DOI: 10.6977/IJoSI.202512_9(6).0003
S. Upadhyay, H. K. Soni/Int. J. Systematic Innovation, 9(6), 27-43 (2025)

Automatic text summarization framework for multi-text and multilingual documents using an ensemble of HIN-MELM-AE and improved DePori model

Sunil Upadhyay*, Hemant Kumar Soni

Department of Computer Science and Engineering, Amity School of Engineering and Technology, Amity University Madhya Pradesh, Gwalior, Madhya Pradesh, India

*Corresponding author E-mail: supadhyay@gwa.amity.edu

(Received 28 November 2024; Final version received 10 November 2025; Accepted 02 December 2025)

Abstract

Automatic text summarization (ATS) has gained increasing significance in recent years due to the rapid growth of textual data across digital platforms. The main objective of ATS is to generate a concise, informative summary from a lengthy document. Multi-document and multilingual summarization has been largely underexplored in previous research. This study presents an improved ensemble learning-based ATS system with slang filtering, using the Hyperfan-IN multilayer extreme learning machine-based autoencoder (HIN-MELM-AE) and the improved Dehghani poor-and-rich optimization algorithm (DePori). The original text undergoes comprehensive preprocessing, after which slang is detected and removed using DePori. Subsequently, the clean text is processed through info-squared C-means clustering, latent Dirichlet allocation-based topic modeling, term frequency–inverse document frequency weighting, and frequent-term extraction. Next, part-of-speech (POS) tagging is performed using a sememe similarity-induced hidden Markov model, and key entities are extracted from the transformed and POS–tagged data. Distilled bidirectional encoder representations from transformers (DBERT) are used to convert these entities into vectors. The final summary is generated through a combination of HIN-MELM-AE, stack autoencoder, variational autoencoder, and DBERT models, followed by cosine similarity calculation, voting-based fusion, re-ranking, and selection of the optimal sentences. Experimental results indicate that the proposed framework achieves superior performance 97.92% of the time, outperforming existing ATS methods.

Keywords: Hyperfan-IN Multilayer Extreme Learning Machine Auto Encoder, Info-Squared Fuzzy C-Means Clustering, Latent Dirichlet Allocation, Parts of Speech, Sentence Bidirectional Encoder Representations from Transformers, Sememe Similarity-Induced Hidden Markov Model, Term Frequency–Inverse Document Frequency, Variational Auto Encoder

1. Introduction

The development of the internet and big data has led to an enormous amount of textual content, which grows exponentially every day. Users spend a considerable amount of time searching for information and may read irrelevant portions of the content (El-Kassas et al., 2021). Therefore, summarizing text resources has become essential and increasingly important. However, manual text summarization is time-consuming and requires significant effort and cost (Yadav et al., 2022). To address this, automatic text summarization (ATS) can save users time and effort by generating summaries automatically. Typically, ATS creates summaries composed of significant sentences and relevant information from the original document (Widyassari et al., 2022). The summaries generated by ATS consider aspects such as readability, information diversity, and sentence ordering (Wahab et al., 2024).

The applications of text summarization include legal document abstraction and clinical text summarization (Zhang et al., 2020).

In general, ATS consists of two main methods: extractive and abstractive. The extractive method identifies the most important parts of the text based on scoring criteria and then combines them to produce a summary. In contrast, the abstractive method is more complex; it involves paraphrasing the text to generate a concise version using words and sentences that differ from the original text document (Syed et al., 2021).

In recent years, many approaches have been developed to perform ATS for single- or multi-document summarization. For a single document, ATS produces a concise summary of the document (Payak et al., 2020). In multi-document summarization, users can quickly gain comprehensive knowledge of a topic by reviewing multiple documents (Awasthi et al., 2021). In existing studies, K-means clustering and Gensim Word2Vec have been used for ATS; however, these approaches require high memory (Haider et al., 2020). Moreover, certain models employ deep reinforcement learning to enhance ATS performance; however, they encounter issues with computational complexity (Alomari et al., 2022).

Similarly, the term frequency–inverse document frequency (TF-IDF) algorithm has been used in several existing studies for ATS; however, it is not suitable for abstractive summarization (Manjari et al., 2020). Several researchers also employed a combined latent semantic analysis (LSA) and bidirectional encoder representations from transformers (BERT) model for ATS—they reported that the model is highly sensitive to term variability (Gupta & Patel, 2021). In addition, sequence-to-sequence Seq2Seq with a recurrent neural network (RNN) has been used to perform ATS; however, these models are affected by vanishing or exploding gradient problems (Prasad et al., 2020). In addition, several studies employed latent Dirichlet allocation (LDA) topic modeling and soft-cosine similarity to summarize text documents effectively; however, these approaches consider limited linguistic features (Jain & Rastogi, 2020; Onah et al., 2022). Moreover, relatively few studies have addressed ATS for multi-document and multilingual summaries. To address these gaps, this study presents an improved ensemble learning-based ATS with slang filtering, using the Hyperfan-IN multilayer extreme learning machine-based autoencoder (HIN-MELM-AE) and the Dehghani poor-and-rich optimization algorithm (DePori) techniques.

Conventional methods have rarely focused on ATS for multi-document and multilingual summaries. Therefore, the proposed model explicitly targets ATS in this multi-document and multilingual setting. In existing studies, handling unknown or out-of-vocabulary (OOV) words was challenging (El-Kassas et al., 2020). In addition, using neural network-based word embeddings for the semantic representation of sentences remained difficult in ATS. Previous studies used less effective part-of-speech (POS) tagging techniques, especially for classifying word classes such as nouns, verbs, and adjectives (Hailu et al., 2020). Owing to noisy and incomplete data, conventional methods exhibited performance stability issues (Jiang et al., 2021). Furthermore, due to the lower score values of existing score-based text summarization techniques, certain important sentences might be excluded from the summary. Most existing ATS methods attained low accuracy and efficiency when summarizing long texts.

The proposed model performs ATS for multi-text documents and multilingual summaries. By incorporating key steps such as preprocessing, slang identification, filtering, and data transformation, it enables effective summarization in these settings. For semantically effective sentence representation, distilled BERT (DBERT)-based entity vectorization is performed. The sememe similarity-induced hidden Markov model (SemSim-HMM) is introduced to perform POS tagging and more accurately classify word categories. Preprocessing steps—including natural language processing (NLP), metadata removal, language identification and translation, as well as uniform resource locator (URL), symbols, and emotions removal—are applied to improve model's stability.

To achieve effective ATS, ensemble methods—including HIN-MELM-AE, stack autoencoder (SAE), variational autoencoder (VAE), and DBERT—are utilized. Data transformation steps, such as info-squared fuzzy C-means (InS-FCM)-based clustering, LDA-based topic modeling, TF-IDF analysis, and frequent term selection, are performed to accurately summarize the longer texts.

In this paper, Section 2 presents the existing works, Section 3 describes the proposed methodology, Section 4 details the materials and methods used in this study, Section 5 discusses the results and discussion, and finally, Section 6 concludes the proposed work with future scope.

2. Literature Review

Hailu et al. (2020) introduced an ATS and evaluation framework based on word embedding. They employed a word embedding-based text summarization technique, and the cosine similarity between each sentence in the document and the keywords was evaluated. The model effectively identified the top-*n* sentences of a source text. However, the study used a less effective POS tagging technique, particularly for classifying word categories such as nouns, verbs, and adjectives.

Jiang et al. (2021) utilized an attention-based bidirectional long short-term memory (LSTM) for hybrid ATS. Four ATS methods—enhanced semantic network, decoder attention based on a pointer network (DA-PN), DA-PN with a coverage mechanism (DA-PN + cover), and mixed learning objective (MLO) function combined with DA-PN + cover (DA-PN + cover + MLO)—were employed. The model effectively addressed the problem of OOV words; however, its performance stability was affected by noisy and incomplete data.

Hernandez-Castaneda et al. (2023) proposed a model for the automatic generation of an objective function for text summarization. In this approach, heuristic functions were automatically generated using genetic programming for ATS. In addition, the model automatically derived an orientation function, resulting in higher-quality summaries. However, the challenge of increased computational complexity with larger text datasets remained.

Zhong & Wang (2022) analyzed ATS for domain adaptation. In this work, a multi-task learning method was employed for ATS, in which bidirectional and auto-regressive transformers were integrated as shared text encoding layers. The model effectively performed ATS across multiple domains; however, it was time-consuming, which introduced significant time complexity issues.

Abo-Bakr and Mohamed (2023) proposed an automatic multi-document text summarization model. Initially, the whole text was preprocessed through sentence segmentation, word tokenization, stop-word removal, and stemming. Then, the large-scale sparse multi-objective optimization algorithm was employed to perform ATS. The model effectively extracted a small set of sentences from a large multi-document text. However, the generated summaries lacked cohesion and semantics, thereby reducing the efficiency of the model.

