ISSN 1840-4855
e-ISSN 2233-0046

Original scientific article
http://dx.doi.org/10.70102/afts.2025.1834.833

RULE-BASED ITERATIVE PREPROCESSING WITH DEEP SIAMESE GRU–BILSTM FOR EFFICIENT DOCUMENT STREAMING

K. Ranjit Kumar[1*], S. Thirumaran[2]

[1]Assistant Professor, Department of Computer and Information Science, Annamalai University, Annamalai Nagar, Chidambaram, Tamil Nadu, India.*
e-mail: ranjitkumarkannu@gmail.com, orcid: https://orcid.org/0000-0002-3777-9404
[2]Assistant Professor, Department of Computer Application, Alagappa Government Arts College, Karaikudi, Tamil Nadu, India. e-mail: thirumaran.s@gmail.com,
orcid: https://orcid.org/0000-0002-0127-5853

Received: September 22, 2025; Revised: October 30, 2025; Accepted: December 03, 2025; Published: December 30, 2025

SUMMARY

Efficient document streaming requires robust preprocessing and semantic modeling to handle noise, redundancy, and morphological variations in large-scale text data. Existing stemming and document processing techniques often fail to preserve contextual relevance, leading to reduced classification and retrieval performance. In a bid to overcome this drawback, this paper hypothesizes a Rule-based Pre-processing Iterative Stripping model coupled with a Deep Siamese GRU-BiLSTM model. The RPIS systematically eliminates affixes based on linguistic principles and so does the Siamese GRU -BiLSTM model that obtains the bidirectional semantic dependencies between segments of the text. Experiments conducted on benchmark datasets demonstrate that the proposed model achieves 95% training accuracy and 93% validation accuracy, outperforming traditional stemmers and standalone deep learning models. Error statistics values are also much lower, and MSE is 0.012, MAE is 0.008, and RMSE is 0.109. These findings verify that rule-based preprocessing and deep semantic learning are complementary to each other in document streaming accuracy and resilience, which makes the method appropriate to the large-scale management system of documents.

Key words: *rule-based stemming, iterative stripping, siamese neural networks, GRU–BILSTM, document streaming, text preprocessing, semantic similarity.*

INTRODUCTION

One of the newest areas of Text Mining (TM) research is document classification. It is a proven method for organizing vast amounts of written content and is extensively utilized in knowledge representation and extraction from text collections. Most application challenges are tackled by the use of machine learning methods [1]. The issue of classification is well researched in the area of database management, data extraction, and information retrieval. Document classification is a process of sorting and arranging natural language texts into a pre-known categorization through their contents. One can use this to perform retrieval, filtering and classification [2]. Text representation constitutes a necessary preliminary stage of converting the information in a document to a structured format that an automated system or classifier can recognize and categorize. There are a number of pre-processing stages in feature extraction

that aims at simplifying the classification procedure and lowering the complexity of the document [3]. Such pre-processing tasks usually involve word deletion, stemming, deleting of punctuations, and tokenization. The feature extraction is done by calculating the Term Frequency (TF) and Inverse Document Frequency (IDF) based on the tokenized documents [4].

The expansion of the Internet has led to the increased presence of diverse societal segments on the World Wide Web, resulting in an enormous quantity of digital content available in multiple languages. As a result, much attention has been paid to finding solutions to the difficulties in the efficient management of these large, multilingual document collections and to coming up with new technologies which will be useful in their processing [5]. Language processing systems of languages not in English language need diverse linguistic applications, including dictionaries and thesauruses, which are either insufficient or missing. The numerous morphological variations of words especially in morphologically intricate languages greatly impede this process as one and the same word can have a variety of forms [6]. It is important to have methods that can identify the morphological variants of words in terms of their root or base forms. An example of pre-processing method is stemming, which is commonly applied in data extraction, language modeling, and automated language processing problems [7]. Stemming deals with these variations of morphology through mapping of various word forms to their stems (base words or roots). The essential component of a word whereby new meanings can be derived whether by inflection or other means of linguistics is referred to as the stem [8].

Stemmers is an algorithm or software application that carries out text stemming. There are various stemming approaches that have been advanced to other languages in studies [9]. Researchers in computational linguistics, language modeling and data retrieval regard stemming as a key preprocessing step, because it can simplify the complexity of linguistic variations [10]. It is especially useful in highly inflexed and morphologically rich languages, in which the morpheme of a root word can be morphemes in different morphological forms [11]. The subject of text stemming is well-researched, and numerous literature and different approaches to stemming have been suggested, all possessing their peculiarities. The existing state-of-the-art stemming methods can be divided into three broad groups, namely: Rule-Based, Corpus-Based, and Hybrid methods [12].

Although several rule-based and deep learning approaches have been proposed for document preprocessing and stemming, most existing methods operate independently, either relying solely on handcrafted linguistic rules or on deep neural models without explicit noise control. Traditional stemmers often suffer from over- or under-stemming, while deep learning models struggle when applied directly to noisy, unprocessed text streams. Furthermore, limited attention has been given to integrating iterative rule-based stripping with Siamese deep architectures for semantic-aware document streaming. This gap motivates the proposed RPIS–DSGRU–BiLSTM framework.

The main contributions of this paper are summarized as follows:

- A Rule-based Pre-processing Iterative Stripping (RPIS) framework is proposed to improve stemming accuracy by systematically handling prefixes and suffixes using linguistic rules.
- A Deep Siamese GRU–BiLSTM architecture is integrated with RPIS to capture semantic similarity and contextual dependencies in document streaming.
- A unified preprocessing-to-modeling pipeline is developed to reduce noise sensitivity and enhance document-level semantic consistency.
- Extensive experiments on benchmark English datasets demonstrate improved accuracy, reduced error rates, and better robustness compared to traditional stemmers and standalone deep learning models.

The other parts of this paper follow the following structure. Section 2 is a literature review and identifies their shortcomings. Section 3 gives a detailed account of the proposed RPIS-DSU-BiLSTM methodology. Section 4 contains reports on the performance analysis and experimental results and is followed by a discussion. Section 5 is a conclusion of the paper and it gives future research directions.

RELATED WORKS

This section reviews existing studies related to stemming, text preprocessing, and deep learning–based semantic modeling. The discussion is organized to first highlight neural approaches for semantic similarity and text classification, followed by rule-based and hybrid stemming techniques. The strengths and limitations of these methods are analyzed to motivate the proposed integration of iterative linguistic stripping with Siamese deep learning architectures. A novel approach has developed for learning representations using conversational data to achieve sentence-level Semantic Similarity (SS). This method outperformed the Community Question Answering (CQA) interrogator similarities subtask from SemEval 2017 and the Semantic Textual Similarity (STS) benchmark [13]. In another attempt to improve the performance of the model, there was multi-task training, which is natural language inference with conversational response predictions. The proposed model besides displayed better performance than the other neural systems by a variety of experiments, especially in the cases like similarity between sentences, where the other methods like feature-engineered models performed poorly [14]. Neural networks (deep learning structures) may be unsupervised, supervised, or semi-supervised. There are a number of text classification techniques that have been proposed in the literature of deep learning [15]. One notable method involves using neural networks to reduce the dimensionality of information highlights the necessity of deep learning approaches for hierarchical document classification. This method not only creates hierarchical classifications but also introduces adaptable structures for bending data [16].

