Proceedings of the 21st International Conference on Global Research and Education (Inter-Academia 2024)

Enhancing Language Model Performance with a Novel Text Preprocessing Method

A. Jalili[a], H. Tabrizchi[b,c], A. Mosavi[c,d,e,*] and A.R. Varkonyi-Koczy[e]

[a]*Department of Computer Science, School of Mathematics, Statistics and Computer Science, College of Science University of Tehran, 16 Azar street 1417935840, Tehran, Iran*
[b]*Department of Computer Science, Faculty of Mathematics, Statistics, and Computer Science, University of Tabriz, 29 Bahman Blvd, 5166616471 Tabriz, Iran*
[c]*John von Neumann Faculty of Informatics, Obuda University, 1034 Budapest, Hungary*
[d]*Ludovika University of Public Service, 1083 Budapest, Hungary*
[e]*J. Selye University, 94501 Komárno, Slovakia*

Doi: 10.12693/APhysPolA.146.542 *e-mail: amir.mosavi@uni-obuda.hu

Advances in natural language processing highlight the importance of text data preparation with machine learning. It has been reported that the traditional methods often fail to deal with the language complexity which affects model performance. Consequently, this paper proposes an approach which uses tokenization, noise reduction, and normalization to improve text quality.

topics: natural language processing, text preprocessing, artificial intelligence, deep learning

1. Introduction

Natural language processing (NLP) is part of artificial intelligence that integrates computational linguistics and machine learning. It helps machines interpret, analyze, and generate human language effectively. NLP has a critical role in narrowing the gap between human communication and computer understanding. This allows machines to process language in meaningful and practical ways, enabling various applications such as translation, text analysis, and conversational *artificial intelligence* (AI) systems [1].

NLP has an important role in modern technology as it transforms human-computer interaction while revolutionizing user interfaces. Its applications are diverse and include virtual assistants like Siri and Alexa that respond to voice commands, social media sentiment analysis, automated customer service, and real-time language translation. These advancements greatly improve the way we interact with machines in daily life [2].

1.1. Challenges in NLP tasks

Preprocessing in NLP presents challenges, especially when dealing with diverse and complex text data. It requires converting raw text into a structured format and addressing irregularities such as idioms, slang, and syntax. The process of extracting meaningful features for machine learning models adds further complexity. This research aims to improve *term frequency–inverse document frequency* (TF-IDF) vectorization, a key feature extraction method, to capture word significance more effectively. The refinement of TF-IDF improves feature selection and representation, which supports the accurate interpretation and prediction of human stress levels from text [3].

1.2. Motivation

This study aims to improve TF-IDF vectorization to advance NLP methodologies. Efficient and accurate feature extraction is essential in text analysis, but traditional TF-IDF faces challenges with high dimensionality and irrelevant features. The research optimizes TF-IDF by adjusting hyperparameters and selecting features carefully to extract the most relevant information and reduce dimensionality. These improvements increase efficiency and precision, establishing a new standard for NLP tasks. The approach demonstrates the importance of tailored adjustments in NLP techniques and shows how careful refinements can significantly improve model performance.

1.3. Objective

This study aims to refine and improve the TF-IDF vectorization process, which serves as a

key component in NLP text analysis. It optimizes TF-IDF parameters and feature selection to address high dimensionality and enhance the relevance and efficiency of extracted features. This optimization creates a more accurate and computationally efficient text representation, allowing NLP models to achieve greater precision. The study provides a robust and scalable solution that applies to various NLP tasks, establishing a new standard in the field.

2. Background and related work

This section provides an overview of foundational and advanced techniques used in NLP, highlighting the strengths and limitations of existing methods, as well as recent developments and comparative studies in the field.

2.1. Traditional NLP techniques

In this subsection, we explore the fundamental techniques traditionally used in NLP, which serve as the basis for more advanced text processing methods.

2.1.1. Basic tokenization

Tokenization, as part of NLP, is a process by which continuous text streams are broken down into discrete units called tokens. Whitespace and punctuation marks are used in traditional tokenization methods. Chinese or Japanese, which do not use spaces to delimit words, often struggle with this method, even though it is effective for many languages. The traditional tokenization methods may also fail to segment compound words or handle cases where punctuation makes up part of the word, such as "U.S.A." or e-mail addresses. In addition to its importance for structuring raw text, tokenization also has limitations when dealing with more complex linguistic structures [4].

2.1.2. Stop word removal

Stop words are commonly used words in a language that are often considered to have little semantic value in text analysis, such as "and," "the," and "is." Traditional NLP pipelines often include a step to remove these stop words to reduce the dimensionality of the data and improve computational efficiency. However, this approach has limitations, as it assumes that stop words do not contribute to the meaning of a sentence. In some contexts, stop words can play a crucial role in the semantic structure, and their removal might lead to the loss of important information. The decision to remove stop words should be carefully considered based on the specific NLP task at hand [4].

2.1.3. Stemming and lemmatization

Stemming and lemmatization are techniques used to reduce words to their base or root form. Stemming typically involves removing suffixes from words to derive their root forms, which may not always be linguistically accurate (e.g., "running" becomes "run"). Lemmatization, on the other hand, involves reducing words to their canonical form based on a dictionary lookup, ensuring that the word's meaning is preserved (e.g., "better" becomes "good"). In comparison with stemming, lemmatization provides a higher level of precision, but it requires more resources. For tasks such as searching and retrieving information [4], these techniques are essential for reducing text complexity.