Hosseinabadi et al. (2022) explored an ATS model based on iterative sentence scoring and extraction schemes. They performed processes such as preprocessing, graph representation, sentence clustering, cluster scoring, sentence selection and elimination, and final rearrangement for ATS. They achieved better results in terms of precision and recall. However, they did not take into account the broader context and the relationship between sentences, which may result in disjointed summaries.

Belwal et al. (2021) employed a topic-based vector space model and a semantic measure for text summarization. They used a combination of topic vector and individual topic vector methods to generate the topic vector from the given document. Eventually, the summarized text obtained from this model was closer to human-generated summaries. The model, however, may lose fine-grained context and subtleties of individual words.

Muniraj et al. (2023) introduced a model for hybrid text summarization of transliterated news articles. They performed ATS using a hybrid Seq2Seq model—the encoder contained three bidirectional LSTM units, and the decoder contained one LSTM unit. The model provided high performance and produced higher Recall-Oriented Understudy Gisting Evaluation (ROUGE) values. However, the hybrid Seq2Seq model required large datasets for training and was prone to overfitting issues, thereby reducing accuracy.

Alami Merrouni et al. (2023) explored a text summarization method for generating both extractive and abstractive summaries. They applied statistical and semantic scoring methods, along with a graph-based approach, were employed for text summarization. The model effectively removed non-essential information and produced coherent, grammatically correct abstractive and extractive summaries. Nevertheless, the model lacked key NLP operations, which affected the quality of the final results.

Kouris et al. (2024) introduced a text summarization model based on semantic graphs. The semantic graph representation, along with reinforcement learning and transformer-based models, was employed for ATS. The proposed model demonstrated promising performance and high robustness. However, the model struggled to handle unknown or OOV words, thereby reducing its effectiveness.

Onan and Alhumyani (2024a) explored an extractive TypeScript framework utilizing fuzzy topic modeling (FuzzyTM) and BERT. They applied fuzzy logic to enhance topic modeling, resulting in more accurate modeling of word-topic relationships. This method achieved ROUGE-1 and ROUGE-2 scores of 45.3774 and 24.1808, respectively. Although this approach improved the quality of text summarization, the performance of the framework was compromised due to its interpretability.

Onan and Alhumyani (2024b) present a novel approach to extractive text summarization (ETS) by integrating large language models with hierarchical positional encoding, which supports deep semantic understanding and improved context awareness in summary generation. However, while the framework shows enhanced performance in capturing and preserving contextual relationships within texts, it incurs high computational costs, which may pose challenges for large-scale or real-time applications.

Hassan et al. (2024) proposed an ETS approach using NLP with an optimal deep learning (ETS-NLPODL) model. They reported that the

ETS-NLPODL model achieved strong performance compared to other models across multiple evaluation measures.

Liu et al. (2024) improved TextRank by using deep contextual embeddings and K-means clustering, yielding more coherent and less redundant summaries with higher ROUGE scores than baseline methods. Although this method is computationally slower and sensitive to cluster initialization, it offers an effective upgrade to classic TextRank for quality-oriented extractive summarization.

Yang et al. (2025) introduced a novel generative adversarial network-based framework that combines transductive LSTM with Distil-BERT embeddings and reinforcement learning to fine-tune sentence selection. The model alleviates greedy biases typical of conventional extractive methods using the discriminator as a reward in a policy gradient setting, and it obtained high ROUGE-1 (52.45), ROUGE-2 (26.46), and ROUGE-L (44.85) scores on standard datasets such as CNN/Daily Mail. The method produces more coherent and concise summaries and offers value for domains where time efficiency and operational costs are critical, such as engineering and healthcare. However, the model's reliance on generative adversarial networks and reinforcement learning makes it computationally expensive, thereby limiting its use to environments with ample computing resources.

Divya and Sripriya (2025) developed a semantics-driven extractive approach utilizing transformer-based multilingual clustering and hybrid clustering algorithms for Tamil and English texts. The system converts text into embeddings using multilingual BERT variants, after which it applies density-based and centroid clustering to identify the most relevant sentences. Evaluations on the constructed bilingual corpora showed that the proposed method outperformed baselines such as LexRank and TextRank in terms of precision and informativeness, resulting in concise summaries that preserve cross-lingual nuances. They reported that their research is instrumental for low-resource language development and that it improved F1-scores for Tamil texts by up to 15%. However, the model may require language-specific adjustments to achieve optimal performance, and it faces challenges in identifying outliers during clustering, which may lead to fragmented summaries in noisy datasets.

Mandale-Jadhav (2025) investigated a multifaceted extractive summarization pipeline that integrated graph-based ranking, deep learning classifiers, and hybrid linguistic-neural features for scalable processing. The system combines TF-IDF for the initial weighting step with graph spreading (using eigenvector centrality) and a small neural component that scores sentence embeddings, enabling efficient processing of benchmarks such as Multi-News. It extends the capabilities of existing methods while achieving a balanced ROUGE performance (for example, a ROUGE-1 value of approximately 48%), and the method is computationally efficient enough for deployment on edge devices. The use of language heuristics also enhances performance across different genres, from technical reports to social media content. Nevertheless, domain-specific tuning is often required to achieve optimal results in specialized fields such as legal or medical domains, and the evaluation revealed issues with handling ambiguous pronouns, which may reduce coherence in lengthy documents.

3. Proposed Novel Ensemble Learning-Based ATS Framework

In the proposed ATS model, the ensemble methods, which include HIN-MELM-AE, SAE, VAE, and DBERT, are used to perform ATS. In addition, the newly improved DePori algorithm is employed to perform slang identification and filtering. The structural diagram of the proposed model is displayed in Fig. 1.

3.1. Text Document

In the first step, the text documents are obtained from the DUC 2004 dataset, and they are used as input to the proposed system. The total *m* number of text documents (N_M^{text}) is expressed as:

$$N_M^{text} \to N_1^{text}, N_2^{text}, N_3^{text}, \ldots\ldots, N_m^{text}$$
$$where\ M = (1, 2, \ldots (m) \tag{1}$$

where $M = (1,2,\ldots,m)$ indicates the total number of (N_M^{text}).

3.2. PreProcessing

Next, the (N_M^{text}) are preprocessed (or cleaned) to remove textual noise and improve the incomplete text. The proposed model contains preprocessing operations such as standard NLP procedures, removal of URLs, symbols, and emojis, elimination of metadata, and language identification, followed by translation when necessary.

3.2.1. NLP operations

In the proposed model, NLP operations, such as stop-word removal, stemming, and lemmatization, are performed for the (N_M^{text}). First, the stop-word removal eliminates the common or unwanted words (e.g., "the," "and," "is," and "of") from the text documents (N_M^{text}). Thus, the stop-word–removed text is indicated as ς.

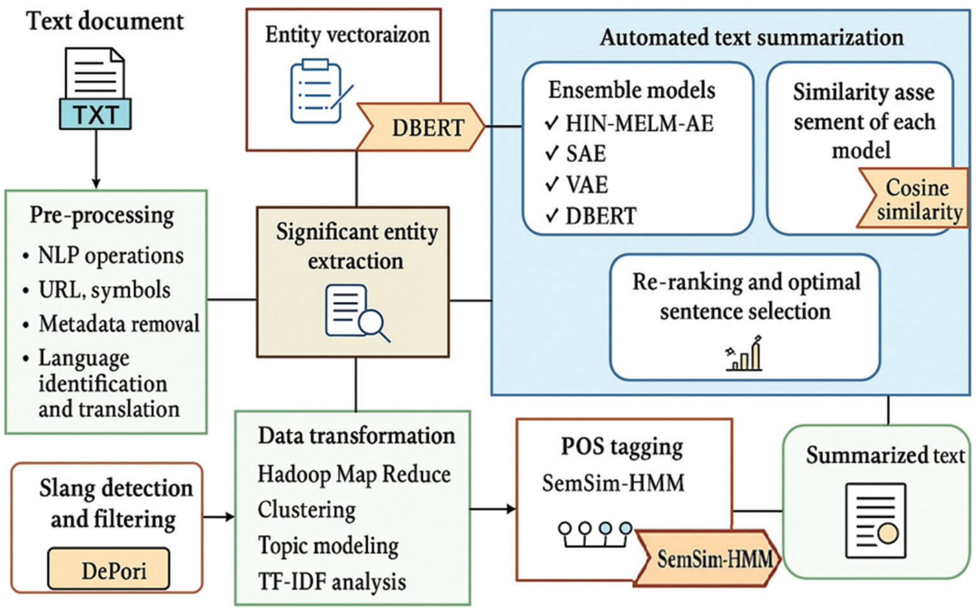

Fig. 1. Structural diagram of the proposed model
Abbreviations: DBERT: distilled bidirectional encoder representations from transformers; DePori: Dehghani poor-and-rich optimization algorithm; HIN-MELM-AE: Hyperfan-IN multilayer extreme learning machine-based autoencoder; NLP: Natural language processing; POS: Parts of speech; SAE: Stack autoencoder; SemSim-HMM: Sememe similarity-induced hidden Markov model; TF–IDF: Term frequency–inverse document frequency; URL: Uniform resource locator; VAE: Variational autoencoder

Next, the last few characters from are removed by the stemming process, which is represented as Ξ. Finally, the lemmatization process converts the word into its meaningful base form. Thus, the lemmatized text is denoted as λ.

