. Fienberg, "A comparison of string distance metrics for name-matching tasks," *IIWeb*, vol. 3, pp. 73–78, 2003.

[3] T. K. Landauer, P. W. Foltz, and D. Laham, "Introduction to latent semantic analysis," *Discourse Process*, vol. 25, nos. 2–3, pp. 259–284, 1998. https://doi.org/10.1080/01638539809545028

[4] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," in *Proceedings of the 1st International Conference on Learning Representations (ICLR 2013)*, Workshop Track, Scottsdale, Arizona, USA, May 2–4, 2013.

[5] A. Budanitsky and G. Hirst, "Evaluating WordNet-based measures of lexical semantic relatedness," *Comput. Linguist.*, vol. 32, no. 1, pp. 13–47, 2006. https://doi.org/10.1162/089120106776173093

[6] S. A. Bernhard and S. Sen, "Detecting collusion in online exams," *J. Comput. Sci. Coll.*, vol. 196, pp. 303–310, 2004.

[7] D. L. McCabe, K. D. Butterfield, and L. K. Treviño, "Academic dishonesty in graduate business programs: Prevalence, causes, and proposed action," *Academy of Management Learning & Education (AMLE)*, vol. 5, no. 3, pp. 294–305, 2006. https://doi.org/10.5465/amle.2006.22697018

[8] J. Vesanto and E. Alhoniemi, "Clustering of the self-organizing map," *IEEE Trans. Neural Netw.*, vol. 11, no. 3, pp. 586–600, 2000. https://doi.org/10.1109/72.846731

[9] T. K. Landauer, P. Foltz, and D. Laham, "Introduction to latent semantic analysis," *Discourse Processes*, vol. 25, nos. 2–3, pp. 259–284, 1998. https://doi.org/10.1080/01638539809545028

[10] I. Matveeva, G. A. Levow, A. Farahat, and C. Royer, "Term representation with generalized latent semantic analysis," in *Curr. Issues Linguist. Theory,* N. Nicolov, K. Bontcheva, G. Angelova, and R. Mitkov, Eds., Amsterdam: John Benjamins Publishing Company, vol. 292, 2007, pp. 45–54. https://doi.org/10.1075/cilt.292.08mat

[11] H. H. Vu, J. Villaneau, and P. F. Marteau, "Sentence similarity by combining explicit semantic analysis and overlapping N-grams," in *Text, Speech and Dialogue, TSD 2014*, in Lecture Notes in Computer Science, P. Sojka, A. Horák, I. Kopeček, and K. Pala, Eds., vol. 8655, 2014, pp. 201–208. https://doi.org/10.1007/978-3-319-10816-2_25

[12] R. Mihalcea, C. Corley, and C. Strapparava, "Corpus-based and knowledge-based measures of text semantic similarity," in *Proceedings of the 21st National Conference on Artificial Intelligence (AAAI '06)*, AAAI Press, vol. 1, 2006, pp. 775–780.

[13] G. A. Miller, R. Beckwith, C. Fellbaum, D. Gross, and K. J. Miller, "Introduction to WordNet: An on-line lexical database," *Int. J. Lexicogr.*, vol. 3, no. 4, pp. 235–244, 1990. https://doi.org/10.1093/ijl/3.4.235

[14] R. Bawarith, A. Basuhail, A. Fattouh, and S. Gamalel-Din, "E-exam cheating detection system," *International Journal of Advanced Computer Science and Applications (IJACSA)*, vol. 8, no. 4, 2017. https://doi.org/10.14569/IJACSA.2017.080425

[15] M. Ghizlane, B. Hicham, and F. H. Reda, "A new model of automatic and continuous online exam monitoring," *IEEE Xplore*., 2020.

[16] Y. Atoum, L. Chen, A. X. Liu, S. D. Hsu, and X. Liu, "Automated online exam proctoring," *IEEE Transaction on Multimedia*, vol. 19, no. 7, pp. 1609–1624, 2017.