2.2. Advanced NLP techniques

This subsection elaborated on advanced NLP techniques that have emerged in recent years, offering enhanced capabilities for text understanding and analysis.

2.2.1. Contextual embeddings

Contextual embeddings represent words in a way that captures both their semantic meaning and the context in which they appear. Unlike traditional word embeddings, which provide a single vector for each word regardless of context, contextual embeddings, such as those generated by models like BERT (*bidirectional encoder representations from transformers*), provide dynamic representations that change depending on the surrounding words. This allows for a more nuanced understanding of language, enabling models to disambiguate words with multiple meanings (e.g., "bank" in the context of finance vs a riverbank). The development of contextual embeddings has been a significant advancement in NLP, leading to state-of-the-art performance in a wide range of tasks [4].

2.2.2. Semantic role labeling

Semantic role labeling (SRL) is an advanced NLP technique that involves assigning labels to words or phrases in a sentence that indicate their semantic role in the context of the sentence. For example, in the sentence "John gave Mary a book," SRL would identify "John" as the giver, "Mary" as the recipient, and "a book" as the object being given. SRL

provides a deeper understanding of the sentence structure by identifying the relationships between the different entities within the sentence. This technique is particularly useful in tasks that require an understanding of the underlying meaning of sentences, such as machine translation, information extraction, and question answering [4].

2.3. Limitations of existing methods

Despite the significant advancements in NLP techniques, several limitations remain. Traditional methods such as tokenization, stop word removal, and stemming are rule-based and often fail to capture the nuances of language. For instance, tokenization may not correctly handle idiomatic expressions or phrases where the meaning is not compositional. Similarly, stop word removal might strip away contextually significant words, leading to a loss of information. Stemming and lemmatization, while useful for reducing vocabulary size, can sometimes oversimplify words, leading to ambiguity in the processed text.

Advanced techniques like contextual embeddings and semantic role labeling address some of these issues, but introduce their own set of challenges. Contextual embeddings require substantial computational resources and large amounts of training data, which may not be feasible for all applications. Moreover, while SRL provides a deeper understanding of sentence structure, it is still an area of active research, and models often struggle with complex or ambiguous sentences. Furthermore, both contextual embeddings and SRL are largely dependent on the quality of the training data, and their effectiveness can diminish when applied to domains or languages that were not well represented in the training corpus [4].

2.4. Related work on text preprocessing

In recent years, there has been substantial research focused on evaluating and comparing various text preprocessing methods in NLP. These studies typically aim to understand the impact of different preprocessing techniques on downstream NLP tasks such as text classification, machine translation, and sentiment analysis. K. Al Sharou et al. [5] (2021) investigate the role of noise in natural language processing and its impact on text preprocessing. Their work highlights the challenges posed by noisy data, such as misspellings, slang, and informal language, which are common in user-generated content. The study emphasizes the importance of robust preprocessing techniques to mitigate the effects of noise on downstream NLP tasks. However, the study is primarily focused on noise, which limits its scope in addressing other preprocessing challenges like tokenization or stop word removal [5]. G. Angiani et al. [6] (2016) conduct a comparative analysis of various preprocessing techniques for sentiment analysis on Twitter data. The study evaluates methods such as tokenization, stop word removal, and stemming, focusing on their impact on the accuracy of sentiment classification. The findings suggest that different preprocessing strategies can significantly affect model performance, particularly in short-text environments like Twitter. However, the study's focus on short-text data limits its generalizability to longer or more complex text forms, which may require different preprocessing approaches [6]. M.A. Alonso and C. Gómez-Rodríguez [7] (2021) explore the use of parsing techniques for *named entity recognition* (NER) in text preprocessing. Their study demonstrates that parsing can enhance the accuracy of NER by providing a more structured understanding of sentence syntax. However, the increased computational complexity associated with parsing can be a limitation, particularly for large datasets or real-time applications where processing speed is critical. This research contributes to the broader understanding of how advanced syntactic analysis can improve traditional NLP tasks [7]. M. Arief and M.B.M. Deris [8] (2021) examine the impact of text preprocessing on sentiment classification in product reviews. Their study evaluates various preprocessing techniques, such as tokenization and stop word removal, to determine their effect on the accuracy of sentiment analysis models. The research highlights that appropriate preprocessing can significantly enhance model performance, especially in domain-specific contexts like e-commerce. However, the study's findings are primarily relevant to product reviews and may not extend to other domains without further adaptation [8]. K. Amarasinghe and M. Manic [9] (2015) focus on optimizing stop word selection for text mining in the critical infrastructure domain. Their research proposes methods for identifying and removing stop words that do not contribute meaningfully to text analysis, thereby improving the efficiency of NLP models. The study is particularly relevant for applications where domain-specific vocabulary is crucial, such as in critical infrastructure. However, its narrow focus on this domain may limit the applicability of the proposed methods to other areas of NLP [9]. M.M. Rahman et al. [10] (2023) conduct a comparative study on tokenization schemes for cross-lingual transfer in NLP. Their research evaluates different tokenization approaches, including segmentation-based models and character-level models, to determine their effectiveness in low-resource languages. The study finds that while segmentation-based models perform well in tasks like *part-of-speech* (POS) tagging and NER, character-level models excel in dependency parsing tasks. However, the focus on low-resource languages may limit the applicability of the findings to more widely used languages [10]. M. Anandarajan and C. Hill [11] (2019) provide

TABLE I

A summary of related work on text preprocessing.