3.2.2. URL, symbol, emotion, and metadata removal

Next, the URL, symbols (e.g., @, #, and/), and emotions are removed from the lemmatized text (λ), which is expressed as δ. Then, the metadata removal is performed on the URL-, symbol-, and emotion-removed text (δ). Here, the hypertext markup language tags are removed; thus, the metadata-removed text is denoted as Γ.

3.2.3. Language identification and translation

After that, the language of the Γ is identified, and the text is translated into the English language. Thus, the English-translated text is defined as ε.

Finally, the preprocessed text ($P_{\partial}^{\bullet}$) is expressed as:

$$P_{\partial}^{\bullet}(\varepsilon) \xrightarrow{Preprocessed} \left[P_1^{\bullet} + P_2^{\bullet} + P_3^{\bullet} + \ldots\ldots + P_D^{\bullet}\right] \quad (2)$$

where $P_D^{\bullet}$ specifies the D^{th} preprocessed text.

3.3. Slang Identification and Filtering

After preprocessing, slang recognition and filtering are performed. Words or expressions in languages other than the source language that are present in the text documents are identified and removed to make the text summarization process more accurate. The improved DePori algorithm is employed for slang identification and filtering. The poor and rich optimization (Pori) algorithm effectively differentiates words into slang and normal words based on the discrimination between poor and rich. Nevertheless, Pori suffers from low convergence speed and local optima issues while solving very complex optimization problems. To address these issues, the Dehghani method (DM) is incorporated into Pori to determine the best member of the population by utilizing information on population positions.

3.3.1. Initialization

Primarily, the initial population is created randomly with uniform distribution between upper-bound and lower-bound values. Here, the initialized population is considered as the preprocessed text ($P_{\partial}^{\bullet}$). In DePori, the main population contains two subpopulations: poor and rich. The main population (J_{main}) is defined as follows,

$$J_{main}\left(P_{\partial}^{\bullet}\right) = J_p + J_r \tag{3}$$

where J_p indicates the initialized population of poor, and J_r denotes the initialized population of rich. Here, (J_{main}) is stored in ascending order, where the first part corresponds to the rich population and the second part corresponds to the poor population. In DePori, all rich population members have better positions than the poor, which is expressed as:

$$asc_1 < asc_2 < asc_3 < \cdots < asc_o < asc_{o+1} < asc_{o+2} < asc_{o+3} < \cdots < asc_{\phi} \tag{4}$$

where asc_o specifies the total number of rich populations, and asc_{ϕ} implies the total number of the main population.

3.3.2. Fitness function

Afterward, the fitness function is calculated based on the lexicon dictionary words (*Lex*) to differentiate the slang and non-slang words. Thus, the fitness ($\Im it$) is defined as:

$$Fit = Lex \times Fit(J_{main}) \tag{5}$$

$$J_{best} = Best(Fit) \tag{6}$$

The best fitness value (*Fit*) obtained corresponds to the best candidate solution J_{best}.

3.3.3. Updating the position of each rich population member

Here, the position of every rich population member is changed. The DM is used to locate the best member of the population using information on population positions, thereby improving the performance of the optimization algorithm. In addition, DM ensures that all members of the population, even the worst, can contribute to population development. Thus, the position update of each rich population member based on DM is given by:

$$\overrightarrow{J_{r,n}^{new}} = \overrightarrow{J_{r,n}^{old}} + I_{DM}^{r}\left[\overrightarrow{J_{r,n}^{old}} - \overrightarrow{J_{p,best}^{old}}\right] \tag{7}$$

$$I_{DM}^{r} = \left[i_{best}^{1} \cdots i_{n}^{r} \cdots i_{best}^{p}\right] \tag{8}$$

where

(i) $\overrightarrow{J_{r,n}^{new}}$ specifies the new value of the rich population's nth position.

(ii) $\overrightarrow{J_{r,n}^{old}}$ denotes the present value of the rich population's nth position.

(iii) I_{DM}^{r} implies the DM for the rich population.

(iv) $\overrightarrow{J_{p,best}^{old}}$ represents the best member of the poor population's present position.

(v) i_n^r defines the current value of the rich population's nth position based on I_{DM}^{r}.

(vi) i_{best}^{p} signifies the best member of the poor population's present position based on I_{DM}^{r}.

In fact, $\overrightarrow{J_{p,best}^{old}}$ is the best member of the poor population; when a rich population member increases the distance from $\overrightarrow{J_{p,best}^{old}}$, its distance from all poor population members is also increased.

3.3.4. Updating the position of each poor population member

The position update of each poor population member is given in Eq. (9). Here, DM is employed to locate the best member of the population utilizing information on population positions.

$$\overrightarrow{J_{p,n}^{new}} = \overrightarrow{J_{p,n}^{old}} + \left[I_{DM}^{p} - \overrightarrow{J_{p,n}^{old}}\right] \tag{9}$$

$$I_{DM}^{p} = \left[i_{r,best}^{1} \cdots i_{r,mean}^{old} \cdots i_{r,worst}^{old}\right] \tag{10}$$

where:

(i) $\overrightarrow{J_{p,n}^{new}}$ specifies the new value of the poor population member's nth position.

(ii) $J_{p,n}^{old}$ indicates the present value of the poor population's nth position.

(iii) I_{DM}^{p} represents DM for the poor population.

(iv) $i_{r,best}^{1}$ is the best member of the rich population based on I_{DM}^{p}.

(v) $i_{r,mean}^{old}$ signifies the average position of rich population members based on I_{DM}^{p}.

(vi) $i_{r,worst}^{old}$ represents the worst position of rich population members based on I_{DM}^{p}.

Getting rich varies for each person, where the I_{DM} causes changes in their positions.

3.3.5. Mutation

The mutation for each rich and poor population is formulated as:

$$Mu = \begin{cases} if\ ran < \Pr_{mut} & \overrightarrow{J_{r,n}^{new}} = \overrightarrow{J_{r,n}^{new}} + ran \\ if\ ran < \Pr_{mut} & \overrightarrow{J_{p,n}^{new}} = \overrightarrow{J_{p,n}^{new}} + ran \end{cases} \tag{11}$$

where M_u indicates the mutation, Pr_{mut} denotes the mutation probability, and *ran* represents the value obtained from the normal distribution with an average of 0 and variance of 1.

This position updating continues until the best position is obtained. Thus, the total *l* number of non-slang texts (γ_z) is defined as,

$\gamma_z \rightarrow [\gamma_1, \gamma_2, \gamma_3, \ldots\ldots s\gamma_l]$ Here $z = (1 \text{ to } l)$ (12)

where γ_1 specifies the first (γ_z) and γ_l indicates the last (γ_z). The pseudocode for the DePori algorithm is depicted below:

After identifying and filtering, the slang (γ_z) is transformed for effective ATS.

3.4. Data Transformation

Next, data transformation is performed on (γ_z) by employing the Hadoop MapReduce for handling large volumes of text and multiple documents effectively. Here, the data transformation comprises steps such

Algorithm 1. Improved pseudocode for Dehghani's poor and rich optimization algorithm

```
Input: Preprocessed text $\left(\wp_{\partial}^{\bullet}\right)$
Output: Non-slang text $\left(\gamma_z\right)$
Begin
    Initialize main population
        $J_{main}\left(\wp_{\partial}^{\bullet}\right) = J_p + J_r$
    Store $J_{main}$ in ascending order
        $asc_1 < asc_2 < asc_3 < \cdots < asc_o < asc_{o+1} < asc_{o+2} < asc_{o+3} < \cdots < asc_{\phi}$
While $iter \le iter_{\max}$
        For each $\left(J_{main}\right)$
            Compute fitness function
                $\Im it = Lex * \Im it\left(J_{main}\right)$
Update position of each rich population member
        $\overrightarrow{J_{r,n}^{new}} = \overrightarrow{J_{r,n}^{old}} + I_{DM}^{r}\left[\overrightarrow{J_{r,n}^{old}} - \overrightarrow{J_{p,best}^{old}}\right]$
  Discover Dehghani Method for rich population
        $I_{DM}^{r} = \left[i_{best}^{1} \cdots i_{n}^{r} \cdots i_{best}^{p}\right]$
  Update position of each poor population member
        $\overrightarrow{J_{p,n}^{new}} = \overrightarrow{J_{p,n}^{old}} + \left[I_{DM}^{p} - \overrightarrow{J_{p,n}^{old}}\right]$
  Estimate Dehghani Method for poor population
        $I_{DM}^{p} = \left[i_{r,best}^{1} \cdots i_{r,mean}^{old} \cdots i_{r,worst}^{old}\right]$
            Find mutation
        $Mu = \begin{cases} if\ ran < \Pr_{mut} & \overrightarrow{J_{r,n}^{new}} = \overrightarrow{J_{r,n}^{new}} + ran \\ if\ ran < \Pr_{mut} & \overrightarrow{J_{p,n}^{new}} = \overrightarrow{J_{p,n}^{new}} + ran \end{cases}$
        End For
    End While
    Obtain Non-slang text $\left(\gamma_z\right)$
End
```

as clustering, topic modeling, TF-IDF analysis, and frequent term selection.