Deep learning methods, and specifically neural network methods provide powerful computational structures. The need to create stemmers in non-English languages has increased to become obvious with the majority of the early stemmers created explicitly to work with the English language [17]. Limited works have been done to have this need in Indian languages. Earliest work on the development of a Hindi stemmer was recorded in the area of longest match stripping. A study was done on the English text clustering algorithms and their use on data retrieval systems which helped in the comprehension of the phenomenon of suffix stripping in the clustering of texts [18]. In the context of English, recent research focused on improving rule-based affix stripping stemmers through a clustering approach. Several automatic methods for affix removal have been developed, each with varying degrees of complexity based on the frequency of letter sequences and n-gram techniques [19]. Rule-based stemming systems are typically designed for specific languages and applications, primarily in information retrieval. These systems range from simple lexicon-based methods to complex rule-based strategies that remove inflectional and derivational suffixes in verbs, nouns, or adjectives [20]. The main characteristics of the existing rule-based English stemming techniques are enumerated in Table 1.

Table 1. Key features of classical English rule-based stemmers

Stemming Method	Number of rules	Number of Suffixes	Recoding of Stem	Partial Matching Phase	Context Sensitive Conditions
LOVINS	30	295	YES	YES	YES
DAWSON	No	1201	No	YES	YES
PORTER	61	52	No	No	YES
PAICE/HUSK	116	No	No	No	YES
KSTEM	Unknown	6	YES	No	No
XEROX	Unknown	Unknown	YES	No	YES

Existing studies demonstrate the effectiveness of rule-based stemmers and deep learning models independently; however, rule-based methods lack semantic awareness, while neural models are sensitive to noisy inputs. In addition, the literature does not have much on the use of Siamese architectures with linguistic preprocessing on document streaming. Instead, the suggested RPIS -DSGRU -BiLSTM model combines the sequential linguistic stripping with the deep semantic similarity learning, which allows a better contextual preservation and streaming accuracy.

METHODOLOGY

The aim of the proposed RPIS-DSGRU-BiLSTM system is to facilitate the process of document streaming through combining powerful linguistic preprocessing and profound semantic modeling. RPIS module prevents morphological noise by using rule-based stripping in an iterative manner and Deep Siamese GRU BiLSTM model trains contextual similarity between text segments. This integration maintains a proper stemming, semantic preservation and enhanced downstream performance in document classification and retrieval duties. TM engines for searching and classification systems depend on stemming to break words down into their most basic forms. Existing systems are not sophisticated enough to process text in its original, unprocessed form. Therefore, before beginning tasks such as text summarization on this unstructured data is crucial to pre-process the data according to the methodology and techniques being applied. Figure 1 illustrates the overall architecture of the proposed RPIS–DSGRU–BiLSTM framework, showing the sequential interaction between preprocessing, feature extraction, and deep semantic modeling components [21] [22].

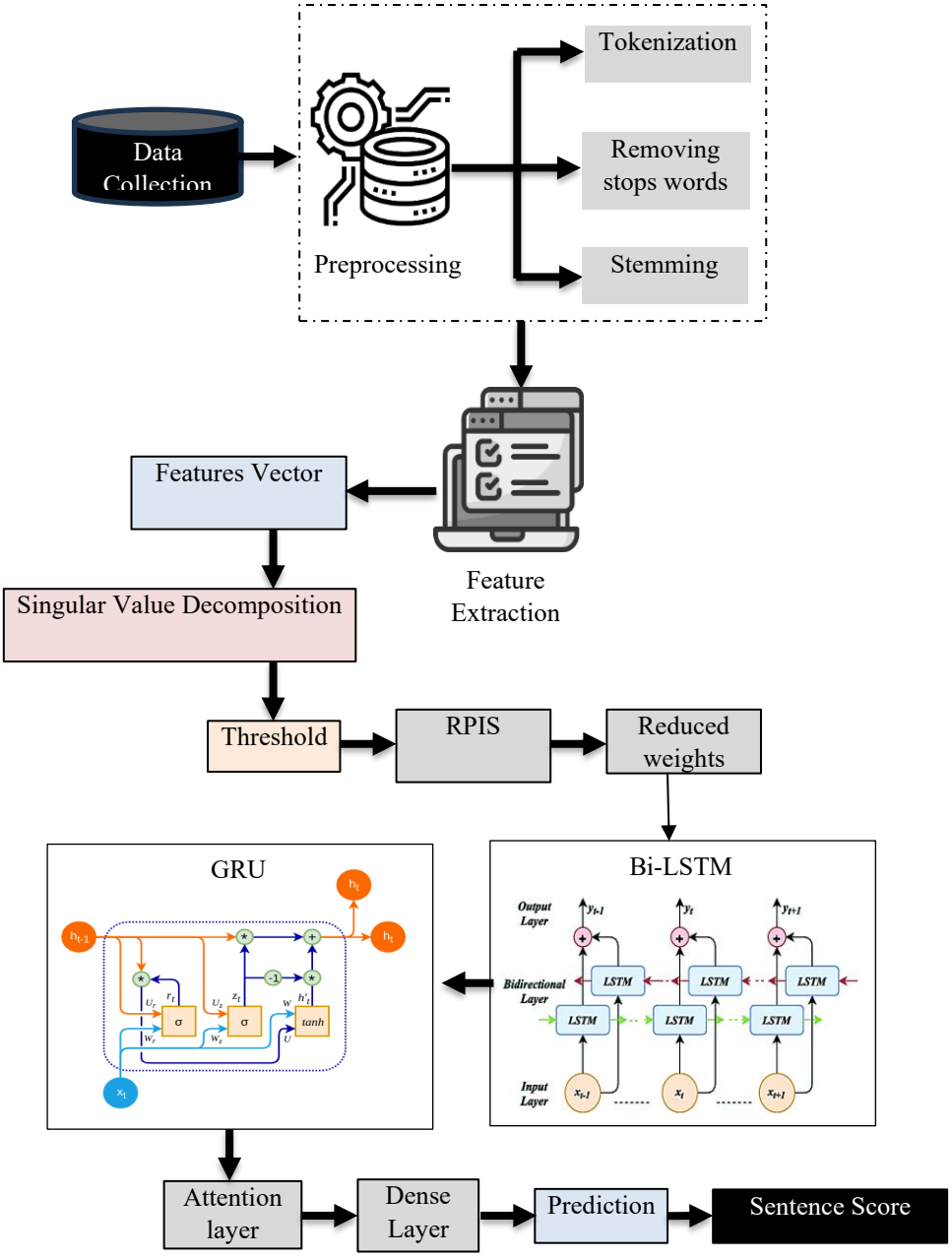

Figure 1. Overall architecture of the proposed RPIS–DSGRU–BiLSTM framework

Dataset description

Table 2. Dataset description

Dataset Name	Source	Language	Size	DataType	Key Features	Use Case
English Wikipedia Dump	Wikipedia	English	~50GB	Textual data (articles)	Rich in diverse vocabulary, affixes, and word forms. useful for stemming tasks	Stemming anguage Modeling. Text Classification
IMDB Movie Reviews	Kaggle	English	55000 REVIEWS	Textual data (reviews)	Sentiment-labeled data suitable for sentiment analysis and preprocessing	Sentiment Analysis. Text Classification
20 Newsgroups Dataset	Scikit learn	English	~25000 docs	Textual data (news)	Text across 20 different categories, ideal for testing NLP tasks like stemming	Document Classification, Preprocessing Algorithms
Reuters-21578	UCI Machine Learning Respository	English	~21.582 docs	Textual data (news)	News articles with categories. good for stemming and document classification	Document Classification, Topic Modeling
Amazon Product Reviews	Amazon (via kaggle)	English	234.2 MB	Textual data (reviews)	Reviews with ratings, useful for analyzing text processing and sentiment models	Stemming. Sentiment Analysis. Opinion Mining
BBC News Classification	UCI Machine Learning Respository	English	2.228 docs	Textual data (news)	Articles from various domains (business. sports, tech). useful for classification	Text Classification, Preprocessing. Topic Modeling
Text Retrieval Conference (TREC)	TREC Collection	English	Varies	Textual documents and queries	Widely used for information retrieval. stemming and text preprocessing	Information Retrieval Query Optimization