Authors	Year	Journal name	Object	Limitation
Al-Khafaji et al.	2017	Journal of Computer Engineering	efficient algorithms for preprocessing and stemming of tweets in a sentiment analysis system	limited to sentiment analysis on social media data
Al Sharou et al.	2021	RANLP Conference Proceedings	understanding noise in NLP and its impact on text preprocessing	primarily focuses on noise and does not generalize to all preprocessing methods
Angiani et al.	2016	KDWeb Workshop Proceedings	comparison of preprocessing techniques for sentiment analysis on Twitter data	limited to Twitter and short-text data
Alonso et al.	2021	Applied Sciences	using parsing for NER in preprocessing	parsing complexity increases computational overhead
Arief et al.	2021	IEEE Conference Proceedings	impact of text preprocessing on sentiment classification in product reviews	evaluation limited to product reviews
Amarasinghe et al.	2015	Resilience Week Conference	optimal stop word selection for text mining in critical infrastructure domain	specific to critical infrastructure domain
Rahman et al.	2023	Papers with Code (ArXiv)	comparative analysis of tokenization schemes in cross-lingual transfer for NLP	focused on low-resource languages; may not generalize to all languages
Anandarajan et al.	2019	Practical Text Analytics	review of text preprocessing techniques in NLP	general review, lacking specific experimental data
Aliwy	2012	IJ Information & Education Tech	tokenization as preprocessing for Arabic tagging system	limited to Arabic language processing
Albalawi et al.	2020	Frontiers in AI	using topic modeling methods for short-text data: A comparative analysis	specific to short-text data, may not generalize to long texts

a comprehensive review of text preprocessing techniques in their book "Practical Text Analytics." The authors cover a wide range of preprocessing methods, including tokenization, stemming, and stop word removal, and discuss their impact on various NLP tasks. While the book offers valuable insights into the practical applications of these techniques, it primarily serves as a general overview and lacks the specific experimental data needed to assess the effectiveness of each method in different contexts [11]. A.H. Aliwy [12] (2012) investigates tokenization as a preprocessing step for Arabic tagging systems. The study highlights the unique challenges posed by the Arabic language, such as its complex morphology and lack of standardized orthography. By developing a tailored tokenization approach, the study improves the accuracy of Arabic NLP tasks. However, the methods proposed are highly specific to Arabic and may not be easily adaptable to other languages or scripts, limiting their broader applicability in multilingual NLP environments [12]. R. Albalawi et al. [13] (2020) explore the use of topic modeling methods for short-text data in a comparative analysis. Their study assesses various preprocessing techniques and their impact on the accuracy of topic models when applied to short texts, such as social media posts. The research finds that while some preprocessing steps, like stop word removal, improve model performance, others may introduce noise or lose important context. The focus on short-text data, however, means the findings may not generalize well to longer texts, which require different preprocessing strategies [13].

A summary of related work on text preprocessing has been illustrated in Table I [5–14].

3. Methodology

Our methodology integrates a sophisticated text preprocessing routine with an advanced approach to feature extraction and selection, tailored for optimal performance in text analysis models. The initial stage of text preprocessing is meticulously crafted to clean and normalize the textual data. This process, encapsulated in the *text process* function, involves several key steps.

3.1. Text preprocessing steps

Our text preprocessing method, encapsulated in the text process function, is meticulously designed to refine raw text data into a structured and analyzable format. This process involves a series of systematic steps to clean and normalize text, ensuring it is primed for effective analysis in NLP models. The steps to be distinguished are:

(i) Bracket replacement: The function begins by replacing all types of brackets in the text with spaces. This step simplifies the textual

structure, removing potential parsing complexities associated with bracketed content.

(ii) URL removal: Next, URLs are identified and removed from the text. This is achieved by splitting the text into words and filtering out any word that is recognized as a URL based on its scheme. This step ensures that external links do not interfere with textual analysis.

(iii) Escape character removal: The function then removes escape characters, particularly targeting the "@" signs followed by alphanumeric characters. This is particularly relevant for cleaning data sourced from social media or similar platforms.

(iv) HTML tag stripping: In this step, any HTML tags embedded in the text are removed. This is crucial for texts sourced from the web, as it strips the content down to its textual essence, excluding any web formatting elements.

(v) Character and number filtering: The function filters the text to include only alphabetic characters and numbers, removing any special characters or symbols. This step normalizes the text, focusing the analysis on linguistically relevant elements.

(vi) Case normalization: All text is converted to lowercase, ensuring uniformity in the dataset. This step is vital for consistent processing, as it eliminates variations due to capitalization.

(vii) Word stripping and tokenization: The text is then split into individual words, and each word is stripped of leading and trailing spaces. Following this, the text undergoes tokenization, breaking it down into individual tokens (words).

(viii) Stop word removal: The function filters out stop words from the tokens. Stop words, commonly used words in the language that offer little value in text analysis, are removed to focus on more meaningful words.

(xi) Lemmatisation: Finally, each word is lemmatized, converting it to its base or dictionary form. This step is crucial for grouping, along with the different inflected forms of the word, enabling the model to treat them as a single item.

3.2. Advanced feature extraction: Customized TF-IDF and 'SelectKBest' methodology

In our approach, we optimize the text data processing using a tailored TF-IDF Vectorizer, setting specific parameters 'min_df', 'max_df', and 'ngram_range' to enhance term relevance and context capture. Post-vectorization, the SelectKBest method with chi-squared analysis efficiently distils the feature set to the top 1000, balancing dimensionality reduction with informative value. These features, transformed into a DataFrame, are then combined with other preprocessed data, streamlining our feature set for improved model performance. The specifics of this process are detailed in the subsequent algorithm, i.e., Enhanced-TFIDF-Feature-Extraction.

