International Journal of Computing and Digital Systems
ISSN (2210-142X)
Int. J. Com. Dig. Sys. 16, No.1 (Jul-2024)

http://dx.doi.org/10.12785/ijcds/160126

Optimizing Deep Learning Architecture for Scalable Abstractive Summarization of Extensive Text Corpus

Krishna Dheeravath[1] **and S Jessica Saritha**[2]

[1]*Research Scholar, Department of Computer Science and Engineering, Jawaharlal Nehru Technological University, Ananthapur, Andhra Pradesh - 515002*
[2]*Assistant Professor, Department of Computer Science and Engineering, Jawaharlal Nehru Technological University, Ananthapur, Andhra Pradesh - 515002*

Received 01 Feb. 2024, Revised 14 Apr. 2024, Accepted 20 Apr. 2024, Published 1 Jul. 2024

Abstract: Text processing plays a prominent role in dealing with the ever-growing volume of information available on the internet as well as digital platforms. Abstractive text summarization and categorization, in particular, aims to generate concise and coherent summaries by paraphrasing the source text while preserving its core meaning and context. This research work focuses on enhancing abstractive text summarization and categorization for the large corpora through the application of a robust deep neural network architecture. With the increasing volume of information available, the need for efficient summarization techniques becomes critical. A pre-training strategy using diverse datasets is employed to improve the model's statistical performance and generalization capabilities. Furthermore, to address the challenge of information overload, an attention-based content selection mechanism is introduced, which highlights essential information from the source text to guide this process. The model's effectiveness is also extended to multi-document summarization, ensuring coherence across related documents. To evaluate the performance, various statistical performance metrices are exploited. In order to judge the novelty of adapted strategy, a benchmarking has been carried out with some state-of-the-art existing frameworks. The obtained results demonstrate the significant potential of this approach in effectively summarizing large corpora and managing the overwhelming amount of textual data available.

Keywords: Deep Learning, Corpus Processing, Computational Linguistics, Language Models, Text Classification

1. Introduction

Natural Language Processing (NLP) is an specific domain dedicated to developing methodologies that enable computing machines to process and interpret textual data composed in natural languages, the languages employed in human-to-human communication such as English. This is markedly different from programming languages that prescribe rigidly defined expressions. Natural languages lack inherent constraints, although they typically adhere to an internal arrangement. Nonetheless, their structure and semantics aren't directly machine-readable, necessitating specialized solutions for the processing and analysing of such data. Furthermore, natural language texts often exhibit high complexity, ambiguity, and expressive variations. The exploration of Natural Language Processing (NLP) is a multifaceted endeavour that entails the understanding and processing of language at its various layers morphological, syntactic, semantic, and pragmatic [1]. It involves analysis from both linguistic and computational viewpoints and incorporating fundamental models and algorithms pertinent to NLP. Properly understanding certain statements often requires broader world knowledge, posing significant challenges in processing natural languages [2]. Despite these complexities, the field of NLP has seen exponential growth, particularly since the advent of the new millennium. Several factors have contributed to this technological acceleration (Elsevier 2018) [3]. Early NLP solutions were predominantly based on manually constructed rule sets, which entailed the time-consuming task of formulating explicit rules that a computer system would adhere to. However, since the late 20th century, the focus has shifted towards statistical NLP approaches leveraging machine learning, primarily supervised learning. These statistical models necessitate annotated datasets featuring input-output pairs, bypassing the need for explicit rule sets [4]. This approach is generally more efficient and flexible, permitting the application of similar algorithms across various problems, given the availability of appropriate data. The exponential increase in textual content availability, brought about by the expansion of the Internet and the advent of Web 2.0, has significantly benefited NLP [5]. As most of the Internet's data is text-based for human communication, the field has benefited from the surge in data. Concurrent advancements in hardware capabilities and computing power have facili-

E-mail address: kittu514@gmail.com, sjsaritha.cse@jntua.ac.in

tated the efficient processing of larger datasets, propelling the adoption of statistical models that outperform rule-based solutions on most NLP tasks. This rapid evolution in the field of NLP has spurred increased interest in commercialising NLP solutions, further stimulating research endeavours in this area.

Considerable commercial applicability has made the problems of semantic text processing among the most popular in current research. Semantic problems in natural language processing include a large number of tasks that imply or aim at a correct understanding of the meaning of texts, such as analysis of sentiments and emotions in the text, determination of semantic similarity, detection of paraphrases, answering questions, retrieving information, creating text summaries, text simplification, machine translation, natural language inference, etc [6]. Within this, the problems of processing short texts are especially highlighted, where, due to the limited length of the available textual content, semantic processing is noticeably more difficult than with longer documents. Although there is no firmly established definition, short texts are usually understood to be one or two sentences long, i.e., texts ranging from a few words to a paragraph. Texts of this length are often found on the Internet in product descriptions, headlines and summaries of news and articles, visitor comments on websites, forums or social networks, etc [7].