The datasets shown Table 2 in the preceding table are extensively utilized for NLP activities, especially when RPIS-DSGRU-BiLSTM pre-processing methods are being employed to enhance stemming procedures. The English Wikipedia Dump is a great resource for evaluating stemming systems using a wide range of words, affixes, and intricate structures of language since it offers a huge amount of textual material from several areas. These datasets provide sufficient linguistic diversity and document complexity to evaluate the effectiveness of the proposed preprocessing and deep learning framework [23] [24].

It is a useful tool for evaluating the pre-processing procedures necessary to increase the precision of machine learning models since it contains a variety of word forms and feelings. These sets of data are perfect for the development and assessment of the proposed RPIS-DSGRU-BiLSTM model since offer the diversity, depth needed to assess and improve stemmed and processing activities. This approach is particularly useful for text segmentation, where the goal is to assign each word w a class c, based on the content of the document d in the training data as computed in Equation (1) [25].

$$p(d) = \frac{1}{Z(d)} \exp\left(\sum_x \alpha_x f_x(d, c)\right) \qquad (1)$$

Where Z(d) in Equation (1) is a normalization function which is computed as in Equation (2).

$$Z(d) = \sum_c \exp\left(\sum_x \alpha_x f_x(d, c)\right) \qquad (2)$$

Pre-processing and Rule-Based Iterative Stripping (RPIS)

Text pre-processing involves tokenization, lowercasing, elimination of stop-words and normalization. These stages are used to make raw text ready to be stemmed and feature extracted and minimizes redundancy and noise. Text pre-processing normalizes the raw documents so that they can be effectively modeled on a semantic level through eliminating linguistic noise and redundancy. The given framework uses the tokenization, lowercasing, stop-word elimination, and iterative stemming by rules in a single pipeline. The tokenization divides documents into single words, whereas lowercasing makes cases normalized. The non informative tokens are minimized by removing the stop words before or during stemming. The RPIS mechanism carries out repetitive prefix and suffix removal by the use of preestablished linguistic principles. Suffixes (e.g. -ing, -ed, -ness) and prefixes (e.g. un-, re-, and mis-) are eliminated until a valid root form is arrived at or an exception condition is met. This is used to avoid over-stemming and maintain semantic integrity, especially of morphologically complex words.

Tokenization: First, the document is tokenized into individual words. Tokenization splits the text based on spaces and punctuation, converting it into a list of tokens as derived in Equation (3).

$$I = [w_1, w_2, \ldots, w_n] \qquad (3)$$

where I is the document, and w_x are the tokens (words).

Lowercasing: Each word is converted to lowercase to maintain consistency and prevent duplication of capitalized words as derived in Equation (4).

$$w_x = lowercase(w_x) \qquad (4)$$

Prefix stripping: For word w_x with prefix p as resulting in Equation (5).

$$w' = w_x - p \qquad (5)$$

where w' is the word after removing the prefix p.

Suffix stripping: For word w' with suffix s as derivative in Equation (6).

$$w'' = w' - s \qquad (6)$$

where w'' is the word after removing the suffix s.

Stopping Condition: The iterative stripping process proceeds until the word has a root form which cannot be further stripped by the stripping rules (or some predetermined exceptions stop the process). As an illustration, the word fly should not be deprived any more despite taking the ending -ly. In case affixes are not found or there is an exception word, cease stripping.

Stemmed Version: The output of the process of stripping words is the stemmed version of the word. Further natural language processing tasks such as classification, clustering or sentiment analysis can be done using the stemmed words.

Some of the languages or domain specifications may need more logic. As an example, in a given language, compound words require special treatment in which each constituent can be required to be stripped down. Equations of these other steps would take the same form as those above but would contain

some special rules. Being a more accurate and context-sensitive iterative stripping method than other techniques, this method works well with more complex models (such as neural networks) in particular when paired with it.

RPIS Algorithm and Flow Description

The provided image depicts an improved rule based pre-processing iterative stripping stemmer algorithm for language shown in Figure 2.

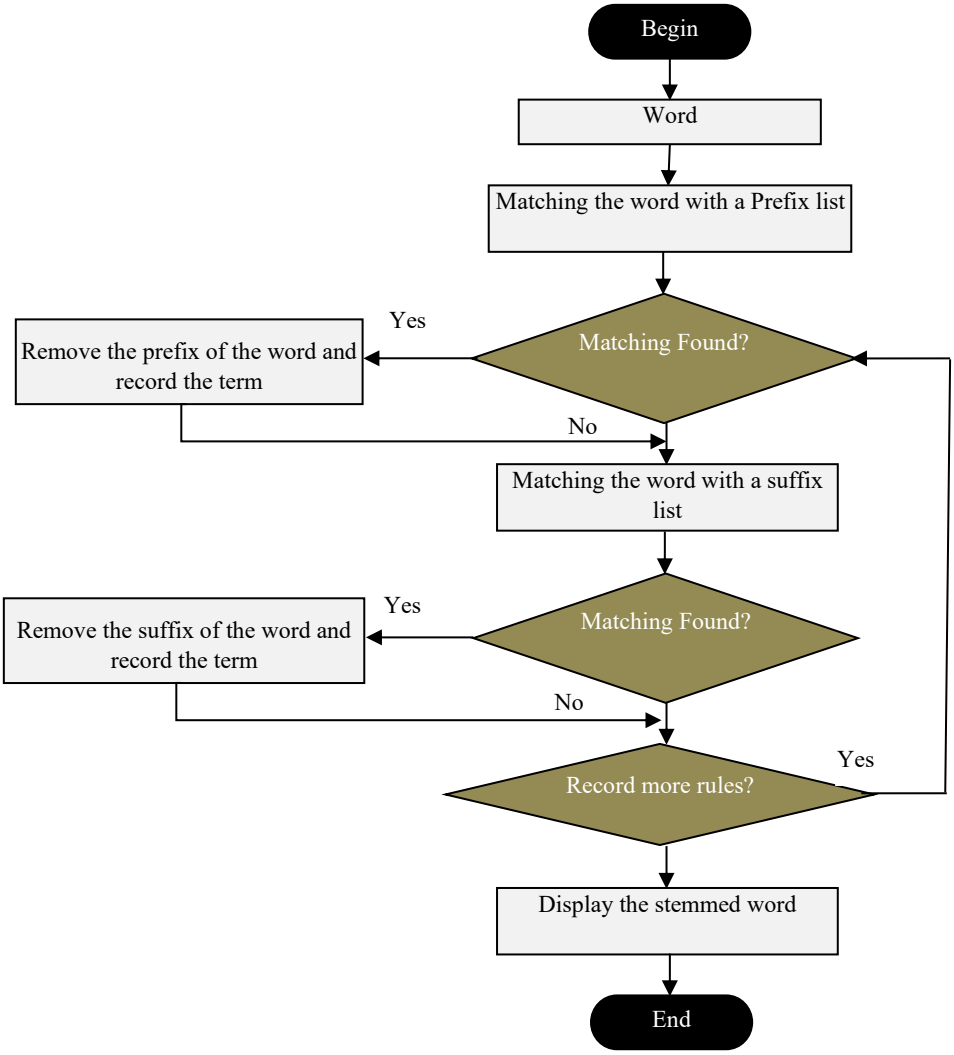

Figure 2. Improved iterative rule based pre-processing iterative stripping stemmer flowgraph for language