3.4.1. Clustering

First, the clustering process is applied (γ_z) to create the text document clusters. Here, the InS-FCM model is employed for clustering. In general, FCM performs better for data with complex structures or overlapping class boundaries. If outliers exist in the data, they affect the clustering outcomes significantly. If the problem of outliers is avoided, the results of clustering will be more accurate. To solve this problem, the info-squared technique is incorporated into FCM, effectively mitigating the effect of outliers. The clustered data ($Z_{r'}^{cl}$) is formulated as:

$$Z_{r'}^{cl} \Rightarrow \left(Z_1^{cl}, Z_2^{cl}, Z_3^{cl}, \ldots\ldots, Z_{J'}^{cl}\right) \tag{13}$$

Where $r' = (1,2,\ldots,J')$ specifies the clustered data.

3.4.2. Topic modeling

Next, the topic modeling is applied to each clustered of data ($Z_{r'}^{cl}$) for creating the cluster topics and terms belonging to each cluster topic. Here, the LDA approach is used for topic modeling. LDA effectively identifies the hidden topic structure in a set of documents. This can be defined as:

$$\log j(Z \mid a,b) \ge E_{ELB} \left[\log f\left(Z,V,\varpi,\varphi \mid a,b\right) - \log k\left(\varpi,\varphi,V\right)\right] \tag{14}$$

Thus, a good approximation of the posterior distribution is determined by iteratively updating the σ and v. Finally, the topic modeling outcomes are represented as T_c.

3.4.3. Term frequency–inverse document frequency analysis

The TF-IDF analysis is performed for T_c to identify the globally frequent terms from the collection of multiple text documents. The TF-IDF analysis reduces the computational complexity of the model. First, the TF(κ) is calculated by the frequency of the specific term present in T_c:

$$\kappa = \left\| \Omega\left(\tilde{d}\right) \middle/ \tau\left(\tilde{d}, T_c\right) \right\| \tag{15}$$

where $\Omega\left(\tilde{d}\right)$ denotes the number of times, the term $\tilde{d}$ is present in T_c, and $\tau\left(\tilde{d}, T_c\right)$ represents the total number of terms in T_c. The IDF (U) is known as the frequency of tokens (*to*) containing the term. Thus, the IDF is defined as:

$$U = \log\left|\frac{\Omega(to)}{to(\tilde{d})}\right| \tag{16}$$

where *Ω*(to) denotes the number of tokens present in $T_{\tilde{c}}$, and $to(\tilde{d})$ denotes the tokens that contain the term $\tilde{d}$. Next, the TF-IDF can be calculated by multiplying the TF and IDF values:

$$\tilde{F} \Rightarrow (\kappa \times U) \tag{17}$$

where $\tilde{F}$ indicates the obtained TF-IDF score.

3.4.4. Frequent term selection

After the TF-IDF analysis ($\tilde{F}$), the frequent terms are selected. Here, sentence filtering is performed on each individual input text document based on the frequent and semantic similar terms generated from the previous stage, which is indicated as c'. Finally, the total λ' number of transformed data ($Y_{q'}$) is given by:

$$Y_{q'} \rightarrow [Y_1 + Y_2 + Y_3 + \cdots \ldots Y_{\lambda'}] \tag{18}$$

where $Y_{\lambda'}$ specifies the $\lambda^{\prime th}$ transformed data.

3.5. POS Tagging

Then, POS is tagged from the ($Y_{q'}$). The SemSim-HMM is used for POS tagging. HMM effectively reduces ambiguities in the sentences and improves tagging accuracy. However, HMM achieves low accuracy for unknown words because the emission probability tends to be zero in such cases. Therefore, the sememe similarity is incorporated with HMM, which evaluates the semantic relationship between sentences through the horizontal semantic relation.

Finally, the tagged POS (Z_ℓ^{POS})is expressed as:

$$Z_\ell^{POS} \rightarrow \left[Z_1^{POS} + Z_2^{POS} + \ldots\ldots + Z_{\rho'}^{POS}\right]$$
$$where\ \ell = (1, 2, ss\rho') \tag{19}$$

where *ρ'* denotes the number of tagged POS. The pseudocode for SemSim-HMM is depicted below:

Thus, SemSim-HMM effectively performs POS tagging.

3.6. Significant Entity Extraction

Next, the significant entities, namely statistical and linguistic features, including code quantity principle, noun and verb phrases, content words,

Algorithm 2. Pseudocode for sememe similarity-induced hidden Markov model

```
Input: Transformed data $(Y_{q'})$
Output: Tagged POS $(\Im_\ell^{POS})$
Begin
    Initialize $(Y_{q'})$
    For each $(Y_{q'})$
        Compute set of $\tilde{M}$ hidden state $(H_{ab})$
        Estimate a sequence of $\tilde{P}$ observations $(O_{pq})$
        Derive transition probability matrix
            $W[l',m'] = \Pr(\mu_{t'+1} = H_v \mid \mu_{t'} = H_u)$
        Evaluate emission probability
            $\tilde{V}[v,l'] = \Pr(\varphi'_{t'} = O_{l'} \mid \mu_{t'} = H_v) + \Im'$
        Discover sememe similarity
            $\Im' = \begin{bmatrix} O_1\ Simil\ (H_1,H_2) + O_2\ Simil(H_1,H_2) \\ +O_3\ Simil\ (H_1,H_2) \end{bmatrix}$
        Find initial state probability
            $\theta_u = \Pr(\mu_1 = H_u)$
        Compute SemSim-HMM
            $\delta = (H, O, W, \tilde{V}, \theta)$
    End for
    Obtain tagged POS $(\Im_\ell^{POS})$
End
```

proper nouns, cue-phrase, biased words, positive keywords, negative keywords, and sentence length, are extracted from the ($Y_{q'}$) and (Z_ℓ^{POS}). The extracted significant entities ($\beta^\circ_{\varpi'}$) are given by:

$$\beta^\circ_{\varpi'} \Rightarrow \left[\beta^\circ_1, \beta^\circ_2, \beta^\circ_3, \ldots\ldots v\beta^\circ_{\delta'}\right] \tag{20}$$

where $\beta^\circ_{\delta'}$ denotes the number of ($\beta^\circ_{\varpi'}$).

3.7. Entity Vectorization

After that, ($\beta^\circ_{\varpi'}$)undergoes entity vectorization for effective ATS. In this case, the DBERT algorithm is used for entity vectorization. DBERT efficiently handles complex language structures and captures the contextual meaning of words within a sentence.

Initially, ($\beta^\circ_{\varpi'}$) is tokenized into υ tokens $\tilde{\tau}_{to}$. The $\tilde{\tau}_{to}$ is then fed into the DBERT model to obtain *i*' number of token embeddings $\tilde{e}_{to}$.

The next step is to obtain a fixed-sized sentence embedding by applying a pooling operation on the

DBERT model. Thus, the sentence embedding (S) is calculated as:

$$S = \frac{1}{\varepsilon '}\sum_{em=1}^{\varepsilon '}\tilde{C}_{em} \tag{21}$$

where ε' indicates the number of contextualized embeddings.

Subsequently, $\tilde{\tau}_{to}$ are handled by the DBERT model, and it produces contextualized embeddings $\tilde{C}_{em}$.

Then, the embeddings of the two sentences S_1 and S_2 are obtained by using the Siamese network structure. This can be written as:

$$\phi' = DBERT(S_1)$$
$$\kappa' = DBERT(S_2) \tag{22}$$

where ϕ' and κ' represent the embeddings. Then, the cosine similarity (Sim) is evaluated between two embeddings ϕ' and κ', which is given by:

$$Sim(\phi ',\kappa ') = \frac{\phi '\cdot \kappa '}{\|\phi '\| \ \|\kappa '\|} \tag{23}$$

Next, a suitable loss function is chosen for training the BERT model. Thus, the contractive loss function ($Loss$) is defined as:

$$Loss(\phi ',\kappa ',\nu ') = \nu ' \cdot Sim(\phi ',\kappa ') + (1-\nu ')\cdot \max(0, ma - Sim(\phi ',\kappa ')) \tag{24}$$

where ma signifies the hyperparameter, and if S_1 and S_2 are similar, then v' is 1; otherwise, v' is 0. Finally, the entity vectorization outcomes are indicated as $X_{\Gamma'}$.

3.8. ATS

Then, ($X_{\Gamma'}$) are subjected as input to the ATS. In this stage, ensemble-based voting, similarity evaluation, and re-ranking with optimal sentence selection are performed to obtain the summarized text.

