



Urdu paraphrased text reuse and plagiarism detection using pre-trained large language models and deep hybrid neural networks

Hafiz Rizwan Iqbal¹ · Muhammad Sharjeel² · Jawad Shafi² · Usama Mehmood¹ · Saeed Ul Hassan¹ · Agha Ali Raza³

Received: 26 September 2023 / Revised: 11 April 2025 / Accepted: 19 April 2025

© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2025

Abstract

The growing prevalence of text reuse and plagiarism in various fields has led to an urgent need for reliable computational methods for detection. However, current commercial plagiarism detection systems are ineffective in identifying paraphrased cases of text reuse, highlighting the need for improvement. Previous research on paraphrased text reuse and plagiarism detection has mainly focused on English, European, Persian, and Arabic languages, and very few studies have been reported on the under-resourced Urdu language. This study aims to overcome this research gap by using a Deep Neural Network (DNN) based architecture and pre-trained Large Language Models (LLMs) for the task of Urdu paraphrased text reuse and plagiarism detection. The architecture called Deep Text Reuse and Paraphrased Plagiarism Detection (D-TRaPPD), relies on LLMs for input and utilizes CNN and LSTM to extract essential textual features. Moreover, we have proposed and evaluated two D-TRaPPD variants, Word Embeddings-D-TRaPPD (WE-D-TRaPPD) and Sentence Embeddings-D-TRaPPD (SE-D-TRaPPD), using two gold standard document-level corpora containing both real and simulated cases of Urdu paraphrased text reuse and plagiarism. The results demonstrate the effectiveness of the D-TRaPPD architecture, with SE-D-TRaPPD achieving the highest F_1 scores of 91.77 for real cases and 95.15 for simulated cases. Furthermore, the results highlight the superiority of our approaches over the state-of-the-art methods for Urdu paraphrased text reuse and plagiarism detection.

Keywords Text reuse · Plagiarism detection · Deep neural networks · Pre-trained large language models (LLMs) · Natural language processing

1 Introduction

Text reuse refers to the practice of borrowing from pre-existing text(s) (source text) to create new text(s) (reused text). Text plagiarism is a form of text reuse, that involves using another person's text without giving proper credit or attribution. The reused text (or plagiarized text) can be a simple copy-paste from the source or paraphrased using various text rewriting methods, e.g., synonym replacement, reordering, expansion/compression, change of word forms, etc. [1]. The unprecedented growth of online text repositories in the recent past and

Extended author information available on the last page of the article

freely available text rewriting tools have made it much simpler to reuse (paraphrase) a piece of text. As a consequence, it has become common practice not only in academia but journalism, media, and even politics [2–5].

With the rising volume of text reuse and plagiarism cases, the interest in developing computational methods for their detection is becoming a hot research topic [6]. A recent survey concluded that the available commercial plagiarism detection systems performed poorly on paraphrased cases of text reuse [5, 7]. Moreover, several experiments carried out to evaluate the performance of plagiarism detection systems clearly show that it is straightforward to identify exact copies but not the highly obfuscated ones [5, 7–11]. These surveys highlight that paraphrased cases of text reuse are hard to detect and it is the need of the hour to develop reliable systems that can efficiently detect paraphrased text reuse and plagiarism. Moreover, such systems have various potential applications in related fields such as question-answering [12], natural language generation [13, 14], and intelligent tutoring systems [15].

Most of the previous research on paraphrased text reuse and plagiarism detection has mainly focused on English, European, Persian, and Arabic languages [5, 6, 16–18] and very few efforts have been reported for the under-resourced Urdu language [19–21]. Urdu belongs to the Indo-Aryan family, majorly spoken in Pakistan and India. It is written right-to-left and has around 500 million speakers around the world [22]. However, in terms of text reuse and plagiarism detection research, it is still in its infancy. Existing studies have focused on developing standard evaluation corpora, performed experiments by using simple surface-level approaches and basic Machine Learning (ML) classifiers [19–21].

Recently, Iqbal et al. [23] has initiated the exploration of DNNs for detecting Urdu paraphrased text reuse and plagiarism. In their initial attempt, they introduced a foundational hybrid deep neural architecture called D-TRaPPD. They demonstrated the effectiveness of D-TRaPPD in detecting Urdu paraphrased text reuse and plagiarism by creating a prototype based on a traditional word embedding model, i.e., FastText [24], and achieved notable improvements over existing approaches. However, this study has left some unexplored areas. It did not investigate contextual word embedding models, e.g., ELMO [25] or the latest mono- and multi-lingual transformers-based Large Language Models (LLMs), e.g., BERT [26], RoBERTa [27], XLM [28] etc., for the task of detecting Urdu paraphrased text reuse and plagiarism. Furthermore, it overlooks the examination of real cases of Urdu paraphrased text reuse and plagiarism at the document level, which presents a valuable avenue for further exploration.

LLMs like GPT [29] and BERT [26] have revolutionized the field of paraphrased text reuse and plagiarism detection by leveraging transformer architectures to effectively capture nuanced semantic relationships [30, 31]. These models are rigorously trained on diverse corpora to produce high-quality contextual embeddings that capture underlying semantic meaning beyond mere syntactic similarities [32, 33]. These capabilities have made LLMs indispensable for developing state-of-the-art text reuse and paraphrase detection systems [34, 35].

Considering Urdu, which is a low-resource language, creating and deploying LLMs for Urdu presents several significant challenges. These include the scarcity of high-quality annotated corpora, the high cost associated with training LLMs, insufficient linguistic resources, and under-representation in NLP research [36, 37]. Given these constraints, it becomes imperative to explore multilingual LLMs as a viable alternative. A multilingual LLM, trained on a combination of linguistically related languages can leverage shared vocabulary and structural similarities to address these challenges effectively [38–40].

In response to the limitations of traditional approaches (i.e., surface-level and machine learning-based), advancements in DNN-based approaches, and the emergence of language

specific and language agnostic LLMs, in this study, we extend Iqbal et al.'s [23] work by introducing significant modifications to the D-TRaPPD architecture (Section 3). The architecture uses the capabilities of both Convolutional neural networks (CNNs) and Long Short Term Memory (LSTM) to extract salient features from the text. When a paraphrased text pair is fed into the D-TRaPPD, it is transformed into embedding matrices. These matrices are subsequently utilized by the CNN module to figure out textual attributes like word positions and their sequential orderings. As our first major contribution, building upon these input embeddings, we have introduced two distinct variants of the D-TRaPPD architecture (Section 3): (i) Word Embeddings-D-TRaPPD (WE-D-TRaPPD) and (ii) Sentence Embeddings-D-TRaPPD (SE-D-TRaPPD). Moreover, our second contribution lies in exploiting the capabilities of a contextual word embedding model (i.e., ELMO) and harnessing the power of state-of-the-art transformers-based mono- and multi-lingual LLMs to identify paraphrased instances of text reuse and plagiarism within the Urdu language. Furthermore, as a third contribution, we have conducted comprehensive evaluations of both variants (i.e., WE-D-TRaPPD and SE-D-TRaPPD) using two esteemed document-level Urdu text reuse and plagiarism corpora, namely COUNTER [19] and UPPC [41], encompassing real and simulated instances of Urdu paraphrased text reuse and plagiarism.

We believe that the introduction of novel D-TRaPPD architecture variants and the comprehensive evaluations conducted represent a significant advancement in the field of Urdu NLP, particularly in the area of paraphrased text reuse and plagiarism detection. This study holds the potential to, (i) facilitate the examination and development of effective paraphrase detection systems, particularly tailored for the Urdu language, (ii) offer a comprehensive assessment of DNN-based methodologies across a spectrum of tasks and datasets, and (iii) inspire further exploration within the realm of Urdu paraphrasing, text reuse, and plagiarism detection. While our primary focus has been on the application of the D-TRaPPD architecture to the Urdu language, we believe its adaptability for other languages with minimal fine-tuning holds promise.

The rest of this paper is structured as follows: Section 2 provides an overview of the related research. Section 3 delves into the details of the proposed variants of D-TRaPPD architecture, while Section 4 outlines the experimental setup. Section 5 presents and discusses the results, and finally, Section 6 concludes the paper while also pointing out potential directions for future research.

2 Literature review

Over the years, many different monolingual text reuse and plagiarism detection approaches have been proposed [5, 7, 21, 42, 43]. These approaches can be broadly categorized into the following three types: (i) surface-level, (ii) semantics-level, and (iii) idea-level approaches. However, our research concentrates solely on semantic-level approaches, as surface-level approaches have already been extensively explored and have established themselves as the cutting-edge for monolingual text reuse and plagiarism detection [9, 20, 21, 44]. Whereas, the exploitation of the idea-level approaches is yet an open challenge and beyond the scope of this research study.