A. Text summarization

Text Summarization is a critical sub-field of NLP that focuses on creating abbreviated, coherent, and accurate renditions of longer documents or pieces of text. The objective is to generate summaries that retain the primary ideas and essential information in the source content. It is an efficient way to condense large volumes of text data, making it quicker and easier for users to understand the text's main points without reading it in its entirety [8]. There are primarily two types of text summarization: "Extractive" and "Abstractive". Table I compares Extractive and Abstractive Text Summarization.

B. Evolution of large text Corpus

The evolution of large text corpora has been a key development in NLP and computational linguistics. One of the main aspects to consider when discussing the evolution of large text corpora is their size. Let's denote the size of a corpus (Number of words) as N. Traditionally, N has increased over time due to storage and processing technology advancements. For instance, in the 1960s, N was typically in the order of 10^4 (tens of thousands of words). Nowadays, it is not uncommon for N to be in the order of 10^9 (billions of words) or even larger. The size of a text corpus plays a crucial role in many NLP tasks [9]. For instance, larger corpora generally lead to more accurate models when training language models. This can be formalized in equation 1:

TABLE I. Comparison between Extractive and Abstractive text summarization

	Extractive Summarization	**Abstractive Summarization**
Technique	Selecting key sentences/phrases from source text	Understanding and rewriting source text
Advantages	Simpler, less error-prone, retains original phrasing	More natural, versatile, and concise; better mimics human summarization
Disadvantages	May lack coherence, does not reduce length significantly	More complex, risk of generating incorrect or nonsensical sentences
Application Examples	News aggregation, search engine snippets	News headline generation, chatbots

$$P(w_i|w_{i-n+1},\ldots,w_{i-1}) \approx \frac{Count(w_{i-n+1},\ldots,w_i)}{Count(w_{i-n+1},\ldots,w_{i-1})} \quad (1)$$

where $P(w_i|w_{i-n+1},\ldots,w_{i-1})$ is the probability of word w_i occurring given the previous $n-1$ words, and $Count(w_{i-n+1},\ldots,w_i)$ and $Count(w_{i-n+1},\ldots,w_{i-1})$ are the occurrences of the respective sequences of words in the corpus. As N increases, these counts become more accurate, leading to more accurate probabilities and, thus, a more accurate language model.

However, the evolution of large text corpora is not only about size. Diversity and quality of the data are also important. For instance, a corpus that includes text from a wide range of domains will likely lead to a more robust and versatile language model. Similarly, a corpus with clean, well-formatted text will generally be more beneficial than a corpus with lots of noise and errors. Advancements in technology have driven the evolution of large text corpora and have significantly impacted the field of NLP. These corpora's size, diversity, and quality are crucial factors that influence their usefulness for various NLP tasks.

C. Deep Neural Nets

Deep Neural Networks (DNNs) are a category of Artificial Neural Networks (ANNs) with more than one hidden layer. These multiple layers provide the model its "depth", leading to the term "deep learning". Using multiple hidden layers, DNNs can model intricate, non-linear relationships in the data. This feature has led to their significant success in NLP tasks, where data is inherently sequential, and the

relationships between elements can be highly complex. A typical Deep Neural Network (DNN) is composed of a specific input layer, several hidden layers, and a final output layer. Within each hidden layer, a group of neurons are present, with each neuron forming connections to every neuron in the preceding and succeeding layers. The training process involves learning the weights associated with these connections. In the context of NLP, the input to the network is usually a sequence of words or characters, transformed into numerical representations or embeddings, capturing the semantic and syntactic properties of the words or characters [10].

One of the most prevalent types of DNN used in NLP is the Recurrent Neural Network (RNN). RNNs are structured to handle sequential data, making them an excellent fit for NLP tasks. They achieve this by maintaining a hidden state vector as a "memory" of past inputs [11]. The hidden state vector evolves at each time step, incorporating the present input and the prior hidden state. This enables the network to capture temporal dependencies within the data. Equations 1 and 2 represent the computational update:

$$h_t = \sigma(W_{hh}h_{t-1} + W_{xh}x_t + b_h) \quad (2)$$

$$y_t = \sigma(W_{hy}h_t + b_y) \quad (3)$$

where In this context, h_t signifies the hidden state at the moment t, x_t denotes the input at the same time point, and y_t represents the output at time t. The weight matrices are W_{hh}, W_{xh}, and W_{hy}, while b_h and b_y are the bias vectors. The activation function, often the hyperbolic tangent function, is indicated by σ. While Recurrent Neural Networks (RNNs) have the theoretical ability to capture dependencies over long time spans, they face challenges with these tasks due to the vanishing and exploding gradient computational problems. More advanced RNN architectures have been proposed to mitigate these issues, such as LSTM [12] and GRU [13]. These structures incorporate gating mechanisms, regulating information flow in the network and facilitating the learning of long-range dependencies. The Transformer model, a recent advancement, employs self-attention mechanisms to capture dependencies among all elements in the input sequence, irrespective of their distance. This model has emerged as the standard for numerous Natural Language Processing (NLP) tasks. The self-attention mechanism assesses the significance of each input element in output computation, enabling the model to focus on the most pertinent aspects of the input. Despite the achievements of deep learning in NLP, numerous challenges persist, such as:

- Training deep learning models necessitates substantial volumes of annotated data, posing challenges in terms of both difficulty and expense.
- These models often lack interpretability, making it difficult to comprehend why they make certain predictions.
- Additionally, they are sensitive to changes in the input distribution and can make unpredictable predictions when faced with inputs differing from their training data.