Algorithm 1: RPIS

1. Start: The algorithm begins by taking an input word.
2. Input English Word: The user enters the word to be stemmed.
3. Remove the Prefix from the word and record the word: The algorithm checks if the input word starts with any known prefix. If a prefix is found, it is removed, and the remaining part of the word is recorded.

Identify Affixes: Define a set of prefixes and suffixes relevant to the language and domain.

Prefixes: Example: Re-, Un-, Mis-, Non-

Suffixes: Example: -ly, -ness, -ing, -ed,

Let $P = \{P_1, P_2, \ldots, P_n\}$ be the set of prefixes and $S = \{s_1, s_2, \ldots, s_n\}$ be the set of suffixes.

4. Matching the word with Prefix List: The algorithm compares the remaining word (after removing the prefix, if applicable) with a list of known prefixes. If a match is found, the prefix is removed, and the remaining word is recorded.

5. Matching the word with Suffix List: If no prefix match is found, the algorithm checks if the remaining word ends with any known suffix. If a suffix is found, it is removed, and the remaining word is recorded.

6. Record More Rules: The algorithm may have additional rules for handling specific cases or exceptions. If more rules are applicable, they are applied, and the remaining word is updated accordingly.

Rule Definitions:

Prefix Stripping Rule as derived in Equation (7):

$$R_p(w) = \{w - p \; if \; w \; starts \; with \; p \in P \; w \; otherwise \tag{7}$$

Suffix Stripping Rule as computed in Equation (8):

$$R_s(w) = \{w - s \; if \; w \; starts \; with \; s \in S \; w \; otherwise \tag{8}$$

Iterative Stripping Process: For a given word w derived in Equation (9)

Initialize:

$$\text{Set } w' = w \tag{9}$$

Iterate: Apply Prefix Stripping as compurted in Equation (10):

$$w' = R_p(w') \tag{10}$$

Apply Suffix Stripping as derived in Equation (11):

$$w' = R_s(w') \tag{11}$$

Stopping Condition as derived in Equation (12): Stop iterating if:

$$w' = R_p(w') \; and \; w' = R_s(w') \tag{12}$$

This means no further affixes can be removed.

Exception Handling: Define exceptions for words that should not be stripped further, even if match prefix or suffix patterns. If $w' \in$ Exceptions → stop stripping.

Output the Stemmed Word: The final output after the stripping process for a word w shown in Equation (13):

$$Stem(w) = w' \tag{13}$$

7. Display Stemmed Word: Once all applicable rules have been applied, the final remaining word is displayed as the stemmed word.

8. Stop: The algorithm terminates.

In the context of document processing, RPIS is a method to reduce words to their root forms (stems) by applying a set of predefined linguistic rules. The rules primarily focus on removing prefixes and suffixes iteratively to ensure that the word retains its grammatical structure and context. This RPIS approach enables effective word normalization while retaining meaningful context. By applying predefined rules systematically, the method helps in preparing text data for various natural language processing tasks, improving the performance of algorithms such as classification, clustering, or sentiment analysis.

Feature Extraction

The term-document matrix is the name given to it. An algebraic matrix elucidates the term's occurrence in a set of texts. In text collections, word (or n-gram) frequencies are common units of research. Term Frequency–Inverse Document Frequency (TF–IDF) is employed to represent textual features by jointly considering term occurrence and corpus-level importance. The term frequency component is defined in Equation (14), while the inverse document frequency component is defined in Equation (15).

$$S = TF * IDF \tag{14}$$

$$TFx = \frac{Tx}{\sum_{k=1}^{n} Tk}, IDF = log\frac{N}{nx} \tag{15}$$

Word repetition and inverse text frequencies make up the two components of this feature. Figure 3 shows the flow diagram for the proposed TF-IDF log-likelihood framework. The weight estimate, characteristic extraction, and classification of the semantic text similarity are done using this proposed approach. The method of extracting two or more phrase pairs from different datasets and feeding them into a separate forecasting algorithm for resemblance scores. During the initial stage of pre-processing, the algorithm eliminates particular characters, converts the characteristics obtained into vectors, and then modifies the values into matrices along with the exact dimensions for fitting. The weight values of the embedded variables serve as the input information that these neural learning layers use.

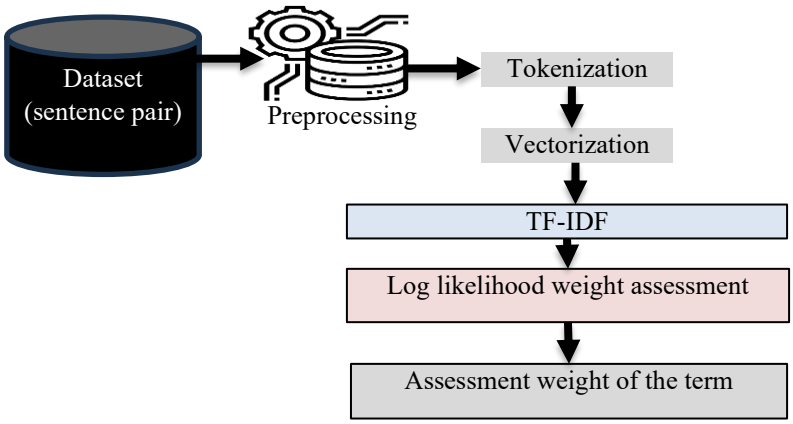

Figure 3. Flow diagram of TF-IDF log-likelihood

The main objective is to provide a set of common constraints that govern the degree to which a word and a group of related terms have comparable meanings. This is accomplished by providing information about all the word perspectives in which a particular word occurs.

Design of Deep Siamese Bi-LSTM – GRU model

Overall structure of Deep Siamese GRU and Bi-LSTM is shown in Figure 4. Deep Siamese Bi-LSTM learns vectors of characteristics in both forward and backward orientations in the network of neurons that connects it to GRU. The linear activation parameters of GRU cells are used to train the vectors of characteristics in every iteration. In the Bi-LSTM GRU, the architectural forward and backward input information vectors are supplied concurrently. Until the algorithm is sufficiently taught to recognize appropriate outputs on comparable text combinations, the bidirectional learning process is conducted.