3.8.1. Ensemble models

The ATS is performed based on the ($X_{\Gamma'}$). In the proposed model, the HIN-MELM-AE is employed to generate the summary text. In general, AEs are used to convert a given input into a lower-dimensional representation. The decoder reconstructs the input from the encoded version. However, AEs are computationally expensive and provide poor performance for text summarization.

To address these issues, multilayer extreme learning machine (ELM)-based AEs (MELM-AEs) are employed for ATS. ELM offers several advantages, such as fast training speed and versatility in handling diverse data types. MELM-AE effectively learns feature representations by adopting a singular value decomposition. Hyperfan-IN initialization is used in MELM-AE for generating the weights and biases of the network to enhance the learning of machine.

Along with the proposed HIN-MELM-AE, SAE, VAE, and DBERT are employed for ATS. Fig. 2 presents the structural visualization of the ensemble models.

3.8.1.1. Hyperfan-IN multilayer extreme learning machine-based AE

The HIN-MELM-AE consists of the input layer, encoder, MELM, decoder, and output layer.

(i) Hyperfan-IN initialization: Here, the weights (ω) and biases (B) of the network are generated by employing Hyperfan-IN initialization, which improves the learning efficiency of ELM. This is expressed as:

$$\omega = \frac{2^1 \operatorname{Re}LU}{2\upsilon '_{\theta'}\, Var\left(e[2]^{X_{\Gamma'}}\right)} \tag{25}$$

$$B = \frac{2^1 \operatorname{Re}LU}{2\upsilon '_{\theta'}\, Var\left(e[2]^{X_{\Gamma'}}\right)} \tag{26}$$

where $\upsilon'_{\theta'}$ specifies the constant, and $Re\ L\ U$ denotes the rectified linear unit function.

(ii) Encoder phase: Primarily, $X_{\Gamma'}$ are subjected to the input layer I_y, which forwards the data to the multiple hidden layers (L_g). This can be determined as:

$$L_g = \xi(\omega\ I_y + B) \tag{27}$$

where ξ indicates the sigmoid activation function.

(iii) Decoder phase: Finally, in the decoder phase, the output layer reconstructs the summarized text. Thus, the output layer operation is defined by:

$$N_y(X_{\Gamma'}) = (\omega\ A_g + B) \tag{28}$$

Thus, the obtained sentences are denoted as λ_{Γ}.

3.8.1.2. SAE

SAE effectively learns the codings of unlabeled data. The SAE contains two parts: an encoder and a decoder.

Firstly, the $X_{\Gamma'}$ is subjected as an input to the input layer In_{AE} which forwards the data to the encoder. Then, the In_{AE} is mapped by the encoder to a latent space representation, which is performed by utilizing

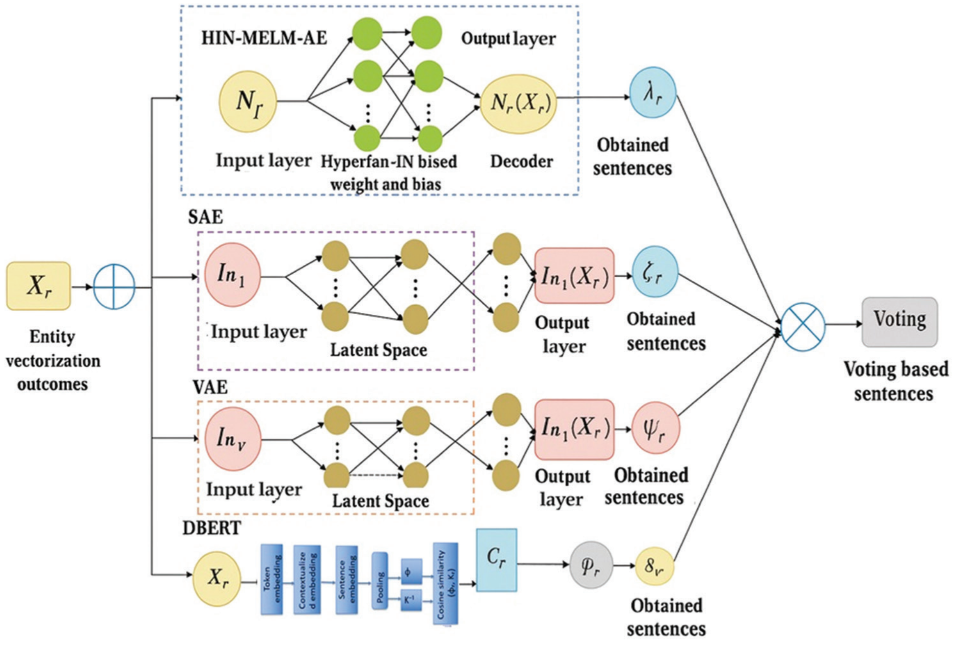

Fig. 2. Structural representation of ensemble models
Abbreviations: DBERT: distilled bidirectional encoder representations from transformers; HIN-MELM-AE: Hyperfan-IN multilayer extreme learning machine-based autoencoder; SAE: Stack autoencoder; VAE: Variational autoencoder

one or more hidden layers. Thus, the encoder operation is written as:

$$\hbar = \sigma\left(\tilde{W}_{en} In_{AE} + \tilde{b}_{en}\right) \quad (29)$$

where $\tilde{W}$ signifies the weight of SAE, $\tilde{b}$ indicates the bias term of SAE, σ denotes the activation function, and $\hbar$ represents the latent space representation. Then, the $\hbar$ is mapped back to the original input space by the decoder. Thus, the decoder operation is given by:

$$In_{AE} = \sigma\left(\tilde{W}_{de}\hbar + \tilde{b}_{de}\right) \quad (30)$$

Next, the decoded information In_{AE} is passed to the output layer, which delivers the outcomes. Eventually, the obtained sentences are represented as $\chi_{n''}$.

3.8.1.3. VAE

A VAE combines a neural network with variational inference. The VAE consists of an encoder, a latent state distribution, and a decoder.

Initially, the $X_{\Gamma'}$ is fed into the input layer $\tilde{I}_{VAE}$, which forwards the data to the encoder. Then, the $\tilde{I}_{VAE}$ is mapped by the encoder to a latent space distribution (la). Thus, the encoder operation is given by:

$$\vartheta_{pa}\left(la \mid \tilde{I}_{VAE}\right) = \tilde{G}\left(la; \mu'_{pa}(\tilde{I}_{VAE}), \alpha_{pa}\left(\tilde{I}_{VAE}\right)^2\right) \quad (31)$$

where $\vartheta_{pa}\left(la \mid \tilde{I}_{VAE}\right)$ specifies the posterior distribution, signifies the encoder parameters, $\tilde{G}$ implies the Gaussian distribution, and μ' and α are the mean and standard deviation, respectively. The latent variable is determined as follows:

$$\tilde{E}(la) = \tilde{G}(0, Id) \quad (32)$$

where Id is the identity matrix. Then, is mapped back to the original input space by the decoder. This is given by:

$$\tilde{E}_{co}\left(\tilde{I}_{VAE} \mid la\right) = \tilde{G}\left(\tilde{I}_{VAE};\ \mu_{co}(la),\ \sigma_{co}(la)^2\right) \quad (33)$$

where $\tilde{E}_{co}\left(\tilde{I}_{VAE} \mid la\right)$ defines the likelihood, and co signifies the decoder parameters. Then, the decoded information $\tilde{I}_{VAE}$ is passed to the output layer, which delivers the outcomes. Eventually, the obtained sentences are represented as $\psi_{s'}$.

3.8.1.4. DBERT

The $X_{\Gamma'}$ is fed to the DBERT model for ATS. The processes involved in the DBERT model are explained in Section 3.7. Lastly, the obtained sentences are indicated as ζ'_{mn}.

Finally, the voting-based sentences ($S_{\partial'}^{vot}$) are obtained, which are represented as:

$$S_{\partial'}^{vot} \rightarrow S_1^{vot}, S_2^{vot}, S_3^{vot}, \ldots\ldots, S_{\tilde{\rho}}^{vot} \; where \; \partial' = (1, 2, bt\tilde{\rho}) \tag{34}$$

where $\partial' = (1, 2, \ldots, \tilde{\rho})$ specifies the number of $S_{\partial'}^{vot}$.

3.8.2. Similarity evaluation of each model

A similarity between sentences is estimated from the voting-based sentences ($S_{\partial'}^{vot}$). Here, cosine similarity is used to calculate the similarity between sentences. Thus, the cosine similarity (*Cos*) is given by:

$$Cos = \frac{S_1^{vot} \cdot S_2^{vot}}{\left\| S_1^{vot} \right\| \left\| S_2^{vot} \right\|} \tag{35}$$

The cosine similarity is calculated for all voting-based sentences. Thus, the evaluated similarity between all the voting-based sentences is indicated as $\Theta_{l'}$.