2.1 Surface-level text reuse and plagiarism detection approaches

The task of detecting Urdu text reuse and plagiarism has traditionally been addressed by computing surface-level similarity between source and reused texts. The approaches used

could be categorized as (i) content-based, (ii) sequence alignment-based, and (iii) style-based. Table 1 presents a summarized view of these traditional approaches applied on two benchmark corpora, i.e., COUNTER (Corpus of Urdu News Text Reuse) [19] and USTRC (Urdu Short Text Reuse Corpus) [45]. For COUNTER, sequence alignment-based approaches have established the state-of-the-art ($F_1 = 0.81$) at document-level. For short texts in USTRC, content-based (i.e., CNG - Character N-Gram) outperformed ($F_1 = 0.775$) all the other approaches.

It can be concluded from the above discussion and the data presented in Table 1 that the traditional approaches used in detecting Urdu text reuse and plagiarism work only at the surface-level. They do not consider the fact that different words can represent the same concepts or that exact words convey different meanings in different contexts. Moreover, these approaches compute similarity by capturing only the lexical characteristics of the text, thereby inherently neglecting the semantic and syntactic properties of the text.

2.2 Semantics-based text reuse and plagiarism detection approaches

To overcome the limitations of surface-level approaches (Section 2.1), various semantics-level approaches have been reported in the literature [7, 17]. These can be categorized as (i) knowledge-based, (ii) corpus-based, (iii) ML-based, and (iv) DNN-based. Since this research focuses on Urdu paraphrased text reuse and plagiarism detection using pre-trained LLMs, we will limit our discussion to DNN-based approaches.

DNN-based semantic-level approaches can be further categorized into three types: (i) CNN-based, (ii) LSTM-based, and (iii) Hybrid. However, a general trend has been observed in the literature that hybrid approaches achieve better results than stand-alone approaches for semantics-based plagiarism detection [46–49].

Hybrid models that combine variants of CNNs and LSTMs to capture both local and global text features, typically yield superior results in paraphrased detection. For instance, Agarwal et.al [50] proposed a hybrid deep learning model combining CNN and RNN for paraphrase detection. The model incorporates a word-pair similarity module to capture fine-grained semantic relationships. Sentence modeling leverages CNN for local features and RNN for long-term dependencies, while the similarity module analyzes word-pair interactions using CNN. Evaluation on Twitter and MSRP corpora demonstrates the model's effectiveness in identifying paraphrases, achieving F1-scores of 0.742 and 0.815, respectively. Hambi et.al [51] introduced a plagiarism detection framework for English texts, incorporating Doc2Vec, Siamese-LSTM, and CNN models to build preprocessing, learning, and detection layers, respectively. Comparative analysis with existing tools highlights the proposed approach's effectiveness in detecting various plagiarism types, achieving a notable accuracy of 98.33%. Mahmoud et.al [52] investigated the efficacy of BERT-based models for Arabic paraphrase detection. The proposed framework leverages AraBERT to extract contextualized embeddings, followed by a Siamese LSTM architecture for modeling sentence pairs. The model is trained and evaluated on OSAC and SemEval corpora, as well as a custom-built paraphrased corpus. Experimental results demonstrate the effectiveness of the proposed approach, achieving F1-scores of 87.31% and 83.97% on OSAC and SemEval, respectively. Hamza et al [53] pioneered the use of ELMo embeddings to capture semantic and syntactic word relationships for Arabic question classification. Their exploration of various deep learning architectures, including CNNs, RNNs, and hybrid models, provided valuable insights into the task. However, their reliance on concatenation for feature fusion and the use of a softmax classifier might limit the model's ability to capture complex interactions between features.

Table 1 Overview of the surface-level similarity-based approaches explored for Urdu paraphrased text reuse and plagiarism detection

Classification	Method	Advantages	Limitations
Content-based	LCS	Identify modifications by altering operations (insertion/del/sub)	Unable to identify block move Only explored for short Urdu texts
	GST	Able to identify block move Achieved good results	Follows greedy technique in selection and matching of tokens Do not address semantic changes Poorly performed for both Document/Sentence
	VSM	Renowned IR approach	level plagiarism detection tasks Sensitive to semantics
	WNG	Simple approach	Lacks in capturing long-term dependencies
	SWNG	Good for document level plagiarism detection	Unable to handle unseen instances
		Good in capturing plagiarism by synonym replacement	Only explored for Document level PD Poorly performed for both binary and ternary classifications Works best only where synonym replacement were used to generate suspicious document.
CNG	Outperformed for both binary and ternary classification tasks	Only explored for Sentence level PD Leads to failure due to vocabulary gap	
	Sequence Alignment -based	GA	Finds optimal alignment between two sequences Use dynamic programming
LA		Able to handle divergent length sequences Use dynamic programming	Only explored for Sentence level PD Poorly performed among all other surface level approaches
Style-based	TTR	Originally designed for authorship attribution, and good in capturing writing style	Not suitable for both Document and Sentence level PD Not perform well when sequences are of different lengths
	TR		
	SR		

Transformers, which are the latest cutting-edge models in the field of language and vision tasks have improved state-of-the-art [26, 29]. Specifically, in the field of NLP, transformers have revolutionized paraphrase detection by incorporating attention along with context in word embeddings. Transformers use an attention mechanism to determine which parts of the input sequence are crucial at each step. Generative Pre-Trained Transformers (GPT, GPT-2, GPT-3) [29] and BERT [26] are two well-known pre-trained LLMs based on transformers. OpenAI's GPT trains the language model unsupervised on a much larger collection of textual data using a similar idea as ELMO [25]. However, GPT and ELMO differ in two key aspects. First, they have different architectures. ELMO trains two separate LSTMs (left-to-right and right-to-left) and combines their representations with shallow concatenation. On the other hand, GPT, built upon the renowned multi-layer transformers [54], predicts future tokens solely in one direction, i.e., from left to right. Second, GPT and ELMO use contextualized embeddings differently. GPT has been extensively evaluated on various NLP tasks, including semantic similarity and paraphrase detection [29].

BERT [26] is a state-of-the-art large language model, similar to GPT, that is trained on large amounts of raw text and fine-tuned on specific tasks. Unlike GPT, BERT's architecture uses a bidirectional Transformer encoder, which allows it to consider both left-to-right and right-to-left contexts during training. BERT is trained using two tasks: (i) Masked Language Model (MLM), where 15% of the tokens in a sequence is randomly masked and the model is tasked with predicting the missing words, and (ii) Next Sentence Prediction (NSP), a binary classification task aimed at determining whether a sentence follows another sentence. BERT has been assessed on multiple NLP tasks, including paraphrase detection, and has demonstrated that considering the context surrounding a word, rather than only after it, is more effective in capturing the word's semantic and syntactic characteristics.

Addressing the limitations of monolingual sentence embedding models, the demand for multilingual or language-agnostic sentence embedding models emerges. Language Agnostic Sentence Representations (LASER) [38] is a pioneering effort, offering versatile multilingual vector representations for sentences across 93 languages. Another contender, Multilingual Universal Sentence Encoder (mUSE) [39], is a pre-trained multilingual and multitask sentence embedding model, leveraging translation-based bridge tasks and two prominent transformers [54], complemented by CNN-based multilingual models. Multilingual BERT (mBERT) adopts the BERT architecture and training methodology [26], trained on concatenated raw text from Wikipedia's 104 languages, without explicit cross-lingual signals alignment. Language-agnostic BERT (LaBERT) [40], a derivative of mBERT [26], generates a shared embedding space for 109 languages, using a unique training approach that merges two pre-trained encoders: MLM and Translation Language Model (TLM). Cross-lingual Language Model (XLM-R), with 'R' representing Robustly Optimized BERT pre-training Approach (RoBERTa) [55], is a transformer-based multilingual large language model, trained on cleaned text from 100 languages. Multilingual models have demonstrated promise across various NLP tasks, particularly in tasks like semantic similarity and paraphrase detection.

3 Urdu paraphrased text reuse and plagiarism detection approaches

This section describes the D-TRaPPD based approaches that we used for the Urdu paraphrased text reuse and plagiarism detection task. D-TRaPPD is loosely based on Iqbal et al. research [23] and contains four major modules, (i) Embeddings Input (ii) Convolutional Neural Network (CNN), (iii) Long-Short Term Memory (LSTM), and (iv) Semantic Similarity Estimation. On the basis of the Embeddings Input module, we have designed two approaches

using the D-TRaPPD architecture (See Fig. 1), (i) WE-D-TRaPPD, and (ii) SE-D-TRaPPD. Apart from the Embeddings Input module, the CNN, LSTM, and Semantic Similarity Estimation modules are common in both approaches.