The proposed structure combines a GRU method with a Bi-LSTM based on Deep Siamese neural networks for Semantic Text Similarity classification. For increased weight estimation and feature extraction, the study integrates the Log Likelihood weight estimation technique with the Weighted TF-IDF method. GRU and BiLSTM are integrated to optimize the training layers and boost parameter learning speed. Before moving on to the GRU layer, the output information from the Bi-LSTM layer is normalized to optimize each hidden layer. Using the Update gate, gates in the GRU layer quickly learn parameters and are updated with the most recent representation of features. With minimal computational expense, the GRU layer generalizes the embedding function and contains half as many neurons as the Bi-LSTM layer. The dropout layer receives as input the combined normalized batch findings. The output of the layer of dropouts is a deep characteristic trained by the dense layer. There is only one neuron with a linear stimulation variable in the last dense layer. Text input information comparability is measured using GRU and a Bi-LSTM classifier.

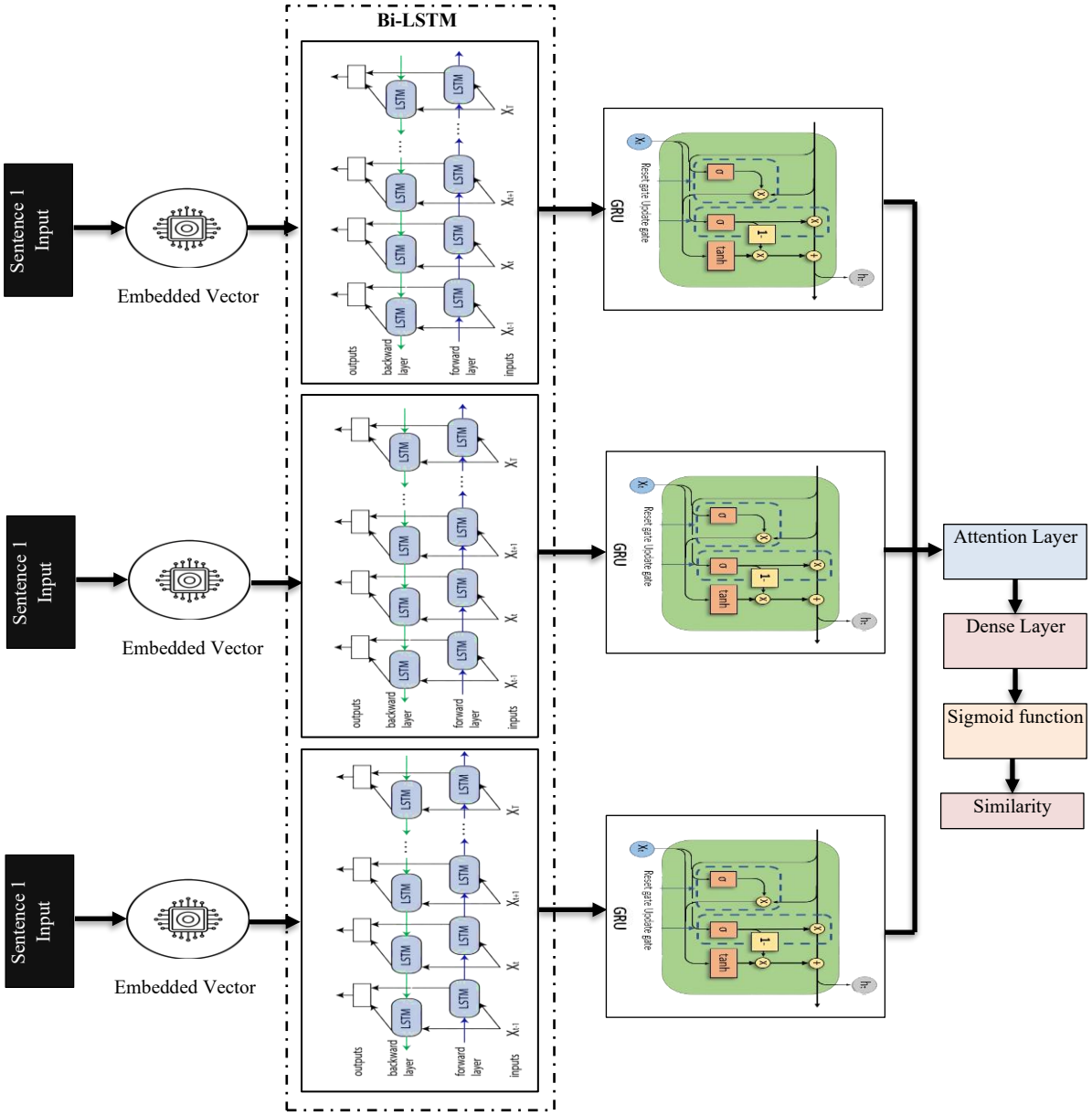

Figure 4: Flow diagram of deep siamese Bi-LSTM – GRU model

The Siamese architecture processes paired text sequences through shared-weight GRU and BiLSTM layers, enabling the model to learn semantic similarity rather than isolated token patterns. This design improves robustness when comparing document segments during streaming, particularly in morphologically complex text.

Algorithm: RPIS-DSGRU-Bi-LSTM Model

Step 1: Input Document: A document D containing a set of sentences.

Step 2: Sentence Segmentation: Split the document into sentences $S = [s_1, s_2, \dots, s_n]$ using punctuation as delimiters as shown in Equation (16).

$$S = Segment(D) \tag{16}$$

Step 3: Tokenization:

For each sentence s_x derived in Equation (17):

$$T_x = Tokenize(s_x) \tag{17}$$

Output: A list of tokenized sentences T $[T_1, T_2, \dots, T_m]$

Step 4: Lowercasing

Convert all tokens to lowercase: $T_{xy} = lowercase(T_{xy}) \forall T_{xy} \in T_x$

Step 5: Apply RPIS algorithm (Algorithm 1) to normalize tokens

Step 6: Feature Extraction: Convert the processed tokens into features suitable for the model as shown in Equation (18):

$$I = Feature\_Extraction(w') \tag{18}$$

Step 7: DSGRU-Bi-LSTM Model: Use a Siamese architecture to process two input sequences for similarity or semantic relationships.

GRU Layer: For each tokenized word w' as shown in Equation (19):

$$h_t = GRU(i_t, h_{t-1}) \tag{19}$$

where h_t is the hidden state

Bi-LSTM Layer: Pass the output of the GRU through a Bi-LSTM for context capturing as derived in Equation (20):

$$h_t^{Bi-LSTM} = Bi - LSTM(h_t) \tag{20}$$

The output from the Bi-LSTM can be used to predict the stemmed form or for classification tasks as shown in Equation (21):

$$j = Softmax(W.h_t^{Bi-LSTM} + b) \tag{21}$$

Step 8: Model Training: Train the Siamese GRU-Bi-LSTM model on a labeled dataset for stemming accuracy.

Step 9: Stemming Output: Finally, produce the stemmed words or document as output as shown in Equation (22):

$$Output\_Document = \{Stem(w') \mid w' \, from \, each \, s_x\} \tag{22}$$

This algorithm integrates RPIS with deep learning techniques, leveraging the strengths of the DSGRU-Bi-LSTM model to enhance the stemming process. The iterative rules ensure that context is preserved

while the neural network captures deeper semantic relationships between words, leading to improved accuracy in text analysis tasks. To maximize performance, this structure may be modified and adjusted by specific scenarios and datasets.