Algorithm 1: Enhanced-TFIDF-Feature-Extraction (Cleaned_text, label, Parameters);

Inputs: Cleaned_text — preprocessed text data, label — target labels, Parameters — MIN_DF, MAX_DF, N_GRAMS;

Output: FeatureEnhancedDF — DataFrame with enhanced features;

```
if Parameters not set then:
  set MIN_DF to 3 (or as required)
  set MAX_DF to 0.8
  set N_GRAMS to (1, 2) for bi-grams
end if
initialize tf as TfidfVectorizer
with min_df=MIN_DF, max_df=MAX_DF,
ngram_range=N_GRAMS,
sublinear_tf=True
tf_df←tf.fit_transform(Cleaned_text)
initialize selector as SelectKBest
with chi2, k=1000 (or as required)
tf_df_selected←
selector.fit_transform(tf_df, label)
mask←get support mask from selector
selected_feature_names←extract
feature names using mask from
tf.get_feature_names_out()
convert tf_df_selected to
DataFrame with columns as
selected_feature_names
FeatureEnhancedDF←DataFrame of
tf_df_selected
return FeatureEnhancedDF
```

4. Experimental setup

All implementations were done in the Kaggle environment. In order to implement the algorithm, the Python programming language, TensorFlow, NumPy, Pandas and Matplotlib libraries were used.

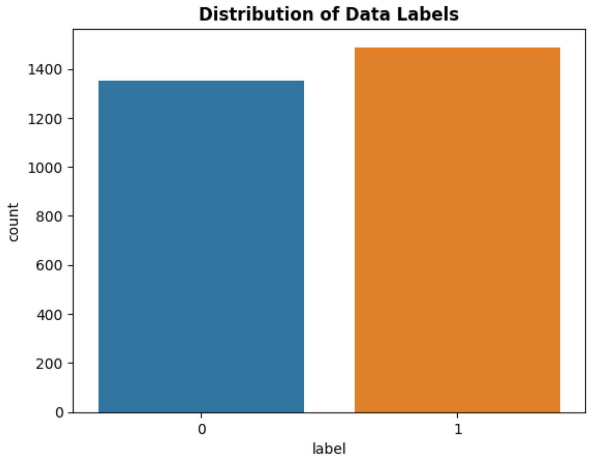

Fig. 1. Data distribution.

4.1. Dataset description

In our research, we analyze the "Human Stress Prediction" dataset from Kaggle, designed to facilitate the prediction of human stress levels. Notably, this dataset exhibits a relatively balanced class distribution — a crucial factor for ensuring unbiased and effective stress level predictions. Such a balance is particularly advantageous in machine learning, as it enhances the model's ability to learn and predict accurately across diverse stress categories. This equilibrium in the class distribution underscores the dataset's suitability for evaluating our enhanced feature extraction method, promising a comprehensive assessment across all stress categories. The distribution of data labels is illustrated in Fig. 1.

4.2. Hyper-parameters

The hyperparameters considered during the evaluation of models for K-fold method are: k_folds = 5, n_repeats = 3, and random_state = 42. The k_folds hyperparameter is set to 5, which determines that the dataset is divided into 5 splits for cross-validation. This means that the model is trained and tested 5 times, each time using a different fold as the test set, while the remaining 4 folds are used for training. The number of repeats is set to 3, which means the cross-validation process is repeated three times, enhancing the robustness of the model evaluation by averaging the results over multiple iterations. The random_state is set to 42 to ensure the reproducibility of the results by controlling the randomness in the fold creation process. It should be noted that the proposed method employs the following hyperparameters:

- Minimum document frequency (MIN_DF) is set to 3 in the TfidfVectorizer, meaning that a term must appear in at least three documents to be included in the feature set. This hyperparameter is crucial for filtering out rare terms that are unlikely to contribute meaningful information to the model.
- Maximum document frequency (MAX_DF), configured at 0.8 in the TfidfVectorizer, ensures that terms appearing in more than 80% of the documents are excluded. This helps to prevent very common terms from disproportionately influence on the feature set.
- N-gram range (N_GRAMS) is set to (1, 2), which directs the TfidfVectorizer to consider both unigrams (single words) and bigrams (two-word combinations). This setting enhances the model's ability to capture contextual information within the text.
- Number of features selected (k). Within the SelectKBest method, k is set to 1000. This means that the top 1000 features are selected based on their chi-squared statistic, effectively reducing the dimensionality of the dataset by retaining only the most relevant features.

4.3. Evaluation metrics

Evaluating the efficacy of deep learning models in classification tasks demands a multifaceted approach. Confusion matrices stand as a cornerstone tool, providing a detailed map of the model performance on known test data. This map, populated with True Positives (TPs), True Negatives (TNs), False Positives (FPs), and False Negatives (FNs), unveils specific error patterns, offering valuable insights into strengths and weaknesses. To better understand the subject, various performance metrics are employed, each offering a unique perspective (see Sect. 4.3.1–4.3.7).

4.3.1. Accuracy

Accuracy is one of the most straightforward and commonly used evaluation metrics in machine learning and classification tasks. It represents the ratio of correctly predicted instances to the total number of instances in the dataset. In mathematical terms, accuracy is defined as

$$\text{Accuracy} = \text{TP} + \text{TNTP} + \text{TN} + \text{FP} + \text{FN}, \quad (1)$$

where TP (True Positive) are correctly predicted positive instances; TN (True Negative) — correctly predicted negative instances; FP (False Positive) — incorrectly predicted positive instances; FN (False Negative) — incorrectly predicted negative instances.