The Embeddings Input module is designed to extract word and sentence embedding vectors from the input texts, convert these vectors into matrices, and feed them into the CNN module. Specifically, two types of embedding matrices are generated: (i) word embedding matrices and (ii) sentence embedding matrices. These embedding matrices are then processed by a Siamese CNN to extract relevant and important structures from the input text. The output of the CNN module, consisting of bi- and tri-gram feature maps, is subsequently fed into the LSTM module for capturing sequential dependencies. The resultant difference vector D is utilized as a feature vector to estimate the semantic similarity between the two input texts. This vector D is passed through two ReLU-activated fully connected layers h , followed by dropout, to fine-tune the estimation of semantic similarity. The complete workflow is also shown in Algorithm 1. Furthermore, the details related to each module are discussed in the following sections.

Algorithm 1 Step-by-Step Procedure for generating embeddings and classifying paraphrased text in proposed WE-D-TRaPPD and SE-D-TRaPPD approaches.

```

1: Input: Text Pair (T1, T2)
2: Output: Similarity Score between T1 and T2
3: Step 1: Preprocessing
4: for each text  $T_i$  in (T1, T2) do
5:   Remove numbers, punctuation, newlines, and non-standard Urdu characters.
6: end for
7: Step 2: Generate Embedding Matrices
8: for each text  $T_i$  in (T1, T2) do
9:   if word-level then
10:    Tokenize text into words.
11:    Generate word embeddings using pre-trained word embeddings.
12:    Concatenate word embeddings to form word embedding matrix  $M_i$ .
13:   else if sentence-level then
14:    Tokenize text into sentences.
15:    Generate sentence embeddings using pre-trained sentence embeddings.
16:    Extract word embeddings from the last encoding layer for each sentence.
17:    Concatenate word embeddings to form sentence embedding matrix  $M_i$ .
18:   end if
19: end for
20: Step 3: Standardize Embedding Matrices
21: if  $M_1$  and  $M_2$  have different dimensions then
22:   Pad the smaller matrix with zeros to match the size of the larger matrix.
23: end if
24: Step 4: Convolutional Neural Network (CNN)
25: for each embedding matrix  $M_i$  in ( $M_1$ ,  $M_2$ ) do
26:   Extract feature maps.
27: end for
28: Step 5: Long Short-Term Memory (LSTM)
29: for feature maps corresponding to pair (T1, T2) do
30:   Compute LSTM features and element wise difference  $D$ .
31: end for
32: Step 6: Semantic Similarity Estimation
33: Calculate the similarity score between T1 and T2.
34: Step 7: Output
35: Normalize the similarity score in the range [0, 1].
36: Use the similarity score to classify if T2 is a paraphrase of T1.

```

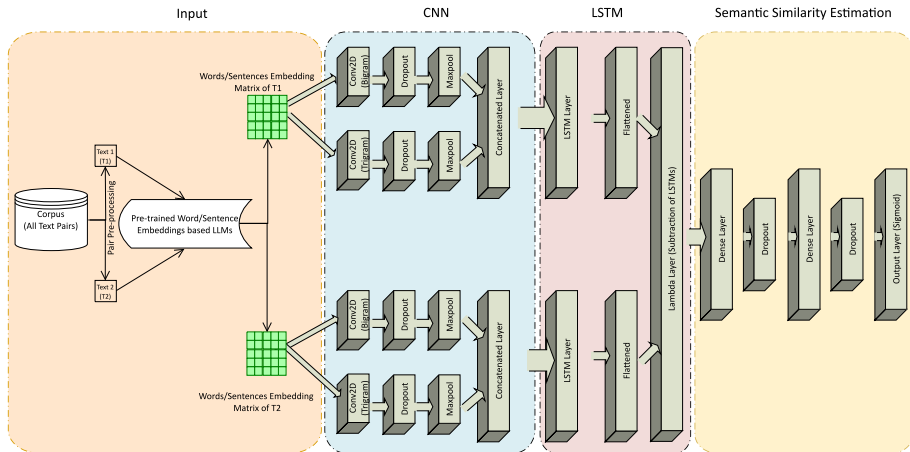


Fig. 1 Abstract level architecture of WE-D-TRaPPD and SE-D-TRaPPD approaches for Urdu paraphrased text reuse and plagiarism detection

3.1 Embeddings input

This module makes use of pre-trained Language Models (LLMs) such as BERT [26], RoBERTa [55], or ELMo [56] to generate embeddings for the input texts. The process begins with pre-processing the Urdu text, which involves removing numbers, punctuation, extra white spaces, newlines, and any non-standard Urdu characters. For each text in the pair (T1 and T2), embeddings are extracted from pre-trained language models and transformed into matrices (M1 and M2), representing texts in a higher-dimensional vector space. Each matrix has an order $d * n$, where d is the dimension of the embedding vector for each word or sentence, and n is the number of words or sentences, depending on the type of language model used.

Given the various pre-trained LLMs employed in this study (see Section 4.2.1), some models generate embeddings for the individual words while others produce embeddings for the entire sentences. Consequently, to create the embedding matrix for each text T1 and T2, two types of embedding matrices are obtained: (i) words embedding matrix and (ii) sentences embedding matrices. The following sections provide details on how each embedding matrix is obtained.

3.1.1 Words embedding matrix

Two approaches were employed to construct word embedding matrices, dependent on the specific LLM:

- Direct Word Embeddings:** In this approach, the input text is first segmented into individual words, as the pre-trained model is a word embedding model (e.g., Word2Vec [57], FastText [24]) that generates embeddings for individual words. These word embeddings are subsequently concatenated to form an embedding matrix of size $d * n$, where d signifies the embedding dimension and n corresponds to the total number of words in the document.
- Extracting Word Embeddings from Sentence-Level Representations:** The input text is initially segmented into individual sentences, as the pre-trained model (e.g., BERT [26], RoBERTa [55]) outputs embeddings for sentences. To obtain embeddings for each word

within the sentences, embeddings are extracted from the last encoding layer of the underlying sentence embedding model. These word embeddings are then concatenated to form an embedding matrix of order $d * n$, where d represents the embedding dimension and n corresponds to the total number of words in the text.

3.1.2 Sentences embedding matrix

Similar to the construction of word embedding matrix, the sentence embedding matrices for each input text are obtained in two ways, based on the type of pre-trained LLMs used:

- **Direct Sentence Embeddings:** In this approach, the input text is initially segmented into individual sentences, as the pre-trained model (e.g., BERT [26], RoBERTa [55]) outputs embeddings sentences. These sentence embeddings are then concatenated to form an embedding matrix of size $d * n$, where d signifies the embedding dimension and n corresponds to the total number of sentences in the text. This matrix effectively encodes the text as a high-dimensional numerical representation.
- **Sentence-Level Representations from Word Embeddings:** To obtain sentence-level embeddings, the input text is first segmented into sentences. Each sentence is then word tokenized and each word is mapped to a dense, d -dimensional vector using a pre-trained word embedding model (e.g., Word2Vec [57], FastText [24]). The word embedding vectors for each sentence are averaged to obtain a sentence vector. These sentence embeddings are subsequently concatenated to form an embedding matrix of size $d * n$, where d signifies the embedding dimension and n corresponds to the total number of sentences in the text.

3.1.3 Embeddings matrix standardization

Given that source and derived texts typically differ in length, both in terms of the number of sentences and the number of words, it is crucial to address this variation to ensure consistent input sizes for subsequent processing. This discrepancy in lengths can pose challenges for deep neural architectures (e.g., CNNs) that require fixed-size input matrices. To manage this variation, we employ a padding strategy. Specifically, we add extra padding of zeros to the smaller matrix, whether it be a word embedding matrix or a sentence embedding matrix, until it matches the dimensions of the larger matrix. This approach ensures that both matrices have a uniform size, enabling the model to process them efficiently and maintain the integrity of the feature representation. Padding with zeros is a common practice in NLP as it effectively neutralizes the impact of the added elements, preventing them from influencing the embedding space while preserving the alignment of meaningful data.

3.2 Convolutional neural networks (CNNs)

While Convolutional Neural Networks (CNNs) are predominantly associated with image recognition tasks, they have demonstrated a robust capability to extract meaningful and significant structures from text data as well, as represented by embedding vectors [58]. This utility arises from the CNN's inherent ability to capture local patterns in data through convolutional and pooling layers, which can extract strong local textual features regardless of their position within the text [59].

In our approach, the embedding matrices produced by the Embeddings Input module (Section 3.1) are fed into a Siamese-CNN (consists of two identical CNNs that work on

different input matrices to produce comparable output vectors) [60]. The primary role of the CNN in this context is to identify and extract relevant patterns and structures from the text data. This involves capturing local region information, which is crucial for understanding the intricacies of text, such as phrase structures and local dependencies. The extracted local features are then used as inputs to the LSTM module, which processes this information to capture sequential dependencies.

The convolutional layers are designed to perform word- or sentence-wise convolutions on the input embedding matrices. To capture local dependencies of varying lengths within the text, we use convolutional kernels of sizes 2 (bi-grams) and 3 (tri-grams), with 128 filters and a stride size of 1 for all spatial dimensions. Bi-gram filters are useful for detecting features related to pairs of embedding vectors (words or sentences), whereas tri-gram filters extend this capability to triplets of vectors (words/sentences) in the embedding matrices. The motivation for using these convolutions stems from their ability to learn discriminating n-gram features, which are highly valuable for tasks such as sentence similarity analysis.