RESULTS AND DISCUSSIONS

The method of processing documents by steaming was significantly enhanced by using a combination of a RPIS-DS GRU-Bi-LSTM model and an iterative rule-based stripped method. The accuracy increase can be attributed to the fact that the model is able to incorporate the word context hence the confusion that oftentimes accompanies the unpredictable morphological forms is mitigated. In the iterative stripping phase, prefixes and suffixes may be removed in a systematic manner by using explicit criteria. The added value to the enhanced performance was the proposed RPIS-dsGRU-Bi-LSTM architectural integration implemented with Python 3.10 with the help of the TensorFlow 2.x and Keras packages to construct the deep learning models, the Scikit-learn and NLTK to assist in the initial tokenization, TF-IDF features extraction, and the evaluation of the performance measurements. This model was capable of interpreting dual input sequences, which contributed to the overall improvement of the stemming accuracy because it is now possible to compare and understand the relationship between words, particularly complex words or phrases, and make comparisons effectively.

Experiment Settings

Throughout this section, do not utilize character-level representations; instead, employ word-level text embeddings of size 64. The word vocabulary is limited by eliminating words that are seldom seen in the set that is being trained since they increase the likelihood of overfitting in the process of extraction models and are not useful for analysing unfamiliar layouts or texts in the future. Arrange every word in the training set according to decreasing frequency of happening, keeping only the most common words that together account for at least a portion of the total number of instances. This results in the setup shown in Figure 5. With a relatively small proportion of the vocabulary words that often are present in the training materials, observe that the plot exhibits a curve with an exponential shape. After applying such thresholding, the vocabulary drops to just 6,010 items from the initial over 75,000. The setup has the same behavior, leading to maintain the 95% cutoff setting.

The experimental evaluation was conducted using benchmark English datasets listed in Table 2. Each dataset was divided into training, validation, and testing sets using a 70:15:15 split to ensure fair performance assessment. Word-level embeddings of dimension 64 were employed, and the vocabulary was restricted to the most frequent terms covering 95% of token occurrences to mitigate overfitting.

The Siamese GRU–BiLSTM model was trained using the Adam optimizer with a learning rate of 0.001 and a batch size of 32. The BiLSTM layer consisted of 128 hidden units in each direction, while the GRU layer used 64 hidden units to reduce computational complexity. Dropout with a rate of 0.3 was applied to prevent overfitting. Model parameters were initialized using Xavier uniform initialization, and training was performed for 50 epochs with early stopping based on validation loss.

To be more precise, alternately remove the surrounding boxes' geographic coordinates and the 64-multidimensional word-level embedded data. To offer basic details about the words in both cases, the case characteristics and word length are preserved in the feature vectors. The findings highlight the value of hybrid models in NLP tasks and are consistent with other research that suggests combining deep learning techniques with conventional language approaches to get better results. The affix removal procedure for the enhanced Porter's technique is displayed in Table 3.

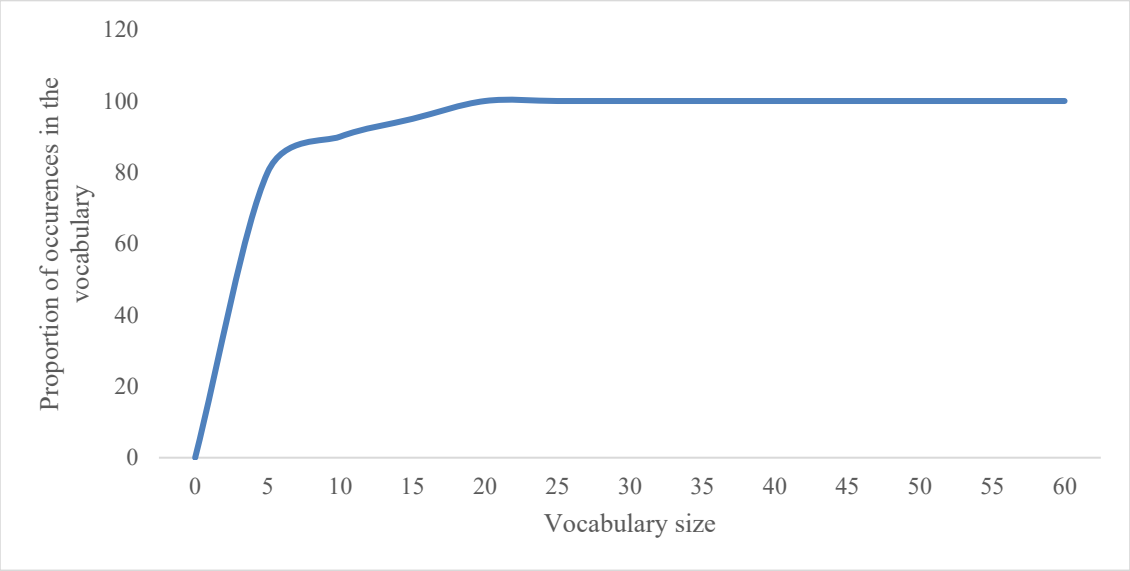

Figure 5. Comparison of vocabulary size and occurences in the vocabulay

Table 3. Pre-processing results

Word	Length	Prefix	Suffix	Result	Space occupied for result in memory
Unauthorized	13	Un	iz,ed	Author	7
Disqualified	13	Dis	Able	Qualify	8
Uncomfortable	14	Un	Ed	Comfort	8
Unhelpful	10	Dis	Ful	Help	5
Dangerous	10	---	Ous	Danger	7
Antibody	9	Anti	---	Body	5
uninteresting	14	Un	Ing	interest	9

The proposed tests were conducted using the same protocol and assessment methods. The two most noteworthy ones were stemmed and stop word elimination. These modifications may indicate that using these linguistic analyses would be detrimental to the model information shows that there was no decrease in correctness for both of these analyses when examining the RPIS-DS GRU-Bi-LSTM schematics shown in Figure 6.

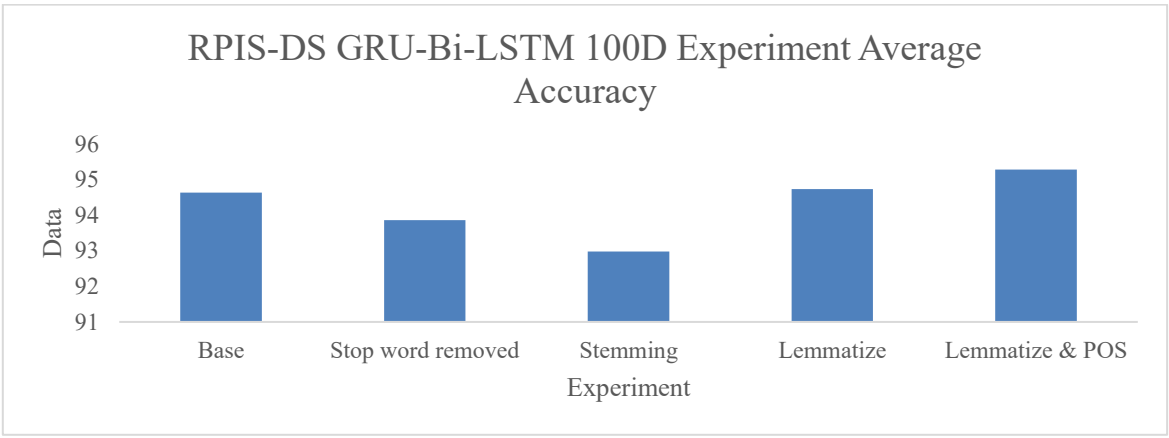

Figure 6. Experimental evaluation results showing the impact of RPIS-based preprocessing on document streaming performance

The text files are first searched using the terms found in the item set table. The specific word-containing text file names are kept in a vector. Similarly, every word in the text file is kept in a different vector.