3.8.3. Re-ranking and optimal sentence selection

After similarity evaluation ($\Theta_{l'}$), the voting-based fusion is performed. Then, the sentences are rearranged, and optimal sentences are selected to obtain the summarized text. Finally, the obtained summarized text ($R_{\tau'}^{text}$) is formulated as:

$$R_{\tau'}^{text} \Rightarrow \left[R_1^{text}, R_2^{text}, R_3^{text}, \ldots\ldots, R_{\mu'}^{text} \right] \tag{36}$$

Where $R_{\mu'}^{text}$ specifies the μ'^{th} summarized text. Thus, the proposed model effectively performs ATS with slang filtering.

4. Materials and Methods

4.1. Software Requirements

The proposed model was implemented in the Python platform. Python is a widely used general-purpose and high-level programming language designed to emphasize code readability. The syntax of Python allows developers to define concepts in fewer lines of code. In addition, Python efficiently integrates with systems and enables rapid development. Python is a multipurpose programming language and has been employed in many applications, such as machine learning, scientific computing, and automation. Furthermore, Python's extensive standard library is equipped with modules and functions for a variety of frequently occurring tasks.

4.2. Hardware Requirements

The hardware requirements for the proposed model are as follows:

(i) Processor: Intel Core i5/Core i7
(ii) Central processing unit speed: 3.20 GHz
(iii) Operating system: Windows 10
(iv) System type: 64-bit
(v) Random-access memory: 4 GB.

4.3. Dataset Description

The DUC 2004 dataset was used to assess the proposed model. This dataset was obtained from publicly available sources. The DUC 2004 dataset consists of 1,358 text documents, of which 80% (*n* = 1,087) were employed for training, and the remaining 20% (*n* = 271) were employed for testing.

5. Results and Discussion

In this section, the performance analysis and comparative assessment of the proposed model are discussed to demonstrate the model's reliability.

5.1. Performance Evaluation of the Proposed HIN-MELM-AE Model

The performance of the proposed model and existing techniques, such as AE, sentence bidirectional encoder representations from transformers (SBERT), LSTM, and RNN, was evaluated to demonstrate the model's reliability.

Fig. 3 illustrates the graphical analysis of the proposed HIN-MELM-AE and existing techniques in terms of accuracy, precision, recall, F-measure, sensitivity, ROUGE 1, and ROUGE 2. The proposed HIN-MELM-AE achieved high accuracy, precision, recall, F-measure, sensitivity, ROUGE 1, and ROUGE 2 of 97.92%, 99.16%, 98.01%, 98.24%, 98.01%, 0.935145, and 0.911371, respectively.

In contrast, the existing techniques achieved lower average accuracy, precision, recall, F-measure, sensitivity, ROUGE 1, and ROUGE 2 of 91.59%, 93.05%, 90.21%, 91.59%, 90.21%, 0.833706, and 0.761878, respectively. This indicates that the proposed HIN-MELM-AE utilized the MELM-based AE to achieve superior performance for text summarization.

A comparative assessment of the proposed model and existing techniques regarding execution time and error is presented in Table 1. Here, the proposed model utilizes the Hyperfan-IN initialization for generating the weights and biases of the network, which enhances the learning process. The proposed HIN-MELM-AE obtained a low execution time and error of 51,015 ms

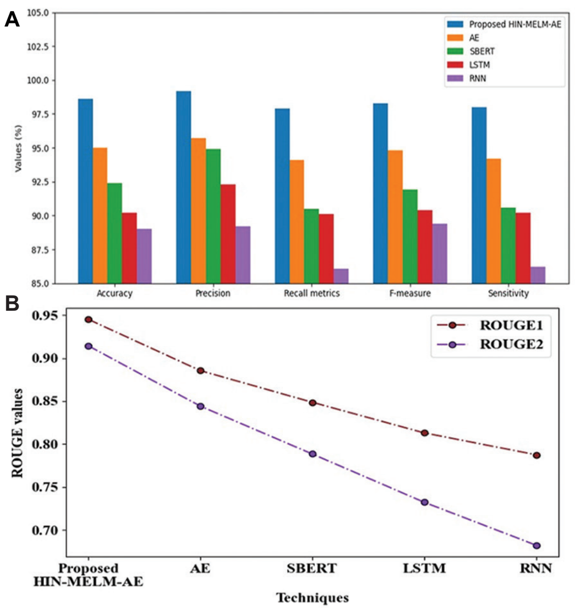

Fig. 3. Graphical analysis in terms of (A) accuracy, precision, recall, F-measure, and sensitivity, and (B) ROUGE 1 and ROUGE 2
Abbreviations: AE: Autoencoder; IN-MELM-AE: Hyperfan-IN multilayer extreme learning machine-based autoencoder; LSTM: Long short-term memory; RNN: Recurrent neural network; ROUGE: Recall-oriented understudy gisting evaluation; SBERT: Sentence bidirectional encoder representations from transformers

and 0.011766, respectively. However, the existing SBERT and RNN achieved a higher execution time of 62,008 ms and 83,012 ms, respectively. In addition, the existing AE and LSTM exhibited higher errors of 0.051064 and 0.097872, respectively. These results indicate that the proposed model performs better for text summarization.

5.2. Performance Estimation of Ensemble Models

The performance estimation of ensemble models was conducted to assess the model's reliability.

Fig. 4 displays the performance evaluation of ensemble models in terms of summary size versus recall and summary size versus ROUGE. The proposed HIN-MELM-AE achieved a high recall and ROUGE of 0.9857 and 0.9132, respectively, for a summary size of 50. On the other hand, AE, VAE, and SBERT achieved a low recall of 0.8942, 0.8215, and 0.7745, respectively, for a summary size of 50.

In addition, AE, VAE, and SBERT achieved lower ROUGE values of 0.8412, 0.7916, and 0.7195, respectively, for a summary size of 50. This indicates that the proposed HIN-MELM-AE achieved superior

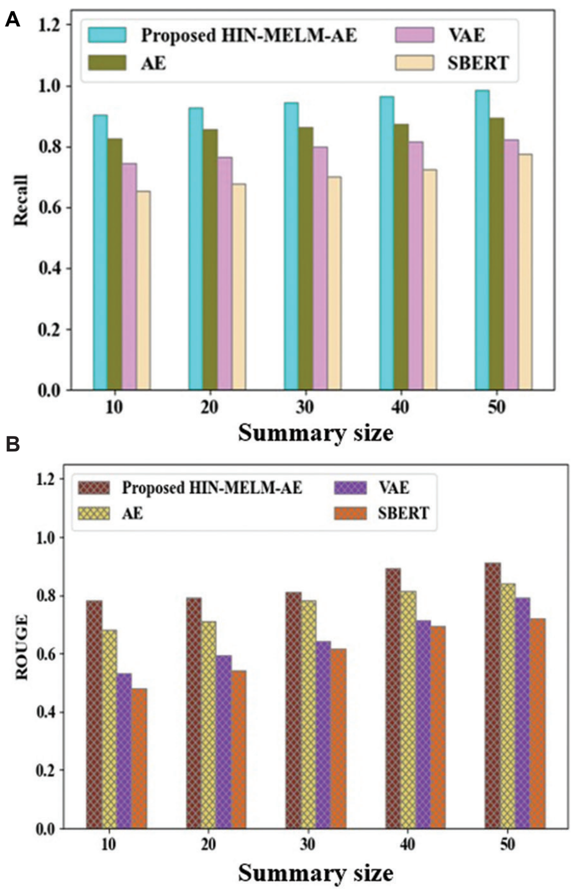

Fig. 4. Performance evaluation of ensemble models in terms of (A) summary size versus recall and (B) summary size versus ROUGE
Abbreviations: AE: Autoencoder; HIN-MELM-AE: Hyperfan-IN multilayer extreme learning machine-based autoencoder; ROUGE: Recall-oriented understudy gisting evaluation; SBERT: Sentence bidirectional encoder representations from transformers; VAE: Variational autoencoder

Table 1. Comparative assessment of the proposed model and existing techniques

Methods	Execution time (ms)	Error
Proposed HIN-MELM-AE	51,015	0.011766
Existing SAE	57,006	0.051064
Existing SBERT	62,008	0.076596
Existing LSTM	68,018	0.097872
Existing RNN	83,012	0.110638

Abbreviations: HIN-MELM-AE: Hyperfan-IN multilayer extreme learning machine-based autoencoder; LSTM: Long short-term memory; RNN: Recurrent neural network; SAE: Stack autoencoder; SBERT: Sentence bidirectional encoder representations from transformers.

performance in text summarization due to the use of MELM-based AE and Hyperfan-IN initialization.

5.3. Performance Assessment of the Proposed DePori

The performance assessment of the proposed DePori was compared with conventional techniques, such as Pori, honey badger algorithm (HBO), fennec fox optimization (FFO), and sail fish optimizer (SFO).

Fig. 5 displays the comparison of fitness versus iteration for the proposed model and traditional methods, including Pori, HBO, FFO, and SFO. The proposed model employed DM to locate the best member of the population using information on population location. The proposed DePori achieved a high fitness of 88.12 in the 10th iteration and 96.37 in the 40th iteration.