The convolutional layers are activated using the Rectified Linear Unit (ReLU) function [61]). ReLU is chosen for its efficiency in avoiding the vanishing gradient problem, thus enabling faster and more effective training of the neural network. To prevent the network from overfitting, dropout layers [62] are employed. Dropout randomly deactivates a subset (empirically chosen 0.5 percent) of neurons during training, which helps in creating a more generalized model that performs well on unseen data.

Following the convolutional and dropout layers, a max-pooling layer with a pooling window of (1, 298) is added. Here, 1 indicates the height of the pooling window, and 298 indicates its width. This layer compresses the width of the input feature map to a single column while retaining its height, effectively reducing the spatial dimensions and highlighting the most significant features across the width of the input. This step enhances the efficiency of subsequent processing and ensures that the focus remains on the most important information.

Finally, both the bi-gram and tri-gram feature maps generated by the convolutional layers are concatenated using a lambda layer. This concatenation ensures that the model leverages information from multiple levels of n-gram features, enriching the representation of the text data and enhancing the model's capability to understand and compare different text sequences effectively.

3.3 Long-short term memory (LSTM)

Once the CNN module processes the input text in the form of embedding matrices, generating bi- and tri-gram feature maps, these maps are fed into the LSTM module. The LSTM module is configured with a 64-dimensional output space, enabling it to capture and represent complex patterns in the sequence data. The LSTM's architecture is enhanced with an L_2 kernel regularizer, which helps prevent overfitting by penalizing large weights, thus ensuring the model generalizes well to unseen data [63].

To facilitate the comparison between two text documents, the output vectors from the LSTM layer, corresponding to the source and derived documents, are processed to compute an element-wise difference vector, $D(\mathbf{1})$. This is achieved using a *lambda* layer, which efficiently calculates the difference between the LSTM's output vectors. The resultant difference vector D captures the dissimilarities between the two input sequences.

$$D(f_1, f_2) = |f_{1i} - f_{2i}| \quad i \in d \quad (1)$$

The difference vector D is then utilized as a feature vector to estimate the semantic similarity between the two text documents. This approach ensures that the model not only considers

the local features extracted by the CNN but also accounts for the sequential dependencies and differences identified by the LSTM. The integration of these architectures thus provides a robust framework for accurately determining the semantic similarity and, consequently, detecting paraphrased or plagiarized text.

3.4 Semantic similarity estimation

To estimate the semantic similarity between the two input text documents, the element-wise difference vector D , which captures the dissimilarities between the source and derived documents, is utilized. This vector D is first passed through two fully connected layers, each activated by the ReLU. Each of these fully connected layers is followed by a dropout layer [64].

$$h = \text{ReLU}(W \cdot D + b) \quad (2)$$

The shared weights, denoted as W , are utilized to connect the hidden layers, while the shared bias, represented as b , is applied to these layers.

$$\text{Sim}(f_1, f_2) = \text{sigmoid}(h(h(D))) \quad (3)$$

At the output layer, a sigmoidal activation function [65] is employed. The sigmoid function is particularly suitable for binary classification tasks because it maps the input to a value between 0 and 1, effectively transforming the output into a probability score. In this context, the sigmoid activation function computes a similarity score between the two feature vectors derived from the input documents. This score ranges from 0 to 1, where values closer to 1 indicate higher similarity, and values closer to 0 indicate lower similarity.

The final stage of the model involves training it to differentiate between paraphrased and non-paraphrased text documents based on the computed similarity scores. By leveraging the features learned through the fully connected layers and the regularization provided by dropout, the model is able to effectively distinguish between documents that are paraphrased or plagiarized and those that are not. This comprehensive approach ensures a robust and accurate assessment of semantic similarity, which is crucial for detecting text reuse and plagiarism in the Urdu language.

4 Experimental set-up

This section describes the corpora, approaches, evaluation methodology, hyper-parameter settings, and evaluation measures used for Urdu paraphrased text reuse and plagiarism detection task.

4.1 Corpora

To evaluate the performance of the proposed D-TRaPPD based approaches (see Section 3) for Urdu paraphrased text reuse and plagiarism detection task, we have conducted a series of experiments using two freely available gold-standard document-level corpora, i.e., (i) COUNTER [19], and (ii) UPPC [41]. COUNTER corpus has a total of 1,200 (source = 600, derived = 600) document pairs with three obfuscation levels, (i) wholly derived (135), (ii) partially derived (288), and (iii) non-derived (177). It contains real cases of Urdu text reuse compiled from newspapers. UPPC corpus has 160 (source = 20, suspicious = 140)

manually generated paraphrased plagiarised and non-plagiarised text documents. Among the suspicious documents, 75 are plagiarised and 65 are non-plagiarised.

The reasons for shortlisting these two corpora are as follows, (i) the largest corpora available for Urdu paraphrased text reuse and plagiarism detection task, (ii) constructed at document level with real and simulated cases of Urdu paraphrased text reuse and plagiarism, (iii) freely available, licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License¹.

4.2 Approaches

We have applied a handful variants of D-TRaPPD based WE-D-TRaPPD and SE-D-TRaPPD approaches.

WE-D-TRaPPD approach variants: (i) W2V-WE-D-TRaPPD, (ii) FastText-WE-D-TRaPPD, (iii) ELMO-W-ED-TRaPPD, (iv) BERT-WE-D-TRaPPD, (v) LaBSE-WE-D-TRaPPD, (vi) DistilRoBERTa-WE-D-TRaPPD, , (vii) DistilBERT-WE-D-TRaPPD, and (viii) USE-SE-D-TRaPPD.

Similarly, SE-D-TRaPPD approach variants: (i) W2V-SE-D-TRaPPD, (ii) FastText-SE-D-TRaPPD, (iii) ELMO-SE-D-TRaPPD, (iv) LASER-SE-D-TRaPPD, (v) BERT-SE-D-TRaPPD, (vi) LaBSE-SE-D-TRaPPD, (vii) DistilRoBERTa-SE-D-TRaPPD, (viii) DistilBERT-SE-D-TRaPPD, and (ix) USE-SE-D-TRaPPD.

For each variant, the step-by-step procedure for generating embeddings and conducting the classification task is detailed in Algorithm 1. This algorithm outlines the entire process, from pre-processing the input texts to the final semantic similarity estimation.

As per our knowledge, this study is the first to apply a diverse range of hybrid DNN approaches using pre-trained LLMs on two document-level Urdu paraphrased text reuse and plagiarism corpora, containing both real and simulated cases.

4.2.1 Pre-trained LLMs

To extract word and sentence embedding vectors, several mono-lingual and multi-lingual pre-trained LLMs are used.

- Urdu-Word2Vec [57]: A pre-trained mono-lingual Urdu Word2Vec (continuous bag-of-word [66]) word embeddings model by [67], further trained on additional Urdu text crawled from various pages over the web. Output embeddings size = 300 dimensions.
- FastText [24]: The character-based pre-trained word vectors for Urdu were generated using fastText API ² and trained on Wikipedia and Common Crawl ³ using continuous bag-of-word (BoW) with position-weights and character 5-gram. The output embeddings have a size of 300 dimensions.
- ELMO (Embeddings for Language Models) [56]: A pre-trained contextualized mono-lingual LLM for Urdu [68], trained over the Wikipedia and Common Crawl data. Output embeddings size = 1024 dimensions.
- LASER [38]: A multilingual sentence embedding model, developed for 93 languages to perform zero-short cross-lingual transfer. It is trained on 223 million parallel sentences. Output embeddings size = 1024 dimensions.

¹ <https://creativecommons.org/licenses/by-nc-sa/4.0/> - Last visited: 8-June-2022

² <https://fasttext.cc/docs/en/crawl-vectors.html>

³ <https://commoncrawl.org/>

- mUSE [28]: A knowledge distilled version of the multi-lingual Universal Sentence Encoder [39], trained for 50+ languages using parallel data from OPUS [69]. It provides both word and sentence embedding vectors with 768 and 512 dimensions, respectively.
- LaBSE [40]: A BERT-based dual-encoder transformer architecture trained on 6 billion translated text pairs for 109 languages. It returns both word and sentence embedding vectors with 768 dimensions.
- XLM-R [28]: A multi-lingual transformer based distilled version of XLM-RoBERTA [55], trained on parallel data of 50 languages. It returns both word and sentence embedding vectors with 768 dimensions.
- mBERT/XLM-R Mean [28]: A joint model, developed by taking the mean-pooling of the outputs of the existing multi-lingual BERT [26] and XLM-RoBERTA [55], trained on parallel textual data of 50 languages. It generates 768-dimensional vectors for both words and sentences.
- Distil-mBERT [28]: A knowledge distilled multi-lingual version of the Distil-BERT [70], fine-tuned on parallel data in 50 languages. It returns a 768-dimensional embedding vector for a given word or sentence.