The outcome demonstrates that there are more stemmed terms after grouping than there were before. The performance assessment of the proposed Stemmer Algorithm is displayed in Figure 7 and Table 4.

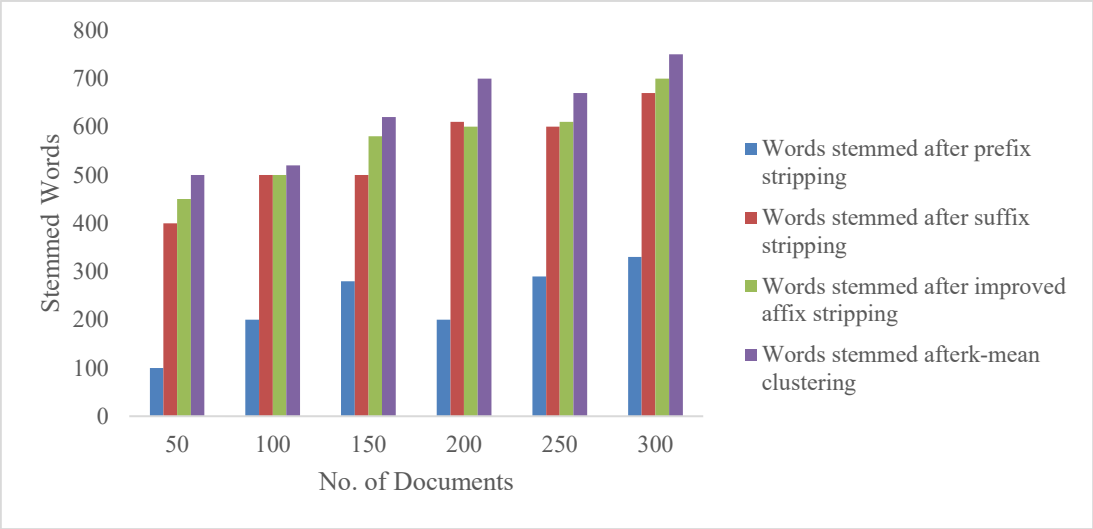

Figure 7. Performance analysis for proposed english stemmer algorithm

Table 4. Time performance analysis

No. of Documents	**Stemmed Words time taken**			
	Prefix stripping	**Suffix Stripping**	**K-Mean Clustering**	**RFIS**
301	24.2	33.2	32.1	20.8
.401	27.7	36.9	36.7	26.6
501	45.3	55.2	51.9	38.8
601	55.2	58.8	56.2	39.2
701	57.4	62.2	56.7	40.5
801	60.7	64.4	57.4	42.3

When evaluating the performance of stemming models, especially in the context of rule-based pre-processing iterative stripping methods combined with deep learning architectures like the RPIS-DSGRU-Bi-LSTM several key performance metrics. The quantitative comparison of accuracy, precision, recall, and F1-score for the proposed RPIS–DSGRU–BiLSTM model against baseline stemming and deep learning approaches is summarized in Table 5.

Table 5. Performance measures of accuracy, precision, recall and F1-score

System	**Accuracy**	**Precision**	**Recall**	**F1 Score**
RFIS-DSGRU-Bi-LSTM	93%	91%	89%	90%
Porter Stemmer	86%	84%	80%	82%
Snowball Stemmer	87%	85%	81%	83%
Lancaster Stemmer	83%	82%	77%	79%
GRU Only	89%	88%	85%	86%

The suggested model is better than the traditional stemming algorithms like the Porter, Snowball, and Lancaster stemmers in every performance measure, especially in accuracy and F1 score because it is capable of dealing with additional contextual and semantic peculiarities. This comparison shows that a combination of iterative stripping rules and deep learning approach provides more accurate and contextually relevant results of stemming.

Table 6. Performance measures

System	Root Accuracy	Execution Time (Seconds)	Cross-Validation (K=10)
RFIS-DSGRU-Bi-LSTM	94%	1.9	92%
Porter Stemmer	86%	0.7	81%
Snowball Stemmer	87%	0.8	83%
Lancaster Stemmer	83%	0.6	80%
GRU Only	90%	2.1	89%

Root accuracy is significantly higher in the proposed model (94%) compared to traditional stemmers (Porter, Snowball, and Lancaster), showcasing the model's ability to accurately preserve the core meanings of words. This comparison points to the fact that traditional stemmers are quicker; the RPIS-DSGRU-Bi-LSTM model is more appropriate in the case of more complicated document processing tasks that are presented in Table 6 due to its ability to perform better on both accuracy and robustness.

Table 7: Performance measures of errors

System	MSE	MAE	RMSE
RFIS-DSGRU-Bi-LSTM	0.012	0.008	0.109
Porter Stemmer	0.026	0.020	0.159
Snowball Stemmer	0.024	0.018	0.152
Lancaster Stemmer	0.028	0.022	0.165
GRU Only	0.016	0.012	0.124

The model proposed has the lowest error rates in terms of MSE (0.012), MAE (0.008) and RMSE (0.109) which results in better outcomes in the prediction of stems and the overall error rate of predictions. Table 7 also demonstrates that iterative stripping rules topped with a deep learning model such as RPIS-DSGRU-Bi-LSTM are highly effective in enhancing the accuracy of stemming without generating many errors.

Table 8. Training and validation accuracy

System	Training Accuracy (%)	Validation accuracy (%)
RFIS-DSGRU-Bi-LSTM	95%	93%
Porter Stemmer	86%	84%
Snowball Stemmer	87%	85%
Lancaster Stemmer	83%	81%
GRU Only	91%	89%

The proposed model exhibits the highest training accuracy (95%) and validation accuracy (93%), indicating strong generalization and effective learning during both training and validation phases. Table 8 shows that the proposed model has an advantage when it comes to training and validation performance, and it is a strong solution to the enhancement of the stemming process in documents.

Table 9. Training and validation loss

System	Training Loss	Validation Loss
RFIS-DSGRU-Bi-LSTM	0.09	0.12
Porter Stemmer	0.26	0.31
Snowball Stemmer	0.23	0.29
Lancaster Stemmer	0.28	0.34
GRU Only	0.14	0.16

The proposed model demonstrates the lowest training loss (0.09) and validation loss (0.12), which means that it performs well in reducing errors in the prediction process in the training and validation stages. It is compared in Table 9 that the proposed model corresponds to a better loss minimization, which guarantees the improved accuracy of document processing.

CONCLUSIONS

This paper introduced a RPIS -DSGRU-BiLSTM architecture of document streaming efficiently with controlled morphological variation and semantic similarity, involving rule-based iterative preprocessing and deep semantic modeling, where the RPIS algorithm was effectively used to balance morphological variation, and the Siamese GRU-BiLSTM architecture effectively used to learn both forward and backward contextual dependencies and semantic similarity. Experimental results on benchmark English datasets demonstrated strong performance, achieving 95% training accuracy and 93% validation accuracy, with low error rates of MSE = 0.012, MAE = 0.008, and RMSE = 0.109, consistently outperforming classical stemming approaches such as Porter, Snowball, and Lancaster stemmers across accuracy, precision, recall, F1-score, and root accuracy metrics. In a practical sense, the framework suggested is applicable to large-scale document handling, information retrieval, and streaming text analytics tasks which need an effective noise suppression and semantic continuity. But the present state of the analysis is confined to the English-language datasets and offline processing conditions, and the computational complexity of the Siamese GRU-BiLSTM model can potentially limit its implementation in the resource-constrained or real-time settings. Future directions will involve extension to multilingual and morphologically rich languages, real-time, and online document streaming, and transformer based or hybrid attention mechanisms to enhance scalability, efficiency and semantic representations.