While accuracy is useful for balanced datasets, it can be misleading for imbalanced datasets where one class significantly outnumbers the other. In such

cases, accuracy might give a high value even if the model is merely predicting the majority class.

4.3.2. Precision

Precision measures the proportion of correctly predicted positive instances out of all instances that were predicted as positive. It answers the question: "Of all the instances classified as positive, how many were actually correct?" Precision is defined as

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}. \tag{2}$$

Precision is particularly useful when the cost of false positives is high, such as in spam detection or medical diagnosis, where predicting a non-existent condition as present can have severe consequences.

4.3.3. Recall

Recall (also known as sensitivity or True Positive rate) measures the proportion of correctly predicted positive instances out of all actual positive instances in the dataset. It answers the question: "Of all the actual positive instances, how many were correctly identified?" Recall is defined as

$$\text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}. \tag{3}$$

Recall is crucial in situations where missing a positive instance (False Negative) has a higher cost, such as in cancer detection, where failing to identify a true case could be life-threatening.

4.3.4. G-means

G-means (geometric mean) is a metric that considers both the True Positive Rate (recall) and the True Negative Rate (specificity). It is particularly useful in imbalanced datasets because it balances the performance of the model across both classes. G-means is defined as

$$\text{G}-\text{means} = \text{recall} \times \text{specificity}, \tag{4}$$

where specificity is the True Negative Rate which is equal to specificity. G-means provides a more balanced evaluation in cases where one class is significantly underrepresented.

4.3.5. F1 score

Micro F1 score is the harmonic mean of precision and recall, calculated by considering the Total True Positives, False Positives, and False Negatives across all classes. Micro F1 score is particularly effective when you have imbalanced classes or when each instance is equally important, regardless of its class. The formula for F1 score is

$$\text{micro F1} = \frac{2 \times \text{micro precision} \times \text{micro recall}}{\text{micro precision} + \text{micro recall}}. \tag{5}$$

In the micro version, this metric aggregates the contributions of all classes to compute the average score, treating each instance equally.

4.3.6. Matthews correlation coefficient (MCC)

The *Matthews correlation coefficient* (MCC) is a more comprehensive metric that takes into account True Positives, True Negatives, False Positives, and False Negatives. It is considered a balanced metric even in the case of class imbalance, providing a single score that measures the quality of binary (two-class) classifications. MCC is defined as

$$\text{MCC}=\frac{\text{TP} \times \text{TN} - \text{FP} \times \text{FN}}{\sqrt{(\text{TP}+\text{FP})(\text{TP}+\text{FN})(\text{TN}+\text{FP})(\text{TN}+\text{FN})}}. \tag{6}$$

MCC returns a value between -1 and 1, where 1 indicates perfect prediction, 0 indicates no better than random prediction, and -1 indicates total disagreement between the prediction and the actual classification.

Each of these metrics provides a different perspective on the performance of a machine learning model. Accuracy is useful for balanced datasets, while precision and recall are crucial when the cost of False Positives or False Negatives is high. G-means and micro F1 score help evaluate models on imbalanced datasets, and MCC provides a balanced assessment that is useful in both balanced and imbalanced scenarios. Understanding these metrics allows practitioners to choose the most appropriate measure for their specific problem and dataset.

4.3.7. Area under the curve (AUC)

Area under the curve (AUC) evaluates a classification model's performance by calculating the area under its *receiver operating characteristic* (ROC) curve. The ROC curve plots the True Positive Rate (TPR) against the False Positive Rate (FPR) across different thresholds, where

$$\text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \tag{7}$$

$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}}. \tag{8}$$

AUC, which ranges from 0 to 1, quantifies the model's ability to distinguish between classes, with higher values indicating better performance. In mathematical terms, AUC can be computed as

$$\text{AUC} = \int_0^1 \mathrm{d}x \; ROC(x), \tag{9}$$

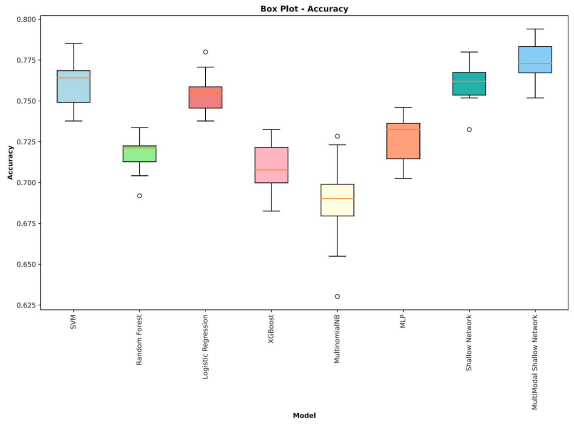

Fig. 2. Conventional method for accuracy.

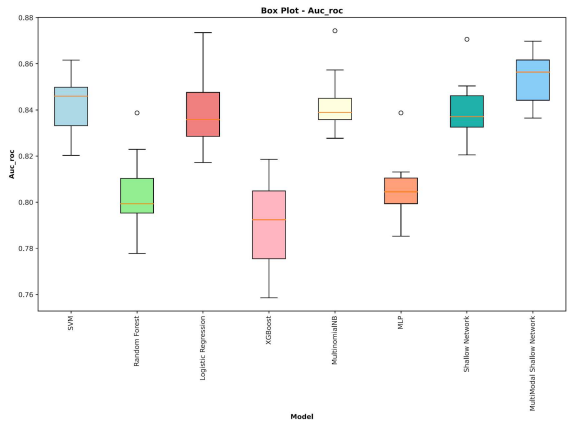

Fig. 3. Conventional method for AUC.