4.3 Evaluation methodology

The primary objective of tackling the problem of Urdu paraphrased text reuse and plagiarism detection as a supervised binary classification task is to distinguish between derived and non-derived text. UPPC corpus has only two obfuscation levels (two classes, i.e., plagiarised and non-plagiarised), the plagiarized class is termed as *derived* whereas non-plagiarized as *non-derived*. On the other hand, COUNTER corpus has three obfuscation levels (wholly derived, partially derived, and non-derived). For binary classification, two classes (i.e., wholly derived and partially derived) are merged to make a single class named *derived*, whereas the non-derived class is considered as *non-derived*. The derived and non-derived classes are integer encoded as 0 and 1, respectively.

4.4 Hyper-parameter Settings

In developing the WE-D-TRaPPD and SE-D-TRaPPD models, we adopted a hybrid CNN-LSTM architecture to harness the strengths of both networks-CNNs for capturing local textual patterns and LSTMs for understanding sequential relationships. While this combination increases the model's complexity, we have carefully managed it through the use of dropout, L2 regularization, optimizers, and rigorous cross-validation. These measures ensure that the model remains robust and generalizes well, despite the added complexity in training and deployment. We hypothesize that the improved performance metrics justify the adoption of this complex hybrid approach, particularly in the context of paraphrase detection for Urdu, where capturing both local and sequential features is crucial.

The implementation of all approaches was done using Keras⁴, which is a widely used deep learning API written in Python. We experimented with various combinations of hyper-parameters and selected those that resulted in better performance. Since we conducted experiments on two distinct corpora for paraphrased text reuse and plagiarism detection task (refer to Section 4.1), most of the hyper-parameter settings remained consistent such as: (i) stratified 10-fold cross-validation, (ii) *validation split* (0.1), (iii) *dropout rate* (0.5), (iv) *optimizer* (adam), (v) *kernel regularizer* (L_2), and (vi) *epochs* (50, 125, 250, 500, 1000,

⁴ <https://keras.io/>

1500, 2000). Smaller *batch sizes* (16,32,64,128) were used because the corpora have a smaller number of paraphrased text reuse and plagiarism cases. For binary classification task *loss* is ‘binary cross-entropy’.

4.5 Evaluation measures

The results of the proposed approach are presented using standard evaluation measures, which have been commonly employed in past studies [19, 45] for Urdu text reuse and plagiarism detection tasks. These measure are *Precision* (see (4)), *Recall* (see (5)), and *F₁ score* (see (6)). Regarding classification, we have denoted True Positives (TP) as the relevant text pairs that were accurately classified as a non-derived class and True Negatives (TN) as the text pairs that were precisely classified as derived.

$$\text{Precision} = \frac{TP}{TP + FP} \quad (4)$$

$$\text{Recall} = \frac{TP}{TP + FN} \quad (5)$$

$$F_1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \quad (6)$$

5 Results and discussion

In this section, we discuss the results of the experiments performed for the Urdu paraphrased text reuse and plagiarism detection task described in Section 4. Our goal was to investigate the performance of our proposed approaches on both real and simulated corpora, as well as to compare them with existing state-of-the-art methods in the field. Through this empirical investigation and comparison, we aimed to determine which approaches are best suited for detecting paraphrased text reuse and plagiarism in Urdu at the document level. Furthermore, we also aimed to identify the strengths and weaknesses of each approach with respect to the observed corpora. To our knowledge, no prior research has provided such a detailed comparison for Urdu.

5.1 Results using SE-D-TRaPPD

Tables 2 and 3 summarize the performance of the SE-D-TRaPPD approach, evaluated on two different corpora, i.e., UPPC and COUNTER. For the UPPC corpus, the overall best performance was achieved by USE ($F_1 = 95.15$) in classifying documents with simulated cases of text reuse and plagiarism as either paraphrased or non-paraphrased. On the other hand, FastText had the lowest performance among all pre-trained LLMs ($F_1 = 82.98$) in this binary classification task. For the COUNTER corpus, USE achieved the highest score ($F_1 = 91.77$) in detecting real cases of text reuse and plagiarism, while DistilBERT had the lowest performance ($F_1 = 82.98$) among the pre-trained LLMs. These results demonstrate the effectiveness of the SE-D-TRaPPD approach for document-level Urdu text reuse and plagiarism detection and highlight the importance of selecting an appropriate pre-trained LLM for the task at hand.

It can be observed from SE-D-TRaPPD results that the state-of-the-art transformer-based sentence embedding models and conventional pre-trained word embedding models show

Table 2 SE-D-TRaPPD results for document-level Urdu paraphrased text reuse and plagiarism detection on UPPC corpus for binary classification - **Bold** numbers indicate the best scores

Approach	Precision	Recall	F1
W2V-SE-D-TRaPPD	87.91	99.26	93.22
FastText-SE-D-TRaPPD	81.37	98.43	88.53
ELMO-SE-D-TRaPPD	90.65	99.20	94.72
LASER-SE-D-TRaPPD	88.97	99.40	93.89
BERT-SE-D-TRaPPD	89.51	98.91	93.97
LaBSE-SE-D-TRaPPD	90.85	99.32	94.89
DistilRoBERTa-SE-D-TRaPPD	90.58	99.23	94.70
DistilBERT-SE-D-TRaPPD	85.82	92.61	89.07
USE-SE-D-TRaPPD	91.01	99.69	95.15

comparable F_1 scores for both UPPC and COUNTER. A possible explanation for this could be the averaging of individual word embedding vectors to create a sentence embedding vector. Even the ELMO word embedding model has achieved the second position among other pre-trained LLMs in accurately classifying documents as either paraphrased or non-paraphrased.

The SE-D-TRaPPD approach exhibits significantly higher F_1 scores for binary classification in UPPC than in COUNTER. This is largely due to the high recall in UPPC, which may be attributed to the small number of obfuscated documents in the corpus, making the model less generalized. Additionally, UPPC's clear differentiation between paraphrased and non-paraphrased classes may also contribute to the high recall. Conversely, the low recall in COUNTER for binary classification may be due to the distribution of classes, with partially and wholly derived documents not belonging to the same distribution as the first class. This creates confusion for the model, leading to a high number of false negatives and a drop in recall. Overall, the SE-D-TRaPPD approach combined with pre-trained LLMs demonstrates high precision and accuracy in distinguishing between paraphrased and non-paraphrased Urdu documents in UPPC.

The significant difference in results between UPPC and COUNTER may be attributed to the type of paraphrased text reuse and plagiarism cases present in each corpus. COUNTER contains real cases, while UPPC includes simulated cases for the Urdu language. Literature suggests that real cases are more challenging to detect than simulated cases. Additionally, the precision-recall difference is substantial in COUNTER, while it is minimal in UPPC, indicating that the data is well-distributed in UPPC but overlapping in COUNTER, resulting in a low recall.

Table 3 SE-D-TRaPPD results for document-level Urdu paraphrased text reuse and plagiarism detection on COUNTER corpus for binary classification

Approach	Precision	Recall	F1
W2V-SE-D-TRaPPD	96.30	87.13	91.00
FastText-SE-D-TRaPPD	95.59	85.27	90.12
ELMO-SE-D-TRaPPD	98.21	85.79	91.58
LASER-SE-D-TRaPPD	98.44	83.28	90.22
BERT-SE-D-TRaPPD	96.57	85.68	90.79
LaBSE-SE-D-TRaPPD	97.60	85.66	91.23
DistilRoBERTa-SE-D-TRaPPD	97.83	86.30	91.70
DistilBERT-SE-D-TRaPPD	85.59	80.56	82.98
USE-SE-D-TRaPPD	97.01	87.07	91.77

Table 4 WE-D-TRaPPD results for document-level Urdu paraphrased text reuse and plagiarism detection on UPPC corpus for binary classification

Approach	Precision	Recall	F1
W2V-WE-D-TRaPPD	79.65	89.36	84.15
FastText-WE-D-TRaPPD	80.54	89.56	84.74
ELMO-WE-D-TRaPPD	84.45	94.40	89.13
BERT-WE-D-TRaPPD	90.44	98.67	94.37
LaBSE-WE-D-TRaPPD	91.32	98.45	94.74
DistilRoBERTa-WE-D-TRaPPD	90.66	98.99	94.64
DistilBERT-WE-D-TRaPPD	87.07	91.42	89.16
USE-WE-D-TRaPPD	90.66	99.22	94.74

The results show that pre-trained LLMs can be effective in detecting text reuse and plagiarism at the document level, but the choice of the model may depend on the characteristics of the corpus being analyzed. The SE-D-TRaPPD approach also seems to be a promising method for this task.