REFERENCES

[1] Jauhar SK, Priyadarshini S, Pratap S, Paul SK. A literature review on applications of Industry 4.0 in Project Management. Operations Management Research. 2023 Dec;16(4):1858-85. https://doi.org/10.1007/s12063-023-00403-x

[2] Lee U, Han A, Lee J, Lee E, Kim J, Kim H, Lim C. Prompt Aloud!: Incorporating image-generative AI into STEAM class with learning analytics using prompt data. Education and Information Technologies. 2024 Jun;29(8):9575-605. https://doi.org/10.1007/s10639-023-12150-4

[3] Mustoip S, Lestari D, Purwati R. Implementation of STEAM Learning Methods to Develop Collaborative and Creative Characters of Elementary School Students. JPS: Journal of Primary School. 2024 Sep 3;1(2):13-20.

[4] Seydali M, Khunjush F, Dogani J. Streaming traffic classification: a hybrid deep learning and big data approach. Cluster Computing. 2024 Jul;27(4):5165-93. https://doi.org/10.1007/s10586-023-04234-0

[5] Fei Z, West GM, Murray P, Dobie G. CNN-based automated approach to crack-feature detection in steam cycle components. International Journal of Pressure Vessels and Piping. 2024 Feb 1;207:105112. https://doi.org/10.1016/j.ijpvp.2023.105112

[6] Arjunan T. Real-time detection of network traffic anomalies in big data environments using deep learning models. International Journal for Research in Applied Science and Engineering Technology. 2024 Mar;12(9):10-22214. https://doi.org/10.22214/ijraset.2024.58946

[7] Duda P, Wojtulewicz M, Rutkowski L. Accelerating deep neural network learning using data stream methodology. Information Sciences. 2024 May 1;669:120575. https://doi.org/10.1016/j.ins.2024.120575

[8] Babooram L, Fowdur TP. Performance analysis of collaborative real-time video quality of service prediction with machine learning algorithms. International Journal of Data Science and Analytics. 2025 Aug;20(2):1513-45. (2025). https://doi.org/10.1007/s41060-024-00548-3

[9] Pookpanich P, Siriborvornratanakul T. Offensive language and hate speech detection using deep learning in football news live streaming chat on YouTube in Thailand. Social Network Analysis and Mining. 2024 Jan 3;14(1):18. https://doi.org/10.1007/s13278-023-01183-9

[10] Roth HR, Xu Z, Hsieh YT, Renduchintala A, Yang IT, Zhang Z, Wen Y, Yang S, Lu K, Kersten K, Ricketts C. Empowering federated learning for massive models with NVIDIA flare. In Federated Learning Systems: Towards Privacy-Preserving Distributed AI 2025 Apr 27 (pp. 1-17). Cham: Springer Nature Switzerland. https://doi.org/10.1007/978-3-031-78841-3_1

[11] Wen Y, Liu X, Yu H. Adaptive tree-like neural network: Overcoming catastrophic forgetting to classify streaming data with concept drifts. Knowledge-Based Systems. 2024 Jun 7;293:111636. https://doi.org/10.1016/j.knosys.2024.111636

[12] Xue P, Chen T, Huang X, Hu Q, Hu J, Zhang H, Yang H, Chen H. Prediction of syngas properties of biomass steam gasification in fluidized bed based on machine learning method. International Journal of Hydrogen Energy. 2024 Jan 2;49:356-70. https://doi.org/10.1016/j.ijhydene.2023.08.259

[13] Baseer KK, Sivakumar K, Veeraiah D, Chhabra G, Lakineni PK, Pasha MJ, Gandikota R, Harikrishnan G. Healthcare diagnostics with an adaptive deep learning model integrated with the Internet of medical Things (IoMT) for predicting heart disease. Biomedical Signal Processing and Control. 2024 Jun 1;92:105988. https://doi.org/10.1016/j.bspc.2024.105988

[14] Yaqub ZT, Oboirien BO, Leion H. Process optimization of chemical looping combustion of solid waste/biomass using machine learning algorithm. Renewable Energy. 2024 May 1;225:120298. https://doi.org/10.1016/j.renene.2024.120298

[15] Khan K. Addressing Fairness and Bias in Machine Learning for Adaptive Video Streaming: Strategies for Enhancing User Experience and Mitigating Algorithmic Discrimination.

[16] Zhou P, Wang L, Liu Z, Hao Y, Hui P, Tarkoma S, Kangasharju J. A survey on generative ai and llm for video generation, understanding, and streaming. arXiv preprint arXiv:2404.16038. 2024 Jan 30. https://doi.org/10.48550/arXiv.2404.16038

[17] Mohandas R, Southern M, O'Connell E, Hayes M. A survey of incremental deep learning for defect detection in manufacturing. Big Data and Cognitive Computing. 2024 Jan 5;8(1):7. https://doi.org/10.3390/bdcc8010007

[18] Horiguchi S, Dohi K, Kawaguchi Y. Streaming active learning for regression problems using regression via classification. InICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) 2024 Apr 14 (pp. 4955-4959). IEEE. https://doi.org/10.1109/ICASSP48485.2024.10448362

[19] Chang CC, Su L. Beast: Online joint beat and downbeat tracking based on streaming transformer. InICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) 2024 Apr 14 (pp. 396-400). IEEE. https://doi.org/10.1109/ICASSP48485.2024.10446611

[20] Raca D, Zahran AH, Sreenan CJ, Sinha RK, Halepovic E, Gopalakrishnan V. Device-based cellular throughput prediction for video streaming: lessons from a real-world evaluation. IEEE Transactions on Machine Learning in Communications and Networking. 2024 Mar 1;2:318-34. https://doi.org/10.1109/TMLCN.2024.3352541

[21] Zheng HS, Liu YY, Hsu CF, Yeh TT. Streamnet: memory-efficient streaming tiny deep learning inference on the microcontroller. Advances in Neural Information Processing Systems. 2023 Dec 15;36:37160-72.

[22] Kumar T, Sharma P, Tanwar J, Alsghier H, Bhushan S, Alhumyani H, Sharma V, Alutaibi AI. Cloud-based video streaming services: Trends, challenges, and opportunities. CAAI Transactions on Intelligence Technology. 2024 Apr;9(2):265-85. https://doi.org/10.1049/cit2.12299

[23] Chen Y, Ma L, Jing L, Yu J. BSDP: Brain-inspired streaming dual-level perturbations for online open world object detection. Pattern Recognition. 2024 Aug 1;152:110472. https://doi.org/10.1016/j.patcog.2024.110472

[24] Selmy HA, Mohamed HK, Medhat W. A predictive analytics framework for sensor data using time series and deep learning techniques. Neural Computing and Applications. 2024 Apr;36(11):6119-32. https://doi.org/10.1007/s00521-023-09398-9

[25] Huang X, Qiao C. Enhancing computational thinking skills through artificial intelligence education at a STEAM high school. Science & Education. 2024 Apr;33(2):383-403. https://doi.org/10.1007/s11191-022-00392-6