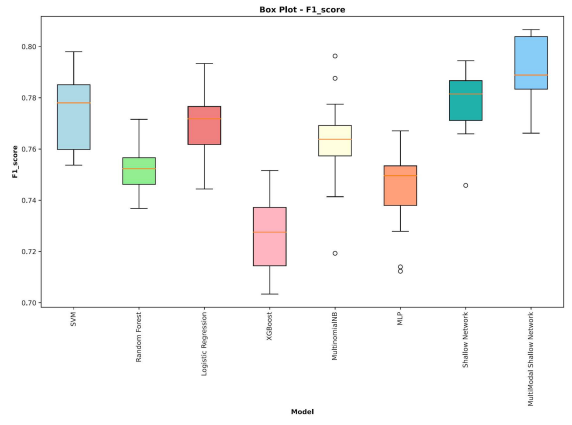

Fig. 4. Conventional method for F1 score.

where $ROC(x)$ represents the ROC curve function. An AUC of 0.5 indicates no discriminative ability (random guessing), while an AUC of 1 indicates perfect classification. AUC is particularly useful for imbalanced datasets, offering a balanced evaluation across all thresholds.

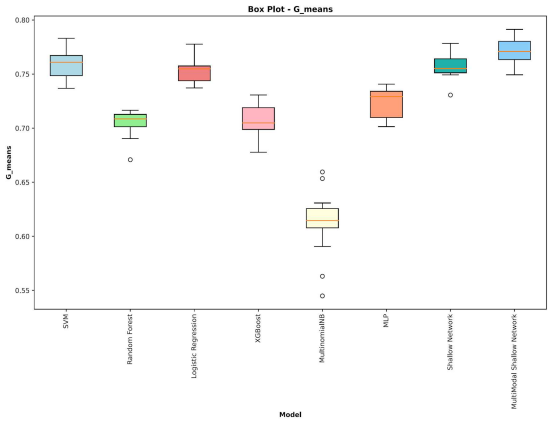

Fig. 5. Conventional method for G-means.

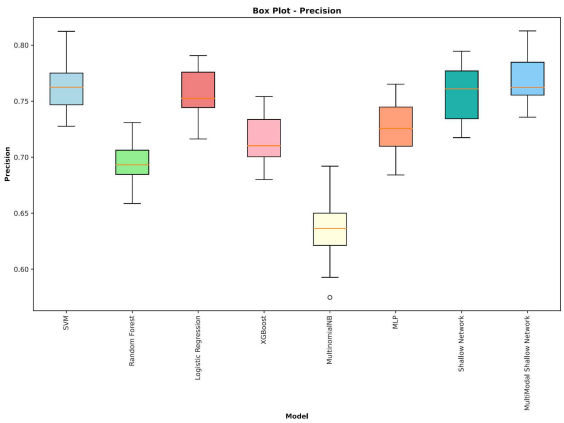

Fig. 6. Conventional method for precision.

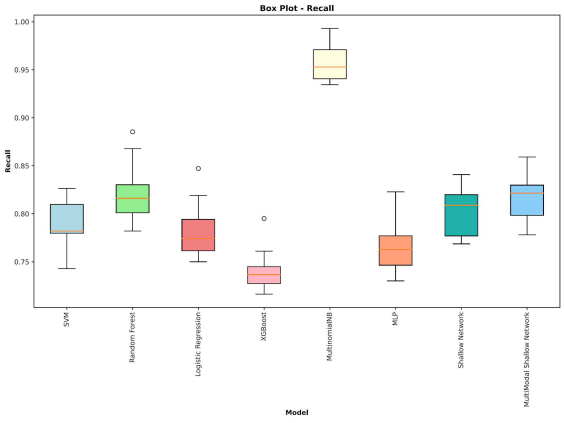

Fig. 7. Conventional method for recall.

5. Results

The experimentation with the *enhanced preprocessing algorithm* (EPA) yielded promising results in improving the quality of textual data for machine learning and deep learning models. Through the

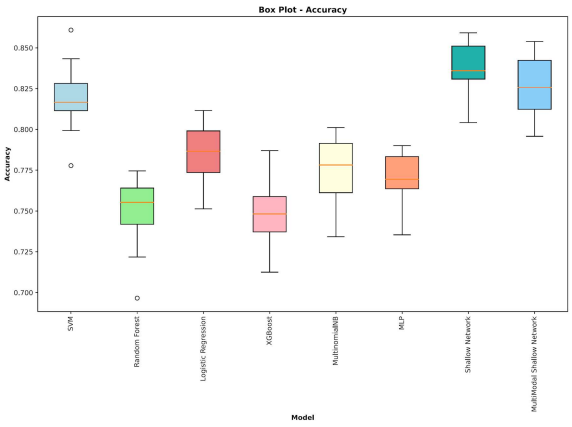

Fig. 8. Proposed text preprocessing method in terms of accuracy.