D. Contribution Highlights

The core contributions aligned into this work are as follows:

- A robust DNN architecture is developed to enhance abstractive text summarization and categorization. This architecture improves upon previous models by incorporating advanced computational primitives.
- The effectiveness of this approach is demonstrated on large corpora, indicating the model's scalability and ability to handle high volumes of data without a significant loss in performance quality and categorization accuracy.
- This work also devises an efficient training process to optimize the model's parameters, ensuring that the deep neural network quickly converges to a solution that provides high-quality text categorization.
- Comprehensive analysis and benchmarking of the results is also performed with other existing techniques, highlighting the robustness of the proposed strategy.

E. Structure of the Paper

The organization of the paper is as follows: In Section 2, a comprehensive literature review scrutinizes previous studies and models pertinent to the field. Section 3 presents the Adapted Framework, describing our robust and scalable approach for handling the challenges identified in the literature review. This section details the design of the framework architecture, including the underlying system architecture and the algorithms employed. Section 4, Computational Analysis, offers a theoretical analysis of our methodology, assessing its computational complexity, efficiency, and scalability. In Section 5, we present an Experimental Evaluation of our approach. We detail the experimental setup, including the data used, the evaluation metrics, and the results of the experiments. This section provides a quantitative analysis of our framework's performance compared to existing methods, demonstrating its accuracy, efficiency, and robustness advantages. Finally, in Section 6, we delve into potential avenues for future research.

2. Literature Review

In machine learning, binary classification focuses on two-class problems, for example, distinguishing between spam and non-spam emails. On the other hand, multi-class classification deals with scenarios where more than two labels need to be assigned, for instance, categorizing text into positive, neutral, or negative sentiments or detecting various emotions. While researchers have invested considerable efforts in two areas, (1) enhancing learning models tailored to specific text classification problems [14], and (2)

designing (neural) architectures specifically suited for text classification [15], there has been comparatively less work on developing universal models capable of handling multi class problems irrespective of the task and the objects to be classified.

A. Generated sentence detection

Currently, many generated sentence detection methods use deep learning. However, detection by deep learning has two problems. The first problem is that learning is necessary for highly accurate detection. This may cause a problem of delay in response when a new and advanced language model is devised. By using only the distributed representation of the results, we are able to handle sentences generated from an unknown language model without learning and, at the same time, achieve extremely high execution speed [16].
The second problem is that these previous studies are based on the condition of inputting a single sentence. Current text generation methodology can output sentences with higher accuracy by identifying the direction of sentences to some extent by learning conversational responses by humans and adjusting hyperparameters. Therefore, when this generation of technology is used, it is highly likely that a human being writes the beginning of the sentence or that the condition is similar. However, since the conventional method uses all sentences for detection, there is a possibility that the sentence written by a human at the beginning may be falsely detected [17]. Therefore, in this research, we aim to detect sentences that combine sentences written by humans and sentences generated by machines, making it a more practical method. In addition, in this research, unlike conventional generated sentence detection, the research subject is not only news but also novels and Wikipedia sentences, etc., making it a highly versatile detection method.

B. State-of-the-art

For increasingly massive data labelling needs, unsupervised learning is once again becoming one of the main challenges of data science. As a result, clustering and visualization, for example, can be very useful for creating value from unlabeled data, particularly textual data. Currently, a wide range of textual data representations is offered to practitioners, including sparse word bags or Bag-Of-Words (BOW) [18] and dense word bags such as Word2vec [19]. and GloVe [20], also called static word embeddings. More recently, text/document representations provided by Transformer-based Pre-trained Language Models (MLPT) like BERT [21] and RoBERTa [22] which produce different representations to represent a document. Despite the multiplication of embedding methods, there is no clear answer regarding the expected performance in an unsupervised context where no label is available. In particular, Transformer-based embeddings are attracting more and more interest, performing very well in many automatic language processing centric tasks such as question answering and Semantic Textual Similarity (STS), but are much less present in the unsupervised domain.
Moreover, [23] demonstrated that the performance obtained by BERT on the unsupervised version is inferior to that obtained by Glove. However, the study solely focused on the last layer of BERT and did not include post-processing. It has been shown that this approach is far from being the most effective strategy to leverage MLPT's capabilities fully. In our experiments, we extend the analysis to all the representations provided by a multi-layer Transformer model, not just the one provided by the last layer. To improve the quality of MLPTs, several retraining strategies are proposed in the literature, such as the one proposed by [24], which retrain a Siamese MLPT on NLI and STS tasks, thus improving the performance obtained by the last layer of BERT and RoBERTa on the N-STS task. This approach is supposed to be well suited for unsupervised tasks, including clustering, but has not been evaluated on the latter. DvBERT [25] is also retrained on a supervised task based on word interactions. On the other hand, several unsupervised approaches are proposed ([26]; [27]; [28]), all based on self-supervised objectives and requiring no labeled data. All the aforementioned approaches have been exclusively evaluated on the N-STS task, and whether they are well suited for clustering is unknown. Another way to improve the results obtained by these representations is to rely on post-processing techniques applied to the output vectors. These approaches mainly exploit dimensionality reduction (DR) based on PCA, which has proven to be efficient enough to capture semantic information while reducing dimensions. In the case of static embeddings, a PCA-based approach proposed by [29] is used to halve the dimensions without altering the performance.
Work by [30] have utilized two transformer-based language models, specifically the Bidirectional and Auto-regressive Transformer (BART) and the Text-To-Text Transfer Transformer (T5), on the CNN_dailymail dataset. Compared to other models mentioned in existing literature for the same task, this model delivers superior performance. Subsequently, authors in [31] suggested improvements to current architectures and models for abstractive text summarization, focusing on fine-tuning hyper-parameters and testing specific encoder-decoder combinations. They conducted numerous experiments on the widely-used CNN_DailyMail dataset to evaluate the effectiveness of various models. However, one limitation of their adapted approach is its significant computational demand, which may not be feasible for environments with limited resources. Additionally, the approach may not generalize well to datasets that differ significantly in style or content from the CNN/DailyMail dataset. Lastly, the models could potentially overfit to specific textual patterns within the training data, leading to less robust performance on unseen texts.