In contrast, HBO achieved lower fitness values of 85.26 in the 10th iteration and 91.26 in the 40th iteration. Other existing methods also obtained low fitness values. This indicates that the proposed model effectively performs slang identification and filtering.

5.4. Performance Analysis of the Proposed InS-FCM

The performance analysis of the proposed InS-FCM and existing techniques was performed to assess the model's performance.

Table 2 presents the performance validation of the proposed InS-FCM and conventional methods, including FCM, K-means, K-medoids, and balanced iterative reducing and clustering using hierarchies (BIRCH) in terms of clustering time and silhouette score. The proposed InS-FCM obtained a low clustering time and a high silhouette score of 6,134 s and 0.9085, respectively. On the other hand, the existing techniques achieved a high mean clustering time of 13,382.25 s and a low silhouette score of 0.7737. In this approach, the info-squared method is combined with FCM, which effectively mitigates outlier-related issues.

Fig. 6 illustrates the clustering accuracy of the proposed model and the existing techniques. The proposed InS-FCM achieved a high clustering accuracy of 96.90%. On the other hand, the existing techniques, such as FCM, K-means, K-medoids, and BIRCH, obtained a low clustering accuracy of 93.89%, 92.49%, 90.35%, and 87.32%, respectively. These results indicate the superiority of the model.

5.5. Comparative Analysis of the Proposed Model

The comparative analysis of the proposed model and related works was performed to assess the model's reliability.

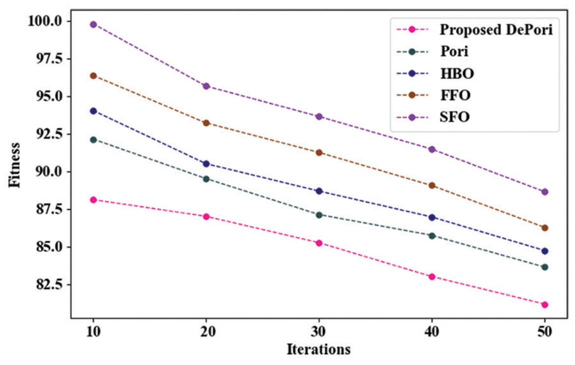

Fig. 5. Fitness versus iteration analysis. Abbreviations: DePori: Dehghani poor and rich optimization algorithm; FFO: Fennex fox optimization; HBO: Honey badger algorithm; Pori: Poor and rich optimization algorithm; SFO: Sail fish optimizer

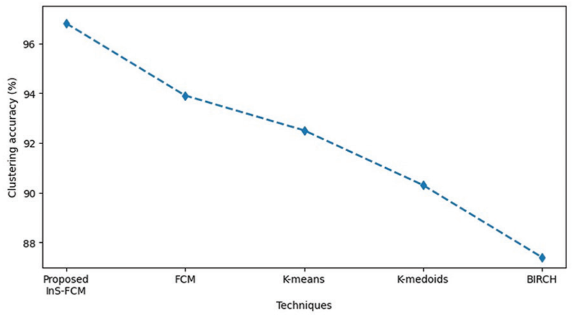

Fig. 6. Graphical representation of clustering accuracy. Abbreviations: BIRCH: Balanced iterative reducing and clustering using hierarchies; FCM: Fuzzy C-means clustering; InS-FCM: Info-squared fuzzy C-means clustering

Table 2. Performance validation in terms of clustering time and silhouette score

Techniques	Clustering time (s)	Silhouette score
Proposed InS-FCM	6,134	0.9085
FCM	9,219	0.8475
K-means	12,237	0.7954
K-Medoids	14,389	0.7542
BIRCH	17,684	0.6978

Abbreviations: BIRCH: Balanced iterative reducing and clustering using hierarchies; FCM: Fuzzy C-means clustering; InS-FCM: Info-squared fuzzy C-means clustering.

Table 3 presents the comparative analysis of the proposed HIN-MELM-AE model and related works. The proposed HIN-MELM-AE model, which utilizes the MELM-based AE and Hyperfan-IN for initialization, achieved a high ROUGE 1 and ROUGE 2

Table 3. Comparative analysis

Techniques	Dataset	ROUGE 1	ROUGE 2	References
HIN-MELM-AE	DUC 2004 dataset	0.94014	0.90437	Proposed model
FA		0.4782	0.2295	Tomer and Kumar (2022)
DNN		0.42813	0.23668	Onan and Alhumyani (2024a)
USE transformer		0.4286	0.1425	Lamsiyah et al. (2021)
Karci's summarization approach		0.58669	0.31804	Hark & Karcı (2020)

Abbreviations: DNN: Deep neural network; FA: Firefly algorithm; HIN-MELM-AE: Hyperfan-IN multilayer extreme learning machine-based autoencoder; MMR: Maximal marginal relevance; ROUGE: Recall-oriented understudy gisting evaluation; USE: Universal sentence encoder.

values of 0.94014 and 0.90437, respectively. However, the existing firefly algorithm achieved a low ROUGE-1 of 0.4782 using the DUC 2004 dataset. Furthermore, the existing maximal marginal relevance method and PageRank algorithm achieved a low ROUGE-1 score of 0.684. Moreover, the existing deep neural network, universal sentence encoder transformer, and Karci summarization approach achieved low ROUGE-1 and ROUGE-2 scores, potentially reflecting their higher computational complexity. These findings indicate that the proposed model performs ATS more effectively.

6. Conclusion

This study presents an improved ensemble learning-based ATS with slang filtering using HIN-MELM-AE and DePori. The DUC 2004 dataset was used to train the proposed model. In this study, processes—such as text document acquisition, preprocessing, slang identification and filtering, data transformation, POS tagging, significant entity extraction, entity vectorization, ensemble models, similarity evaluation of each model, and re-ranking and optimal sentence selection—were performed. The proposed HIN-MELM-AE achieved high accuracy, precision, and recall of 97.92%, 99.16%, and 98.01%, respectively. In addition, the proposed InS-FCM achieved a low clustering time and high clustering accuracy of 6,234 s and 96.90%, respectively. Furthermore, the proposed DePori achieved a high fitness of 88.12 in the 10th iteration and 99.81 in the 50th iteration. These findings indicate that the proposed model performs ATS with slang filtering more effectively. However, the proposed model is limited to summarizing text from documents and does not handle context in videos. Future studies could focus on developing techniques for summarizing context in videos to further enhance the model's applicability.

Acknowledgments

The authors would like to thank the supervisor and mentors for their invaluable guidance, support, and encouragement throughout the research. The authors are also grateful to the institution for offering the essential facilities and resources needed to complete this work.

Funding

None.

Conflict of Interest

The authors declare that they have no competing interests.

Author Contributions

Conceptualization: Sunil Upadhyay
Data curation: Sunil Upadhyay
Methodology: Sunil Upadhyay
Writing – original draft: Sunil Upadhyay
Writing – review & editing: All authors

Availability of Data

The dataset used in this study was obtained from https://www.kaggle.com/datasets/usmanniazi/duc-2004-dataset.

References

Abo-Bakr, H., & Mohamed, S.A. (2023). Automatic multi-documents text summarization by a large-scale sparse multi-objective optimization algorithm. *Complex and Intelligent Systems*, 9(4), 4629–4644 https://doi.org/10.1007/s40747-023-00967-y

Alami Merrouni, Z., Frikh, B., & Ouhbi, B. (2023). EXABSUM: A new text summarization approach for generating extractive and abstractive summaries. *Journal of Big Data*, 10(1), 1–34. https://doi.org/10.1186/s40537-023-00836-y

Alomari, A., Idris, N., Sabri, A.Q.M., & Alsmadi, I. (2022). Deep reinforcement and transfer learning for abstractive text summarization: A review.

Computer Speech and Language, 71, 1–43. https://doi.org/10.1016/j.csl.2021.101276

Awasthi, I., Gupta, K., Bhogal, P.S., Anand, S.S., & Soni, P.K. (2021). Natural Language Processing (NLP) Based Text Summarization-A Survey. In: *Proceedings of the 6th International Conference on Inventive Computation Technologies*, p1310–1317. https://doi.org/10.1109/icict50816.2021.9358703

Belwal, R.C., Rai, S., & Gupta, A. (2021). Text summarization using topic-based vector space model and semantic measure. *Information Processing and Management*, 58(3), 102536. https://doi.org/10.1016/j.ipm.2021.102536

Divya, S., & Sripriya, N. (2025). Semantic based extractive document summarization using deep learning model. *ICTACT Journal on Soft Computing*, 15(4), 3669–3681. https://doi.org/10.21917/ijsc.2025.0509

El-Kassas, W.S., Salama, C.R., Rafea, A.A., & Mohamed, H.K. (2021). Automatic text summarization: A comprehensive survey. *Expert Systems with Applications*, 165, 1–46. https://doi.org/10.1016/j.eswa.2020.113679

Gupta, H., & Patel, M. (2021). Method of Text Summarization Using Lsa and Sentence Based Topic Modelling with Bert. In: *International Conference on Artificial Intelligence and Smart Systems*, p511–517. https://doi.org/10.1109/ICAIS50930.2021.9395976

Haider, M.M., Hossin, M.A., Mahi, H.R., & Arif, H. (2020). Automatic Text Summarization Using Gensim Word2Vec and K-Means Clustering Algorithm. In: *IEEE Region 10 Symposium, TENSYMP 2020*, p283–286. https://doi.org/10.1109/TENSYMP50017.2020.9230670

Hailu, T.T., Yu, J., & Fantaye, T.G. (2020). A framework for word embedding based automatic text summarization and evaluation. *Information (Switzerland)*, 11(2), 1–23. https://doi.org/10.3390/info11020078

Hark, C., & Karci, A. (2020). Karci summarization: A simple and effective approach for automatic text summarization using Karcı entropy. *Information Processing and Management*, 57(3), 1–16. https://doi.org/10.1016/j.ipm.2019.102187

Hassan, A.A., Al-Onazi, B.B., Maashi, M., Darem, A.A., Abunadi, I., & Mahmud, A. (2024). Enhancing extractive text summarization using natural language processing with an optimal deep learning model. *AIMS Mathematics*, 9(5), 12588–12609.