5.2 Results using WE-D-TRaPPD

Tables 4 and 5 present the result of the study, which evaluated the effectiveness of various pre-trained word embedding models for detecting document-level paraphrased text reuse and plagiarism using the WE-D-TRaPPD approach. Two different corpora, UPPC and COUNTER, were used for binary classification of paraphrased and non-paraphrased documents. Results show that the WE-D-TRaPPD approach performed better for the UPPC corpus than for the COUNTER corpus. LaBSE and USE achieved the best overall performance for the UPPC corpus, with an F_1 score of 94.74. In contrast, Word2Vec has the lowest performance, with an F_1 score of 84.15. For the COUNTER corpus, the best-performing model was DistilRoBERTa, with an F_1 score of 91.42, while DistilBERT had the lowest performance, with an F_1 score of 82.66. The findings suggest that the choice of a pre-trained LLM is important for the effective detection of document-level paraphrased text reuse and plagiarism for Urdu.

The state-of-the-art transformers-based sentence embedding models show comparable F_1 scores to conventional pre-trained word embedding models for both UPPC and COUNTER corpora. The averaging of individual word embedding vectors to generate sentence embedding vectors could be a possible reason for the competitive results produced by these models. In UPPC, the WE-D-TRaPPD approach achieves higher F_1 scores for binary classification

Table 5 WE-D-TRaPPD results for document-level Urdu paraphrased text reuse and plagiarism detection on COUNTER corpus for binary classification

Approach	Precision	Recall	F1
W2V-WE-D-TRaPPD	96.96	85.58	90.91
FastText-WE-D-TRaPPD	96.66	86.12	91.08
ELMO-WE-D-TRaPPD	97.58	85.44	91.10
BERT-WE-D-TRaPPD	96.53	85.69	90.78
LaBSE-WE-D-TRaPPD	96.50	85.94	90.91
DistilRoBERTa-WE-D-TRaPPD	95.65	87.56	91.42
DistilBERT-WE-D-TRaPPD	91.32	80.12	85.35
USE-WE-D-TRaPPD	95.57	87.52	91.36

compared to COUNTER, mainly due to high recall. The small number of obfuscated documents in UPPC and the clear distinction between paraphrased and non-paraphrased classes in the corpus could be responsible for this. However, the COUNTER corpus has a possibly overlapping distribution of documents, which may result in low recall. Furthermore, the type of paraphrased text reuse and plagiarism cases in UPPC and COUNTER could also contribute to the significant difference between the results. COUNTER contains real cases, which are known to be harder to detect than simulated cases, whereas UPPC only has simulated cases. Additionally, there is a large difference between the precision and recall for COUNTER, while the difference is minimal for UPPC, indicating better data distribution in the latter. Overall, the WE-D-TRaPPD approaches combined with LaBSE and USE pre-trained LLMs are highly accurate in distinguishing between paraphrased and non-paraphrased Urdu documents in the UPPC corpus.

5.3 Best results comparison

The present study evaluated the performance of several approaches for detecting document-level paraphrased text reuse and plagiarism detection in the Urdu language. The results are summarized in Table 6, where the best results achieved by each approach are presented for both COUNTER and UPPC corpora. The table shows that the SE-D-TRaPPD approach outperformed the WE-D-TRaPPD approach for both corpora. However, the binary classification task results were higher for UPPC than for COUNTER corpus, indicating that real cases of paraphrased text reuse and plagiarism detection are harder to detect than simulated cases in Urdu.

In addition, the table reveals that the SE-D-TRaPPD approach produced slightly better results than WE-D-TRaPPD. This could be attributed to the pre-trained LLMs, where the sentence-level models, such as USE and DistilRoBERTa, were modified for word-level embeddings, potentially resulting in lower-quality embeddings for words than sentences.

Moreover, the F_1 scores increased as the computational complexity of the models increased, from word embeddings to sentence embeddings. Overall, these findings demonstrate the effectiveness of the proposed approaches for detecting document-level paraphrased text reuse and plagiarism detection in Urdu.

5.4 Results comparison with the state-of-the-art urdu paraphrased text reuse and plagiarism detection

Table 7 presents a detailed comparison of the proposed document-level paraphrased text reuse and plagiarism detection approaches against the current state-of-the-art approaches for

Table 6 Summarized best results of proposed approaches for document-level Urdu paraphrased text reuse and plagiarism detection for both COUNTER and UPPC corpora

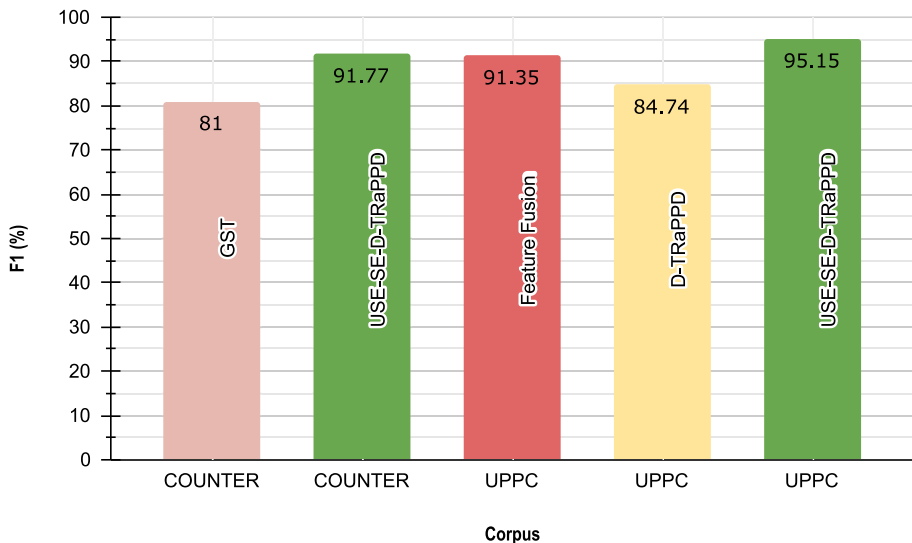
Corpus	Approach	Precision	Recall	F1
COUNTER	USE-SE-D-TRaPPD	97.01	87.07	91.77
COUNTER	DistilRoBERTa-WE-D-TRaPPD	95.65	87.56	91.42
UPPC	USE-SE-D-TRaPPD	91.01	99.69	95.15
UPPC	LaBSE-WE-D-TRaPPD	91.32	98.45	94.74
UPPC	USE-WE-D-TRaPPD	90.66	99.22	94.74

Table 7 Document-level D-TRaPPD: Comparison between the state-of-the-art and the newly proposed approaches for both COUNTER and UPPC corpora

Corpus	Classifier	Approach	Precision	Recall	F1
Existing state-of-the-art					
COUNTER	NB	GST [71]	–	–	81.00
UPPC	GNB	Feature-fusion [21]	–	–	91.35
UPPC	DNN	D-TRaPPD [23]	80.54	89.56	84.74
Proposed Approaches Scores					
COUNTER	DNN	USE-SE-D-TRaPPD	97.01	87.07	91.77
UPPC	DNN	USE-SE-D-TRaPPD	91.01	99.69	95.15

Urdu. The “Classifier” column indicates whether the approach utilizes a traditional machine learning classifier or a DNN-based classifier. The ‘-’ indicates that the authors did not provide precision and recall metrics for their approaches. The table is divided into two sections: “Existing State-of-the-Art” for the benchmark scores and “Proposed Approaches Scores” for the results of our proposed approaches using pre-trained LLMs. It is evident from the Table 7 that the newly proposed approaches outperform the existing state-of-the-art approaches for Urdu paraphrased text reuse and plagiarism detection for both COUNTER and UPPC corpora.

To reinforce the effectiveness of the proposed approaches and justify the adoption of more complex DNN architectures like WE-D-TRaPPD and SE-D-TRaPPD for Urdu paraphrased text reuse and plagiarism detection, we have included a comparative illustration. Figure 2 presents the F_1 scores achieved by various models on both the COUNTER and UPPC corpora, with the x-axis representing different models and the y-axis displaying the F_1 score percentages. The figure clearly demonstrates that our proposed models surpassed the performance of previous state-of-the-art approaches.

**Fig. 2** Performance Evaluation of Proposed and Existing State-of-the-art Approaches for COUNTER and UPPC corpora

5.4.1 COUNTER Corpus Results Comparison

We took inspiration from the work of Sharjeel et al. [19], and considered it as a baseline for document-level Urdu paraphrased text reuse and plagiarism detection on the COUNTER corpus. They achieved an F_1 score of 81.00 using Naive Bayes (NB) classifier and a content-based approach, namely GST [71]. For COUNTER, our proposed D-TRaPPD approaches, USE-SE-D-TRaPPD ($F_1 = 91.77$), outperformed the existing surface-level approach (content-based with $F_1 = 81.00$) in binary classification.