C. Limitations of existing approaches

Acknowledging some inherent limitations that may affect its practical application is essential. One notable limitation is the computational complexity of deep neural networks, especially when dealing with extensive corpora, which could lead to increased resource requirements and longer processing times. Additionally, overfitting is a con-

cern in deep learning, and the proposed architecture may suffer from this issue, affecting the model's generalization on new and unseen data. Furthermore, abstractive summarization requires a deep understanding of context and language ambiguity, which may challenge the proposed model in generating coherent and contextually accurate summaries. Moreover, evaluating the performance of abstractive text summarization models can be subjective, as existing metrics may not fully capture the quality and nuances of generated summaries. Despite these limitations, the paper's contribution provides valuable insights into advancing abstractive text summarization techniques and addressing these challenges can lead to further improvements in the field [1]. A prominent drawback of recently published approaches is their considerable computational requirements, which might be impractical for settings with restricted resources. These approaches demand substantial processing power and memory, which can limit their applicability in environments lacking advanced computational infrastructure. Furthermore, these models often struggle to adapt to datasets that vary markedly in style or content from the CNN/DailyMail dataset, indicating a lack of generalizability. This limitation hinders their effectiveness across diverse textual domains, potentially affecting their utility in real-world applications where data variability is common. Additionally, there is a risk that these models may overfit to specific patterns and idiosyncrasies within their training data. This overfitting can lead to a degradation in performance when the models encounter new, previously unseen texts, thereby reducing their overall robustness and reliability in practical scenarios.

3. Adapted Framework

This section outlines the overview of our modified framework, where we deviate from utilizing BERT purely as a feature extraction model and instead adopt the fine-tuning approach. In this adaptation, we extend the BERT model by incorporating an additional end-to-end deep network layer, referred to as RNN. The process entails BERT generating contextualized embedding vectors for each word, which are subsequently fed through the RNN deep network layers. A visual representation of this procedure is illustrated in Figure 1. The feature vector is constructed by concatenating the output neurons for each word from the intermediate layer. Following this, each vector undergoes dimension reduction through a densely connected neural network. The final reduced vector is subjected to classification using PReLu. Additionally, we integrate three additional learning algorithms—Word2Vec, Glove, and fastText with pre-trained word embeddings to augment the overall performance of the model.

A. *Forward inference with RNNs*

Forward inference in an NN maps a linear or non-linear input sequence to corresponding output sequence. In RNN, forward inference is almost identical to FFNs. To compute the output y_t based on the input x_t, the output values of the hidden layer h_t are essential. This process