[17] Ali M. Duhaim, Safaa O. Al-Mamory, and Mohammed Salih Mahdi, "Cheating detection in online exams during Covid-19 pandemic using data mining techniques," *Webology*, vol. 19, no. 1, pp. 341–366, 2022. https://doi.org/10.14704/WEB/V19I1/WEB19026

[18] Y. Jiang, G. Li, J. Feng, and W.-S. Li, "String similarity joins: An experimental evaluation," in *Proc. VLDB Endow.*, vol. 7, 2014, no. 8, pp. 625–636. https://doi.org/10.14778/2732296.2732299

[19] F. Friendly, "Jaro–Winkler distance improvement for approximate string search using indexing data for multiuser application," *J. Phys. Conf. Ser.*, vol. 1361, pp. 1–7, 2019. https://doi.org/10.1088/1742-6596/1361/1/012080

[20] W. H. Gomaa and A. A. Fahmy, "Survey of text similarity approaches," *Int. J. Comput. Appl.*, vol. 68, no. 13, pp. 13–18, 2013. https://doi.org/10.5120/11638-7118

[21] Salton and M. E. Lesk, *Computer Evaluation of Indexing and Text Processing*, Prentice Hall, Ing. Englewood Cliffs, New Jersey, 1971, pp. 143–180.

[22] Y. Li, D. McLean, Z. Bandar, J. O'Shea, and K. Crockett, "Sentence similarity based on semantic nets and corpus statistics," *IEEE Transactions on Knowledge and Data Engineering*, vol. 18, no. 8, pp. 1138–1150, 2006. https://doi.org/10.1109/TKDE.2006.130

[23] A. Islam and D. Inkpen, "Semantic text similarity using corpus-based word similarity and string similarity," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 2, no. 2, pp. 1–25, 2008. https://doi.org/10.1145/1376815.1376819

[24] A. Nitish, A. Kartik, and B. Paul, "DERI&UPM: Pushing corpus based relatedness to similarity: Shared task system description," in *First Joint Conference on Lexical and Computational Semantics (*SEM)*, Association for Computational Linguistics, Montreal, Canada, 2012, pp. 643–647.

[25] S. Schleimer, D. S. Wilkerson, and A. Aiken, "Winnowing: Local algorithms for document fingerprinting," in *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data*, 2003, pp. 76–85. https://doi.org/10.1145/872757.872770

[26] S. Niwattanakul, J. Singthongchai, E. Naenudorn, and S. Wanapu, "Using of jaccard coefficient for keywords similarity," in *Proceedings of the International MultiConference of Engineers and Computer Scientists*, March 13–15, Hong Kong, vol. I, 2013, pp. 380–384.

[27] A. Z. Broder, "On the resemblance and containment of documents," *Compression Complex Seq*, pp. 21–29, 1997.

[28] A. Z. Broder, S. C. Glassman, M. S. Manasse, and G. Zweig, "Syntactic clustering of the web," *Comput. Netw. ISDN Syst.*, vol. 29, nos. 8–13, pp. 1157–1166, 1997. https://doi.org/10.1016/S0169-7552(97)00031-7

[29] S. Ranshous, M. Chaudhary, and N. F. Samatova, "Efficient outlier detection in hyperedge streams using minhash and locality-sensitive hashing," in *Complex Netw. Their Appl. VI*, *COMPLEX NETWORKS 2017, Studies in Computational Intelligence*, C. Cherifi, H. Cherifi, M. Karsai, and M. Musolesi, Eds., Springer, Cham, vol. 689, 2018, pp. 105–116. https://doi.org/10.1007/978-3-319-72150-7_9

8 AUTHORS

Nabila El Rhezzali is with the ITQAN Team, LyRica Lab, Information Sciences School, Rabat, Morocco (E-mail: nabila.el-rhezzali@esi.ac.ma).

Imane Hilal is with the ITQAN Team, LyRica Lab, Information Sciences School, Rabat, Morocco.

Meriem Hnida is with the ITQAN Team, LyRica Lab, Information Sciences School, Rabat, Morocco.