5.4.2 UPPC corpus results comparison

For the UPPC dataset, Feature-Fusion approach [21] established a baseline for document-level paraphrase plagiarism detection with an F_1 score of 91.35 using Gaussian Naive Bayes (GNB) classifier. To further validate our proposed method, we compared our results with recent D-TRaPPD [23], using a DNN-based classifier with word embedding language models for the UPPC corpus and achieved F_1 score of 84.74. This comparison is critical, as it highlights the advantages of employing a more complex architecture like D-TRaPPD.

The results demonstrate the effectiveness of our proposed variant, USE-SE-D-TRaPPD ($F_1 = 95.15$), significantly outperformed both the traditional machine learning approach and the DNN-based baseline. The superior performance of our proposed approach justifies the increased complexity and cost associated with using pre-trained LLMs, as they provide more robust and accurate paraphrase detection capabilities.

6 Conclusion and future direction

In conclusion, this paper presents a novel approach to detect paraphrased text reuse and plagiarism in the Urdu language using a DNN-based architecture called D-TRaPPD that incorporates pre-trained LLMs. The proposed architecture uses CNN and LSTM to extract silent features from the input text and has two variants (i.e., WE-D-TRaPPD and SE-D-TRaPPD) based on word embeddings and sentence embeddings. The proposed approaches are evaluated using two gold-standard document-level Urdu text reuse and plagiarism corpora, COUNTER and UPPC.

The results indicate that the USE-SE-D-TRaPPD is the most appropriate approach for distinguishing between various levels of Urdu paraphrased text reuse and plagiarism. Moreover, the low scores obtained for real cases of text reuse, compared to the high scores achieved for simulated cases, indicate that detecting real cases at the document level is challenging and requires additional research. Moreover, the proposed approaches have empirically demonstrated superiority over state-of-the-art methods for Urdu paraphrased text reuse and plagiarism detection task.

While the proposed WE-D-TRaPPD and SE-D-TRaPPD models have demonstrated promising results in detecting Urdu paraphrased text reuse and plagiarism, several avenues for future research could further refine and expand upon these approaches. To address the scarcity of annotated Urdu data, future research could investigate few-shot and zero-shot learning methods. These techniques aim to enable models to learn effectively from limited labeled examples, making them more adaptable to new, unseen data. Finally, incorporating Urdu-specific preprocessing methods, such as stemming, lemmatization, and handling orthographic variations, directly into the DNN models could further improve their performance.

By pursuing these research directions, future studies could build upon the foundations laid by this work and contribute to the advancement of Urdu paraphrased text reuse and plagiarism detection.

Author Contributions All authors contributed to the research and manuscript preparation. Their specific roles are as follows:

- Hafiz Rizwan Iqbal: Conceptualized the idea, designed the approaches, conducted experiments, compiled results, wrote the manuscript, and handled submission.
- M. Sharjeel: Assisted in refining the idea and provided feedback on the manuscript.
- Jawad Shafi: Reviewed the manuscript and provided feedback.
- Usama Mehmood: Reviewed the manuscript and rebuttal document, and provided feedback.
- Agha Ali Raza: Supervised the research, secured funding, and provided guidance throughout the study.

Funding This research has received funding from the Higher Education Commission (HEC) of Pakistan as part of the National Research Program for Universities (NRPU) grant (Project ID : 9854/Punjab/NRPU/R&D/HEC/2017), and the Information Technology University, Lahore, Pakistan.

Data Availability The data for this study was gathered from the publicly available datasets referenced in Section 4.1. Therefore, data sharing does not apply to this article as there is no data generated in the course of this research.

Declarations

Ethical Approval Not applicable.

Competing Interests The authors declare no competing interests.

References

1. Clough P, Gaizauskas R () Corpora and text re-use. Handbook of corpus linguistics, handbooks of linguistics and communication science, pp 1249–1271
2. Butakov S, Scherbinin V (2009) The toolbox for local and global plagiarism detection. *Comput Educ* 52(4):781–788
3. Osman AH, Salim N, Abuobieda A (2012) Survey of text plagiarism detection. *Comput Eng Appl J (ComEngApp)* 1(1):37–45
4. Sousa-Silva R (2017) Detecting translingual plagiarism and the backlash against translation plagiarists. *Language and Law= Linguagem e Direito* 1(1)
5. Foltýnek T, Meuschke N, Gipp B (2019) Academic plagiarism detection: a systematic literature review. *ACM Comput Surv* 52:1–42
6. Wahle JP, Ruas T, Foltýnek T, Meuschke N, Gipp B (2022) Identifying machine-paraphrased plagiarism. Springer, In international conference on information, pp 393–413
7. Chowdhury HA, Bhattacharyya DK (2018) Plagiarism: Taxonomy, tools and detection techniques. In Proceedings of the 19th national convention on knowledge, library and information networking
8. Potthast M, Stein B, Barrón-Cedeño A, Rosso P (2010) An evaluation framework for plagiarism detection. In Proceedings of the 23rd international conference on computational linguistics: Posters, Association for Computational Linguistics, pp 997–1005
9. Potthast M, Hagen M, Gollub T, Tippmann M, Kiesel J, Rosso P, Stamatas E, Stein B (2013) Overview of the 5th international competition on plagiarism detection. In CLEF conference on multilingual and multimodal information access evaluation, CELCT, pp 301–331
10. Barrón-Cedeño A, Vila M, Martí MA, Rosso P (2013) Plagiarism meets paraphrasing: Insights for the next generation in automatic plagiarism detection. *Comput Linguist* 39(4):917–947
11. Franco-Salvador M, Rosso P, Montes-y-Gómez M (2016) A systematic study of knowledge graph analysis for cross-language plagiarism detection. *Inf Process Manag* 52:550–570
12. Noraset T, Lowphansirikul L, Tuarob S (2021) Wabiqa: A wikipedia-based thai question-answering system. *Inf Process Manag* 58:102–431
13. Zandie R, Mahoor MH (2023) Topical language generation using transformers. *Nat Lang Eng* 29(2):337–359

14. Paris CL, Swartout WR, Mann WC (2013) Natural language generation in artificial intelligence and comput linguist, vol 119. Springer, Berlin
15. Forsythe C, Bernard ML, Goldsmith TE (2006) Cognitive systems: Human cognitive models in systems design. Psychology Press, London
16. Gupta D et al (2016) Study on extrinsic text plagiarism detection techniques and tools. *J Eng Sci Technol Rev* 9(5)
17. Foltýnek T, Ruas T, Scharpf P, Meuschke N, Schubotz M, Grosky W, Gipp B (2020) Detecting machine-obfuscated plagiarism. Springer, In international conference on information, pp 816–827
18. Vrbanc T, Meštrović A (2020) Corpus-based paraphrase detection experiments and review. *Inf* 11(5):241
19. Sharjeel M, Nawab RMA, Rayson P (2017) Counter: corpus of urdu news text reuse. *Lang Resour Eval* 51(3):777–803
20. Muhammad S (2020) Mono-and cross-lingual paraphrased text reuse and extrinsic plagiarism detection. Lancaster University, United Kingdom. Phd Thesis <https://eprints.lancs.ac.uk/id/eprint/145060/>
21. Hafeez H, Muneer I, Sharjeel M, Ashraf MA, Adeel Nawab RM (2023) Urdu short paraphrase detection at sentence level. *ACM Trans Asian Low-Resourc Lang Inf Process* 22(4):1–20
22. Shafi J (2019) An urdu semantic tagger-lexicons, corpora, methods and tools. Lancaster University, United Kingdom. Phd Thesis <https://eprints.lancs.ac.uk/id/eprint/140746/>
23. Iqbal HR, Maqsood R, Raza AA, Hassan SU (2023) Urdu paraphrase detection: A novel dnn-based implementation using a semi-automatically generated corpus. *Nat Lang Eng* 1–31
24. Grave E, Bojanowski P, Gupta P, Joulin A, Mikolov T (2018) Learning word vectors for 157 languages. In proceedings of the international conference on language resources and evaluation (LREC)
25. Peters ME, Neumann M, Iyyer M, Gardner M, Clark C, Lee K, Zettlemoyer L (2018) Deep contextualized word representations. In proceedings of the 2018 conference of the north american chapter of the association for comput linguist: Human language technologies. (Long Papers), vol 1. pp 2227–2237
26. Devlin J, Chang MW, Lee K, Toutanova K (2019) Bert: Pre-training of deep bidirectional transformers for language understanding. In proceedings of the 2019 conference of the north american chapter of the association for comput linguist: Human language technologies. (Long and Short Papers), vol 1. pp 4171–4186
27. Liu Y, Ott M, Goyal N, Du J, Joshi M, Chen D, Levy O, Lewis M, Zettlemoyer L, Stoyanov V (2019) Roberta: A robustly optimized BERT pretraining approach. *CoRR abs/1907.11692*
28. Reimers N, Gurevych I (2020) Making monolingual sentence embeddings multilingual using knowledge distillation. In proceedings of the 2020 conference on empirical methods in natural language processing, pp 4512–4525
29. Radford A, Narasimhan K, Salimans T, Sutskever I et al (2018) Improving language understanding by generative pre-training
30. Khalil M, Er E (2023) Will chatgpt get you caught? rethinking of plagiarism detection. Springer, In international conference on human-computer interaction, pp 475–487
31. Bin-Nashwan SA, Sadallah M, Bouteraa M (2023) Use of chatgpt in academia: Academic integrity hangs in the balance. *Technol Soc* 75:102370
32. Vrbanc T, Meštrović A (2023) Comparison study of unsupervised paraphrase detection: Deep learning—the key for semantic similarity detection. *Expert Syst* 40(9):13386
33. Naveed H, Khan AU, Qiu S, Saqib M, Anwar S, Usman M, Akhtar N, Barnes N, Mian A (2023) A comprehensive overview of large language models. *arXiv preprint arXiv:2307.06435*
34. Reimers N, Gurevych I () Sentence-bert: Sentence embeddings using siamese bert-networks. In proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing, pp 3982–3992
35. Alam F, Chowdhury SA, Boughorbel S, Hasanain M (2024) LLMs for low resource languages in multilingual, multimodal and dialectal settings. In proceedings of the 18th conference of the european chapter of the association for comput linguist: Tutorial abstracts, pp 27–33
36. Duong L (2017) Natural language processing for resource-poor languages. University of Melbourne
37. Brown T, Mann B, Ryder N, Subbiah M, Kaplan JD, Dhariwal P, Neelakantan A, Shyam P, Sastry G, Askell A et al (2020) Language models are few-shot learners. *Adv Neural Inf Process Syst* 33:1877–1901
38. Artexe M, Schwenk H (2019) Massively multilingual sentence embeddings for zero-shot cross-lingual transfer and beyond. *Trans Assoc Comput Linguist* 7:597–610
39. Yang Y, Cer D, Ahmad A, Guo M, Law J, Constant N, Abrego GH, Yuan S, Tar C, Sung YH et al (2020) Multilingual universal sentence encoder for semantic retrieval. In proceedings of the 58th annual meeting of the association for computational linguistics: System demonstrations, pp 87–94
40. Feng F, Yang Y, Cer D, Arivazhagan N, Wang W (2020) Language-agnostic BERT sentence embedding. *CoRR abs:2007.01852* (2020)