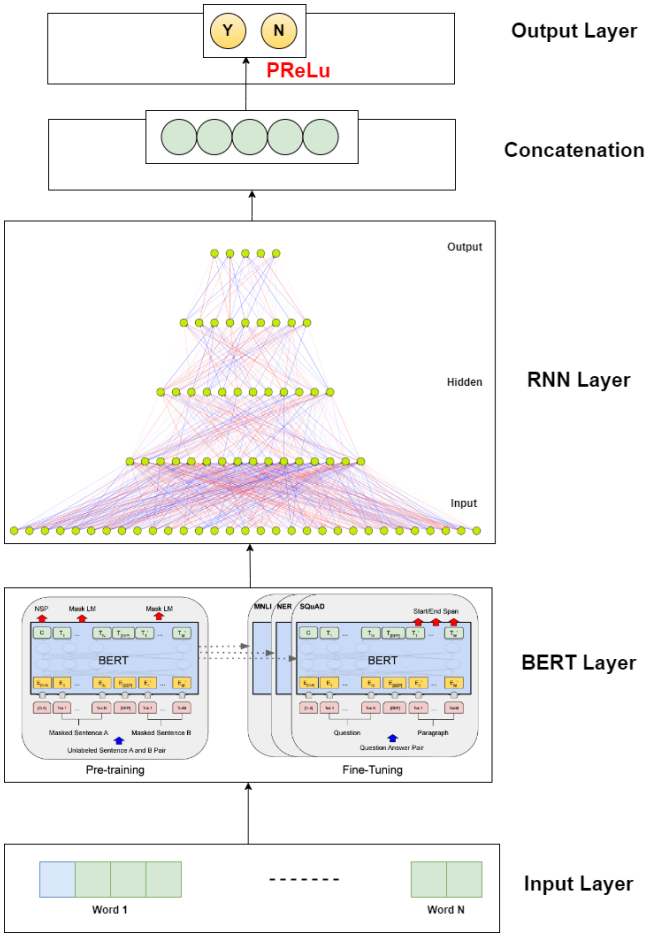

Figure 1. Representative mechanism of BERT to RNN in word categorisation

entails multiplying the input x_t by the weight matrix i.e., W and the output of the previous time step's hidden layer h_{t-1} by the weight matrix i.e., U. These resulting values are then added together, and this sum is processed through the relevant activation function g to compute the current hidden layer's output value, h_t. After determining the values for the hidden layer, the typical calculation for the output vector is carried out.

$$h_t = g(Uh_{t-1} + Wx_t) \tag{4}$$

$$y_t = f(Vh_t) \tag{5}$$

Let's denote the dimensions of the particular input, hidden (intermediate), as well as output layers as d_{in}, d_h, along with d_{out}, respectively. Using these notations, our three weight matrices are $W \in \mathbb{R}^{d_h \times d_{in}}$, $U \in \mathbb{R}^{d_h \times d_h}$, and $V \in \mathbb{R}^{d_{out} \times d_h}$.

In the frequently encountered scenario of mild classification, the computation of y involves applying a softmax function, providing a probability distribution across the potential classes.

$$y_t = softmax(Vh_t) \tag{6}$$

Definition1.2.1(softmax).Mappingsoftmax: $\mathbb{R}^n \to \mathbb{R}^n, n \in \mathbb{N}, zax \in \mathbb{R}^n$ defined by:

$$softmax(x) = \frac{\exp(x)}{\sum_{i=1}^{n} \exp(x_i)} \tag{7}$$

we call softmax. Here, exp represents an exponential function by component darkness, and the division is also by components.

B. BERT

Bidirectional Encoder Representations from Transformers (BERT) [21] is a computationally innovative model derived from the encoder component of the Transformer model. BERT is most specifically crafted for pre-training deep bidirectional representations from the unlabeled text, utilizing both the left as well as right contexts across all layers. The unique feature of BERT allows a pre-trained model to be further fine-tuned for diverse variety of tasks by simply adding an additional output layer, without requiring substantial modifications to the task-specific architecture. This process of refining an already pre-trained model is referred to as fine-tuning. In the context of BERT, the focus is on downstream tasks—tasks that aim to be addressed by adapting a pre-trained model.

1) Description of BERT

The BERT framework primarily comprises two essential phases i.e, pre-training along with fine-tuning. During pre-training, the model is exhaustively trained on a set of unlabeled data, tackling different tasks. In the specific fine-tuning stage, the model starts with parameters from the pre-training phase and then fine-tunes all parameters with labeled data tailored to a specific downstream task. While each task results in a uniquely adjusted model, they all originate from the same foundational pre-trained parameters. BERT's unified architecture remains consistent across diverse tasks, with minimal variations between the pre-trained and downstream architectures. N denotes the count of layers, d_model the length of the output of sub-layers, d_ff the internal dimension of the FFN layer, and h the number of the attention head, we list two models in BERT: $BERT_{BASE}$ (N = 12, d_{model} = 768, d_{ff} = 3072, h = 12, total parameters = 110M) and $BERT_{LARGE}$ (N = 24, d_{model} = 1024, d_{ff} = 4096, h = 16, total parameters = 340M).

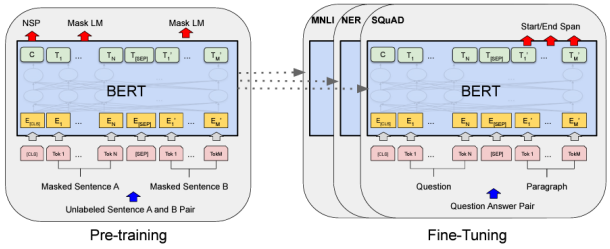

Figure 2. BERT pre-training and refinement procedure

Figure 2 shows the overall BERT pre-training and refinement procedure. With the exception of the output layers, identical architecture is employed in both pre-training as well as fine-tuning. The identical set of pre-trained parameters initializes the model for various downstream tasks. Throughout the tuning process, all parameters undergo adjustment. The inclusion of special tokens is noteworthy—CLS → serves as a unique token added to the start of each input, while SEP → is a distinctive separation token (eg it separates the question from the context to be answered).