Hernandez-Castaneda, A., Garcia-Hernandez, R.A., & Ledeneva, Y. (2023). Toward the automatic generation of an objective function for extractive text summarization. *IEEE Access*, 11, 51455–51464. https://doi.org/10.1109/access.2023.3279101

Hosseinabadi, S., Kelarestaghi, M., & Eshghi, F. (2022). ISSE: A new iterative sentence scoring and extraction scheme for automatic text summarization. *International Journal of Computers and Applications*, 44(6), 1–6. https://doi.org/10.1080/1206212X.2020.1829844

El-Kassas, Wafaa S., Cherif R. Salama, Ahmed A. Rafea, and Hoda K. Mohamed. (2020). EdgeSumm: Graph-Based framework for automatic text summarization. *Information Processing and Management,* 57(6), 102264. https://doi.org/10.1016/j.ipm.2020.102264.

Jain, M., & Rastogi, H. (2020). Automatic Text Summarization using Soft-Cosine Similarity and Centrality Measures. In: *Proceedings of the 4th International Conference on Electronics, Communication and Aerospace Technology*, p. 1021–1028. https://doi.org/10.1109/iceca49313.2020.9297583

Jiang, J., Zhang, H., Dai, C., Zhao, Q., Feng, H., Ji, Z., et al. (2021). Enhancements of attention-based bidirectional LSTM for hybrid automatic text summarization. *IEEE Access*, 9, 123660–123671. https://doi.org/10.1109/access.2021.3110143

Kouris, P., Alexandridis, G., & Stafylopatis, A. (2024). Text summarization based on semantic graphs: An abstract meaning representation graph-to-text deep learning approach. *Journal of Big Data*, 11(1), 1–39. https://doi.org/10.1186/s40537-024-00950-5

Lamsiyah, S., Mahdaouy, E., Espinasse, B., & Ouatik, A. (2021). An unsupervised method for extractive multi-document summarization based on centroid approach and sentence embeddings. *Expert Systems with Applications*, 167, 114152.

Liu, W., Sun, Y., Yu, B., Wang, H., Peng, Q., Hou, M., et al. (2024). Automatic text summarization method based on improved TextRank algorithm and K-Means clustering. *Knowledge-Based Systems*, 287, 111447. https://doi.org/10.1016/j.knosys.2024.111447

Mandale-Jadhav, A. (2025). Text summarization using natural language processing. *Journal of Electrical Systems*, 20(11), 3410–3417. https://doi.org/10.52783/jes.8095

Manjari, K.U., Rousha, S., Sumanth, D., & Devi, J.S. (2020). Extractive Text Summarization from Web pages using Selenium and TF-IDF algorithm. In: *Proceedings of the 4th International Conference on Trends in Electronics and Informatics*, p648–652. https://doi.org/10.1109/icoei48184.2020.9142938

Muniraj, P., Sabarmathi, K.R., Leelavathi, R., &

Balaji, B.S. (2023). HNTSumm: Hybrid text summarization of transliterated news articles. *International Journal of Intelligent Networks*, 4, 53–61. https://doi.org/10.1016/j.ijin.2023.03.001

Onah, D.F.O., Pang, E.L.L., & El-Haj, M. (2022). A Data-driven Latent Semantic Analysis for Automatic Text Summarization using LDA Topic Modelling. *IEEE International Conference on Big Data (Big Data)*, 2771–2780. https://doi.org/10.1109/BigData55660.2022.10020259

Onan, A., & Alhumyani, H.A. (2024a). DeepExtract: Semantic-driven extractive text summarization framework using LLMs and hierarchical positional encoding. *Journal of King Saud University - Computer and Information Sciences*, 36(8), 1–19. https://doi.org/10.1016/j.jksuci.2024.102178

Onan, A., & Alhumyani, H.A. (2024b). FuzzyTP-BERT: Enhancing extractive text summarization with fuzzy topic modeling and transformer networks. *Journal of King Saud University - Computer and Information Sciences*, 36(6), 102080.

Payak, A., Rai, S., Shrivastava, K., & Gulwani, R. (2020). Automatic Text Summarization and Keyword Extraction using Natural Language Processing. In: *Proceedings of the International Conference on Electronics and Sustainable Communication Systems*, p98–103. https://doi.org/10.1109/icesc48915.2020.9155852

Prasad, C., Kallimani, J.S., Harekal, D., & Sharma, N. (2020). Automatic Text Summarization Model using seq2seq Technique. In: *Proceedings of the 4th International Conference on IoT in Social, Mobile, Analytics and Cloud*, p599–604. https://doi.org/10.1109/I-SMAC49090.2020.9243572

Syed, A.A., Gaol, F.L., & Matsuo, T. (2021). A survey of the state-of-the-art models in neural abstractive text summarization. *IEEE Access*, 9, 13248–13265. https://doi.org/10.1109/access.2021.3052783

Tomer, M., & Kumar, M. (2022). Multi-document extractive text summarization based on firefly algorithm. *Journal of King Saud University - Computer and Information Sciences*, 34(8), 6057–6065. https://doi.org/10.1016/j.jksuci.2021.04.004

Wahab, M.H.H., Ali, N.H., Hamid, N.A.W.A., Subramaniam, S.K., Latip, R., & Othman, M. (2024). A review on optimization-based automatic text summarization approach. *IEEE Access*, 12, 4892–4909. https://doi.org/10.1109/access.2023.3348075

Widyassari, A.P., Rustad, S., Shidik, G.F., Noersasongko, E., Syukur, A., Affandy, A., et al. (2022). Review of automatic text summarization techniques and methods. *Journal of King Saud University - Computer and Information Sciences*, 34(4), 1029–1046. https://doi.org/10.1016/j.jksuci.2020.05.006

Yadav, D., Katna, R., Yadav, A.K., & Morato, J. (2022). Feature based automatic text summarization methods: A comprehensive state-of-the-art survey. *IEEE Access*, 10, 133981–134003. https://doi.org/10.1109/access.2022.3231016

Yang, J., Wang, H., Qin, H., Sun, Y., Khan, A.A., Por, L.Y., et al. (2025). A generative adversarial network-based extractive text summarization using transductive and reinforcement learning. *IEEE Access*, 13, 65490–65509. https://doi.org/10.1109/access.2025.3558266

Zhang, M., Zhou, G., Yu, W., & Liu, W. (2020). A Survey of Automatic Text Summarization Technology Based on Deep Learning. In: *Proceedings - International Conference on Artificial Intelligence and Computer Engineering*, p211–217. https://doi.org/10.1109/icaice51518.2020.00047

Zhong, J., & Wang, Z. (2022). MTL-DAS: Automatic text summarization for domain adaptation. *Computational Intelligence and Neuroscience*, 2022, 1–10. https://doi.org/10.1155/2022/4851828

AUTHOR BIOGRAPHIES

Sunil Upadhyay received his M.TECH. Degree from the Department of Computer Science and Engineering at RGEC Meerut, Gautam Buddh Technical University, formerly known as UPTU, Lucknow, U.P., India, in 2012. He is pursuing his Ph.D. degree in Computer Science and Engineering from Amity University, Madhya Pradesh. His current research interest includes NLP, Data analytics, Artificial Intelligence, and Machine Learning.

Hemant Kumar Soni completed his M Tech (IT) at Bundelkhand University, Jhansi, Uttar Pradesh, India, and a Doctoral degree in Computer Science and Engineering from Amity University, Madhya Pradesh, Gwalior, India. He has 26 years of teaching experience for UG and PG courses in Computer Science and is presently working as a Professor in the Department of Computer Science and Engineering at Amity University, Madhya Pradesh, India. His research interests are in Natural Language Processing, Data Science, Machine Learning, and Data Mining. He published many research papers in Web of Science, SCI, and Scopus Indexed Journals. His research articles received high levels of citations in Scopus and Google Scholar. He is a Reviewer of many referred journals including IEEE Transactions.