41. Sharjeel M, Rayson P, Nawab RMA (2016) Uppc-urdu paraphrase plagiarism corpus. In proceedings of the 10th international conference on language resources and evaluation (LREC), pp 1832–1836
42. Alzahrani SM, Salim N, Abraham A (2011) Understanding plagiarism linguistic patterns, textual features, and detection methods. *IEEE Trans Syst Man Cybern Part C (Appl Rev)* 42(2):133–149
43. Eisa TAE, Salim N, Alzahrani S (2015) Existing plagiarism detection techniques. *Online Information Review*
44. Cedeno AB et al (2013) On the mono-and cross-language detection of text re-use and plagiarism. *Procesamiento del Lenguaje Natural* 50:103–105
45. Sameen S, Sharjeel M, Nawab RMA, Rayson P, Muneer I (2017) Measuring short text reuse for the urdu language. *IEEE Access* 6:7412–7421
46. Kim Y, Jernite Y, Sontag D, Rush AM (2015) Character-aware neural language models. In proceedings of the AAAI conference on artificial intelligence
47. Wang Z, Hamza W, Florian R (2017) Bilateral multi-perspective matching for natural language sentences. In proceedings of the 26th international joint conference on artificial intelligence, pp 4144–4150
48. Cer D, Diab M, Agirre E, Lopez-Gazpio I, Specia L (2017) Semeval-2017 task 1: Semantic textual similarity-multilingual and cross-lingual focused evaluation. In proceedings of the 11th international workshop on semantic evaluation (SemEval-2017), pp 1–14
49. Chandrasekaran D, Mago V (2020) Evolution of semantic similarity-a survey. *ACM Comput Surv* 54
50. Agarwal B, Ramampiaro H, Langseth H, Ruocco M (2018) A deep network model for paraphrase detection in short text messages. *Inf Process Manag* 54(6):922–937
51. El Mostafa Hambi FB (2020) A new online plagiarism detection system based on deep learning. *Int J Adv Comput Sci Appl* 11(9):470–478
52. Mahmoud A, Zrigui M (2022) Siamese arabert-1stm model based approach for arabic paraphrase detection. In proceedings of the 36th pacific asia conference on language, information and computation, pp 545–553
53. Hamza A, En-Nahnahi N, Ouatik SEA (2020) Contextual word representation and deep neural networks-based method for arabic question classification. *Adv Sci Technol Eng Syst J* 5(5):478–484
54. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser L, Polosukhin I (2017) Attention is all you need. In proceedings of the 31st international conference on neural information processing systems, pp 6000–6010
55. Conneau A, Khandelwal K, Goyal N, Chaudhary V, Wenzek G, Guzmán F, Grave E, Ott M, Zettlemoyer L, Stoyanov V (2019) Unsupervised cross-lingual representation learning at scale. In proceedings of the 58th annual meeting of the association for computational linguistics, pp 8440–8451
56. Che W, Liu Y, Wang Y, Zheng B, Liu T (2018) Towards better UD parsing: Deep contextualized word embeddings, ensemble, and treebank concatenation. In proceedings of the CoNLL 2018 shared task: Multilingual parsing from raw text to universal dependencies, pp 55–64
57. Qasmi NH, Zia HB, Athar A, Raza AA (2020) Simplifyur: Unsupervised lexical text simplification for urdu. In proceedings of the 12th language resources evaluation conference, pp 3484–3489
58. Goldberg Y, Hirst G (2017) Neural network methods in natural language processing. *morgan & claypool publishers*(2017). 9781627052986 (zitiert auf Seite 69)
59. Goldberg Y (2016) A primer on neural network models for natural language processing. *J Artif Intell Res* 57:345–420
60. Chicco D (2021) Siamese neural networks: An overview. *Artif Neural Netw* 73–94
61. Nair V, Hinton GE (2010) Rectified linear units improve restricted boltzmann machines. In *ICML*
62. Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R (2014) Dropout: a simple way to prevent neural networks from overfitting. *J Mach Learn Res* 15(1):1929–1958
63. Hochreiter S, Schmidhuber J (1997) Long short-term memory. *Neural Comput* 9(8):1735–1780
64. Kingma DP, Ba J (2015) Adam: A method for stochastic optimization. In 3rd international conference on learning representations
65. Han J, Moraga C (1995) The influence of the sigmoid function parameters on the speed of backpropagation learning. Springer, In international workshop on artificial neural networks, pp 195–201
66. Mikolov T, Chen K, Corrado GG, Dean J (2013) Efficient estimation of word representations in vector space. In *ICLR*
67. Haider S (2018) Urdu word embeddings. In proceedings of the 11th international conference on language resources and evaluation
68. Fares M, Kutuzov A, Oepen S, Velldal E (2017) Word vectors, reuse, and replicability: Towards a community repository of large-text resources. In proceedings of the 21st nordic conference on computational linguistics, pp 271–276
69. Tiedemann J (2012) Parallel data, tools and interfaces in opus. *Lrec* 2012:2214–2218
70. Sanh V, Debut L, Chaumond J, Wolf T (2019) Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *CoRR abs:1910.01108*

71. Wise MJ (1996) Yap3: Improved detection of similarities in computer program and other texts. In proceedings of the 27th sigcse technical symposium on computer science education, pp 130–134

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Springer Nature or its licensor (e.g. a society or other partner) holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.

Authors and Affiliations

Hafiz Rizwan Iqbal¹ · Muhammad Sharjeel² · Jawad Shafi² · Usama Mehmood¹ · Saeed Ul Hassan¹ · Agha Ali Raza³

✉ Hafiz Rizwan Iqbal
rizwan.iqbal@itu.edu.pk

Muhammad Sharjeel
muhammadsharjeel@cuilahore.edu.pk

Jawad Shafi
jawadshafi@cuilahore.edu.pk

Usama Mehmood
usamamehmood@itu.edu.pk

Saeed Ul Hassan
saeed-ul-hassan@itu.edu.pk

Agha Ali Raza
agha.ali.raza@lums.edu.pk

¹ Department of Computer Science, Information Technology University, Arfa Software Technology , 346-B, Ferozepur Road, Lahore, Pakistan

² Department of Computer Science, COMSATS University Islamabad (CUI), Lahore Campus, Defence Road, Lahore, Pakistan

³ School of Science and Engineering, Lahore University of Management Sciences (LUMS), DHA Phase 5, Opposite Sector U, Lahore, Pakistan