Also BERT does not use ReLU activation function [32] but GELU (Gaussian Error Linear Unit) [33]. GELU proved to be a better alternative for the ReLU function and is used in most Transformer models. It is defined using the distribution function of the normal (Gaussian) distribution, i.e. using the random variable $X \sim N(0,1)$ and one can mathematically approximate the GELU with $0.5x\left(1+\tanh\left[\sqrt{2/\pi}\left(x+0.044715x^3\right)\right]\right) or x\sigma(1.702x)$:

$$GELU(x) := xP(X \le x) = x\Phi(x) = x \cdot \frac{1}{2}[1 + erf(x/\sqrt{2})] \tag{8}$$

In order for BERT to handle various downstream tasks, it exploits an input representation that can represent both a single sentence and a pair of certain sentences in a single sequence of statistical tokens. Therefore, a "sentence" can also be a text of several sentences in the context of BERT. A "sequence" represents a sequence of BERT input tokens, and it can be either a single sentence or the consolidation of two sentences consolidated together.

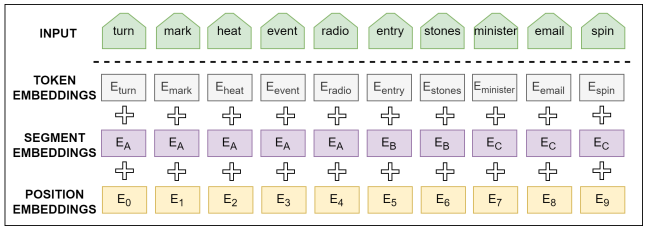

Figure 3. Representation of the BERT input → (Input investments are the sum of token, segment and position investments)

BERT employs WordChunk embeddings with a vocabulary comprising 30,000 tokens. The initial token in every sequence is consistently a special categorization token (CLS). The last hidden state associated with this token serves as the aggregated sequence representation for the categorization task. Sequences consist of pairs of sentences grouped together. The sentences are differentiated in two ways: first, by the inclusion of a special token (SEP) between them, and second, through appending learned embeddings to each token, which identify whether it is part of sentence entity A or B. As depicted in Figure 3, the input embeddings are represented by E. The final hidden non-linear vector of the special [CLS] token is indicated by $C \in \mathbb{R}^{d_{model}}$, and the

final hidden vector for the i^{th} input token is represented as $T_i \in \mathbb{R}^{d_{model}}$. The input depiction for each token is formed by summing the respective token, segment, as well as position embeddings.

2) BERT Pre-training and Post-training

Here, BERT is used similarly to a language model, ie. as a model that predicts input sequences which is the next token in the sequence. But BERT will not predict the next word, but we mask some particular percentage of the input tokens randomly, and then BERT predicts those masked tokens. The authors of BERT call this procedure MLM (masked language model). In this scenario, the ultimate hidden vectors linked to the masked tokens are directed to the output softmax. Although this approach enables the development of a bidirectional computationally pre-trained model, it introduces a notable disparity between pre-training along with fine-tuning stages due to the absence of the [MASK] token in fine-tuning. To mitigate this, BERT intermittently replaces "masked" words with the [MASK] token, rather than consistently. The exhaustive training data generator randomly selects 15% of the token positions for the prediction purposes. If the i^{th} token is empirically chosen, it is then replaced with:

- *MASK* token specifically 80% of the time.
- *Random* token in 10% of cases.
- Neglect the original token in 10% of cases. Then, T_i is exploited to predict the empirically original token with the cross-entropy centric loss.

Post-training:
Tuning the BERT model is relatively straightforward, owing to the inherent design of Transformer's self-aware mechanism. The flexibility of this architecture allows BERT to be applied to a diverse array of downstream tasks, regardless of whether these tasks require processing of a single text or pairs of texts. This versatility is achieved by the model's ability to adapt to different task-specific inputs and outputs. When preparing to utilize BERT for a particular downstream task, we simply substitute the task-specific inputs and outputs into the model. To illustrate, if the task involves sentiment analysis, the input would be the text to be analyzed, and the output would be the sentiment label. Similarly, for a text summarization task, the input would be the original text, while the output would be the summary text. One of the keys to BERT's broad applicability is its process of fine-tuning, wherein all the parameters of the model are adjusted for the specific task at hand. Fine-tuning enables the model to perform optimally on the task by adjusting to the idiosyncrasies of the specific data and task requirements. Finally, the output layer of BERT is utilized differently based on the nature of the task. For token-level tasks, such as question answering or named entity recognition, token representations are passed to the output layer. Each token representation captures the semantics of a specific word in the context of its surrounding words. Alternatively, for classification tasks, the representation of the special [CLS] token is passed to the output layer. The [CLS] token, which is added at the beginning of the input, encapsulates a representation of the entire sequence, and is thus suitable for tasks that require a holistic understanding of the input text. The choice of proposed architecture and technique type i.e., RNN (e.g., LSTM, GRU) in this work is fundamental due to their differences in handling long-term dependencies and memory management. LSTMs are designed to avoid the long-term dependency problem of vanilla RNNs by incorporating memory cells that allow them to store and access information over long sequences. This makes LSTMs particularly suited for tasks that require understanding complex context over large text spans. GRUs, on the other hand, provide a simpler but often equally effective alternative to LSTMs, with fewer parameters and a more streamlined architecture. Specifically, LSTM is preferred over GRUs in this work.
As a pre-training method, GloVe is leveraged. GloVe is designed to capture both global statistics and local context of words in a corpus, potentially complementing BERT's deep bidirectional context understanding. Combining these embeddings could enhance the model's linguistic understanding and adaptability to different contexts or domains. Self-attention mechanism within BERT architecture is used here as a key feature that allows it to capture the context of words from both directions in a sentence. We introduced attention heads with varying focuses, that allows the model to better capture different types of relationships in the text, improving performance on tasks like entity recognition or sentiment analysis.