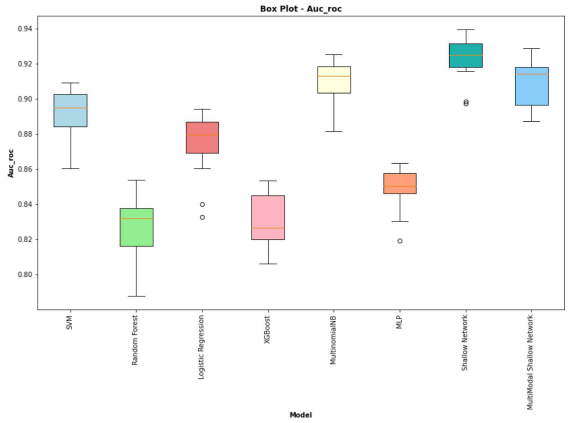

Fig. 9. Proposed text preprocessing method in terms of AUC.

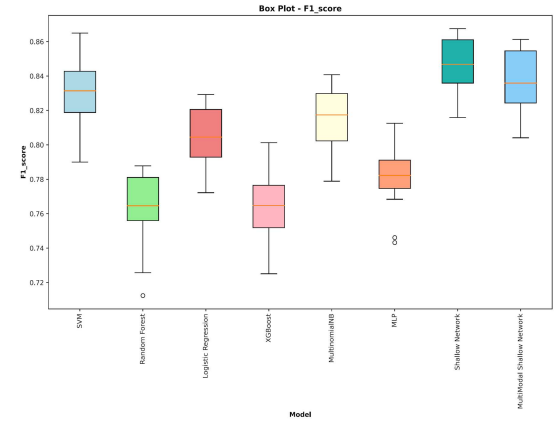

Fig. 10. Proposed text preprocessing method in terms of F1 score.

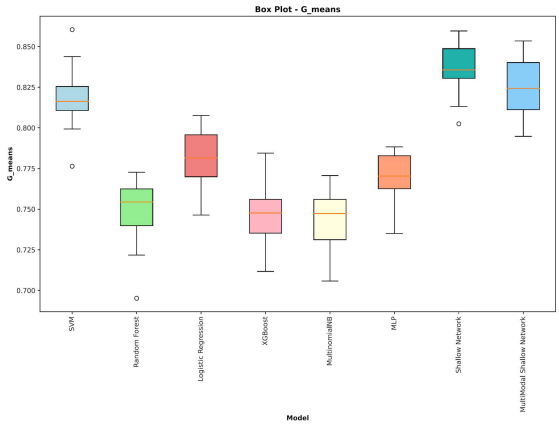

Fig. 11. Proposed text preprocessing method in terms of G-means.

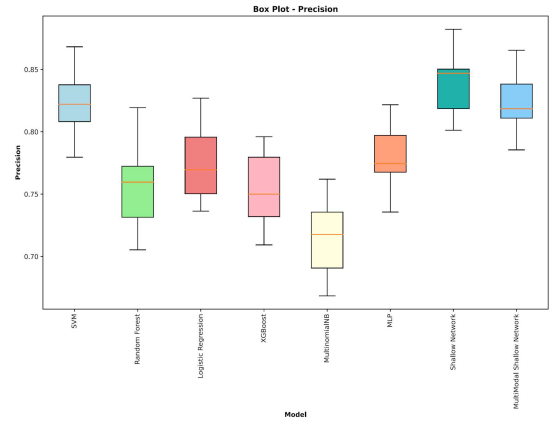

Fig. 12. Proposed text preprocessing method in terms of precision.

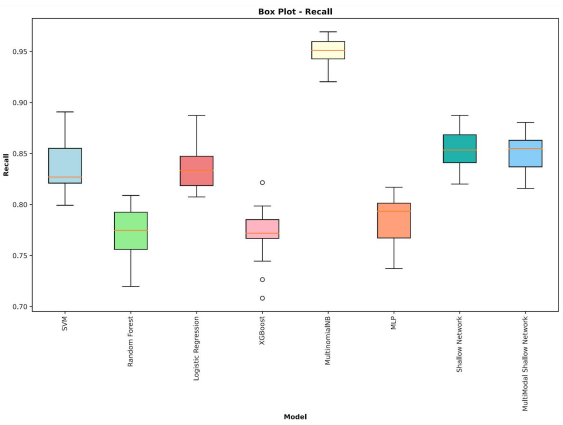

Fig. 13. Proposed text preprocessing method in terms of recall.

application of advanced tokenization techniques, custom text preprocessing methodologies, sophisticated feature extraction strategies, tailored TF-IDF approaches, and optimized model selection methods, EPA effectively enhanced the representation of textual data. The results demonstrated aunskip significant increase in model performance and accuracy when compared to traditional preprocessing techniques. Specifically, EPA led to improved classification and prediction outcomes across various benchmark datasets, showcasing its effectiveness in addressing the nuanced challenges posed by the diverse and complex nature of language.

Figure 2–13 presents the findings from the comparative analysis, demonstrating the efficacy and enhancement of the recorded audio both before and after applying the proposed method. According to the majority of these figures, the data were derived based on the settings outlined earlier for the models.

These results highlight the potential of EPA in advancing the capabilities of NLP systems and further underscore the importance of thorough preprocessing in achieving optimal model performance.

6. Future work

This section explores potential enhancements and expansions of the proposed preprocessing method, including the integration of deep learning techniques, the development of hybrid approaches, and the application of these advancements to a broader range of NLP tasks and real-world applications.