4. Computational Complexity Analysis

BERT is exploited as a computational module for Transformer-based model with a structure that's divided into multiple layers, attention heads, and tokens. It has a significant computational complexity, and it's specifically denoted as $O(\beta^2)$ due to the self-attention mechanism where β is the sequence length. Suppose, model size as $\mathcal{S}$, the number of self-attention heads in each transformer are $\mathcal{H}$ and input sequence length as $\mathcal{L}$ then the computational complexity is estimated as $O(S.H.L^2)$. Additionally, in case of RNNs, consider Hidden Layer size is α and sequence length is γ then the computational complexity of this module would be = $O(\alpha.\gamma + \epsilon)$.

Addressing the three significant limitations (Computational complexity, Over-fitting, Contextual understanding and ambiguity) in the context of large corpora and abstractive summarization tasks offers a clear perspective on the model's practical applicability and the challenges it may face in real-world scenarios.

- In the context of large corpora, computational complexity is a crucial concern because it directly impacts the feasibility and scalability of the proposed model. Abstractive summarization tasks, which involve gen-

erating concise summaries that capture the essence of large texts, require processing and understanding extensive sequences of data. Given that the complexity of some NLP models, including those based on transformers, scales quadratically with the sequence length, deploying such models for summarization tasks involving large documents or datasets can be computationally prohibitive. This limitation is especially pertinent when considering real-time applications or environments with limited computational resources.

- Overfitting is a critical issue in computationally intelligent models trained on extensive datasets, as it can lead to models that perform well on training data but poorly generalize to unseen data. This is particularly relevant for abstractive summarization tasks where the model needs to generate summaries for a wide range of texts not encountered during training. The risk of over-fitting increases with the complexity of the model and the specificity of the training data.

- For abstractive summarization tasks, the ability of a model to understand context and resolve ambiguity is paramount. These tasks often require the model to interpret complex narratives, discern relevant information, and generate summaries that are coherent, concise, and reflective of the original text's intent. The limitations in a model's ability to handle contextual understanding and ambiguity can lead to summaries that are inaccurate, misleading, or lack the nuance of the source material.

5. Experimental Evaluation and Benchmarking

This section presents a detailed description of the dataset, the experimental setup, and the computational libraries utilized. It also discusses the statistical performance evaluation metrics employed, and provides an overview of the simulation pipeline together with the results obtained.

A. Dataset description

We have used BBC dataset [34] for simulation and experimental evaluation in this work. The BBC dataset is a collection of text documents compiled by the BBC for text categorization tasks, often used for machine learning research. The dataset is public and a popular resource for researchers and practitioners in Natural Language Processing (NLP) and related fields. This dataset consists of 2,225 articles in five topic areas/labels i.e., (Business, Entertainment, Politics, Sport, and Tech), with approximately equal representation for each category. This dataset is used for text categorization tasks, including topic classification and sentiment analysis.
It is taken from the Kaggle platform and originated from BBC News [34], containing 2225 comments classified into five classes shown in Table II.

TABLE II. Technical characteristics of the BBC-text dataset

ID	Active Classes	Count of Examples
0	Business	510
1	Entertainment	386
2	Politics	417
3	Sport	511
3	Tech	401
	Total	**2225**

B. Experimental setup, tools and computational libraries exploited

We have performed experiments on large text corpora, specifically news articles. The primary purpose of these experiments was to assess the performance of our proposed Robust Deep Neural Network Architecture in generating coherent and informative summaries. To implement our model, we primarily used Python programming language because of its vast support for scientific computing libraries. We leveraged the power of several libraries such as:

- **TensorFlow and Keras:** For designing, training, and evaluating our deep learning models. TensorFlow offered a comprehensive and flexible platform for machine learning, and Keras provided a high-level neural networks API that was user-friendly and easy to prototype.

- **NumPy and Pandas:** These were used for handling numerical computations and data manipulation tasks, respectively.

- **NLTK and Spacy:** These were exploited for preliminary processing tasks like tokenization, identifying parts of speech, and recognizing named entities.

- **Gensim:** This was used for training word embeddings and handling other unsupervised text analytics tasks.

Our experimental setup involved pre-processing the data, splitting it into training, validation, and testing sets, and then feeding the training and validation sets into our model for training. The model was then evaluated on the unseen testing set to gauge its performance.