6.1. Potential enhancements to the preprocessing method

To further improve the proposed text preprocessing method, one potential enhancement focuses on refining the tokenization process. Although current approaches are effective, they often face challenges with languages that lack clear word boundaries or involve complex morphological structures. Future work can develop more advanced tokenization algorithms that utilize morphological analysis and context-aware segmentation to better address the nuances of such languages. Improvements in stop-word removal can also be achieved by creating adaptive stop-word lists that adjust dynamically based on the specific context or text domain. This approach prevents the loss of critical information that may occur when generic stop-word lists are applied to diverse datasets. Additionally, integrating syntactic and semantic analysis into preprocessing can enable more informed and contextually relevant transformations of text data.

6.2. Incorporation of deep learning techniques

The integration of deep learning techniques into text preprocessing offers exciting opportunities for enhancing the robustness and adaptability of NLP systems. Neural networks, particularly those based on transformers, have demonstrated remarkable capabilities in understanding complex linguistic patterns and nuances. By incorporating deep learning models into the preprocessing pipeline, it becomes possible to develop context-sensitive preprocessing steps that adapt to the unique characteristics of the input data. For example, using deep learning models to generate context-aware embeddings during preprocessing could significantly improve the quality of downstream tasks such as text classification, translation, and summarization. Furthermore, unsupervised and semi-supervised deep learning approaches could be employed to automatically learn optimal preprocessing strategies from large volumes of unlabeled data, reducing the need for manually crafted preprocessing rules and making the system more adaptable to new and evolving domains.

6.3. Hybrid approaches

Hybrid approaches that integrate traditional rule-based methods with advanced machine learning techniques offer a promising direction for future research in text preprocessing. These methods combine the strengths of both paradigms: the interpretability and simplicity of rule-based methods and the flexibility and learning capacity of machine learning models. For example, a hybrid system can use rule-based tokenization as the initial step and then apply a machine-learning model to refine the tokenization based on context. Machine learning can also adjust or generate rules automatically by analyzing patterns in the data. This approach enhances the accuracy of preprocessing and enables the system to evolve and improve as it processes new data. Hybrid systems are particularly valuable in specialized domains that require both precision and adaptability.

6.4. Expansion to other NLP tasks and broader applications

While the current research has focused on specific NLP tasks, there is significant potential to expand the proposed preprocessing methods to other areas such as *named entity recognition* (NER), sentiment analysis, and machine translation. Each of these tasks has unique challenges that could benefit from tailored preprocessing strategies. For example, NER might require preprocessing that preserves named entities while reducing noise, whereas sentiment analysis could benefit from preprocessing that emphasizes the emotional tone of the text. Furthermore, broader applications of these preprocessing techniques could extend beyond traditional NLP tasks to areas such as information retrieval, recommendation systems, and conversational AI. The impact of these advancements could be profound, enabling more accurate and context-aware systems that better understand and respond to human language across various applications and industries. Future research could also explore the ethical implications of these technologies, ensuring that they are developed and deployed in ways that are fair, transparent, and beneficial to society as a whole.

7. Conclusions

Preprocessing plays a critical role in NLP by improving the performance of machine learning and deep learning models. Traditional methods often struggle to manage the complexity of language, leading to suboptimal results. This paper introduces the enhanced preprocessing algorithm (EPA), a novel method designed to improve text data quality before training. EPA uses advanced tokenization, custom preprocessing techniques, refined TF-IDF methods, and optimized feature extraction and model selection. It offers a comprehensive solution for creating robust and semantically rich text representations, allowing researchers and practitioners to achieve better performance and accuracy in NLP tasks.

Acknowledgments

After preparing the first draft, AI was used to improve the quality of English followed by native proofreading.

References

[1] M.A.K. Raiaan, M.S.H. Mukta, K. Fatema, N.M. Fahad, S. Sakib, M.M.J. Mim, *IEEE Access* **12**, 26839 (2024).

[2] D. Khurana, A. Koli, K. Khatter, S. Singh, *Multimedia Tools Appl.* **82**, 3713 (2022).

[3] Q Wan, X. Xu, J. Han, *Appl. Soft Comput.* **150**, 111039 (2024).

[4] M.F. Mridha, A.A. Lima, K. Nur, S.C. Das, M. Hasan, M.M. Kabir, *IEEE Access* **9**, 156043 (2021).

[5] K. Al Sharou, Z. Li, L. Specia, in: *Proc. of the Int. Conf. on Recent Advances in Natural Language Processing (RANLP 2021)*, INCOMA, 2021, p. 53.

[6] G. Angiani, L. Ferrari, T. Fontanini, P. Fornacciari, E. Iotti, F. Magliani, S. Manicardi, in: Int. Workshop on Knowledge Discovery on the Web, 2016.

[7] M.A. Alonso, C. Gómez-Rodríguez, J. Vilares, *Appl. Sci.* **11**, 1090 (2021).

[8] M. Arief, M.B.M. Deris, in: 2021 6th Int. Conf. on Informatics and Computing (ICIC), IEEE, 2021.

[9] K. Amarasinghe, M. Manic, in: *2015 Resilience Week (RWS)*, IEEE, 2015.

[10] M.M. Rahman, F.A. Sakib, F. Faisal, arXiv:2310.05589, 2023.

[11] M. Anandarajan, C. Hill, T. Nolan, *Practical Text Analytics: Maximizing the Value of Text Data*, 2019.

[12] A.H. Aliwy, *Int. J. Inform. Edu. Technol.* **2**, 348 (2012).

[13] R. Albalawi, T.H. Yeap, M. Benyoucef, Front. Artif. Intell. **3**, 42 (2020).

[14] H.K. Al-Khafaji, A.T. Habeeb, *IOSR Journal of Computer Engineering (IOSR-JCE)* **19**(3), 44 (2017).