C. Statistical performance evaluation metrics used

Table III depicts statistical performance of simulation. The performance of our framework, powered by a Robust Deep Neural Network Architecture, is statistically evaluated using several statistical metrics i.e.,

- **Accuracy:** This is one of the simplest and most straightforward evaluation metrics. It computes the fraction of predictions our model got empirically feasible.

- **ROUGE ("Recall-Oriented Understudy for Gisting Evaluation"):** ROUGE comprises metrics utilized for the assessing automatic summarization as

TABLE III. Statistical performance

Parameter	Training Loss	Model Accuracy
Epoch 1	0.715	0.411
Epoch 2	0.495	0.677
Epoch 3	0.246	0.896
Epoch 4	0.102	0.966
Epoch 5	0.056	0.988
Epoch 6	0.049	0.989
Epoch 7	0.280	0.991

well as machine translation. Its methodology involves comparing an automatically generated summary or translation with a set of reference summaries. The primary variants of ROUGE we used are ROUGE-N (precision, recall, and F-score of n-grams), ROUGE-L (longest possible common subsequence), and ROUGE-S (skip-bigram).

- **Precision:** Precision is another fundamental metric. It quantifies the number of correctly predicted words or phrases present in the summarization output. A higher precision indicates fewer false-positive results.
- **F1 Score:** The F1 score, ranging from 0 to 1 with 1 being the highest achievable score, represents the harmonic mean between precision and recall, offering a comprehensive assessment of these two metrics.

These statistical performance evaluation metrics, collectively, allowed us to objectively gauge the efficacy of our model.

D. Simulation pipeline and Obtained Results

Initially, the selected dataset is ingested as the input to the proposed deep neural network architecture. This is followed by an extensive data preprocessing stage, where normalization, tokenization, and possibly other transformations are conducted to make the data compatible with the model's requirements. Upon the completion of preprocessing, the prepared data is utilized for the training phase. This phase involves the use of sophisticated optimization algorithms to adjust the model's parameters, aiming to minimize the discrepancy between the model's predictions and actual outcomes, thus constructing the optimal model. After the training phase, the model undergoes a comprehensive evaluation process. The test dataset, independent from the training set, is employed in this step to assess the model's generalizability and predictive performance. This ensures that the model's performance is not merely memorized from the training data, but that it can also efficiently handle new, unseen data.

A Test instance:
predict(device, model, "Deputy Prime Minister John Prescott said a good deal had been secured.")
The predicted class is: politics

Figure 4 presents the performance graph for text categorization proposed framework. Ultimately, various statistical performance parameters, such as - accuracy, recall, precision, and F1-score, among others, are computed. These metrics generally provide quantitative measures of the effectiveness and robustness of the finalized model, guiding the judgement of its performance and illuminating potential areas for improvement. This methodical process underpins the proposed architecture's capacity to handle complex text summarization and categorization tasks. Dataset is divided as (85 : 15)% corresponding to training and { validation; testing }. Total number of Epochs were chosen as 7 and learning rate = 0.0005. Upon evaluation of the deep neural network model using the independent test dataset, we achieved a remarkable accuracy of 99.2%. This result implies that the model correctly predicted the category or summarization of the text for 99.2% of the instances in the test data, underpinning the effectiveness and robustness of the architecture in handling complex text summarization and categorization tasks.

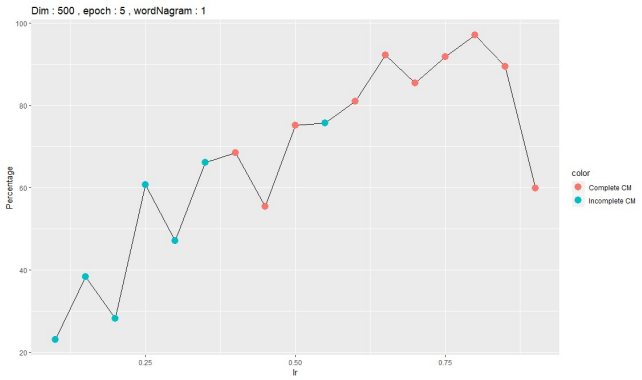

Figure 4. Depiction for text categorization performance

6. Conclusion and Future Work

In concluding this work, we introduced a robust deep neural network (DNN) architecture specifically engineered to advance the processes of abstractive text summarization and categorization. This architecture incorporates advanced computational primitives that mark a significant improvement over earlier models. It has shown notable scalability when applied to large data sets, maintaining high performance quality and categorization accuracy without considerable degradation. Moreover, we developed an efficient training regimen that swiftly steers the network toward finding the optimal solution, thereby facilitating high-quality text categorization. The robustness of our proposed approach was thoroughly validated through extensive benchmarking against existing techniques. In these comparisons, our model consistently demonstrated superior outcomes, highlighting its effectiveness and potential as a leading solution in the field.
Looking ahead, We aim to enhance the model's efficiency by integrating advanced methodologies such as transfer learning and multi-task learning. These techniques are expected to further elevate the model's performance across

diverse datasets. Additionally, the model's ability to handle noisy or imperfect data has not yet been fully explored, presenting an opportunity to increase its practical utility in real-world applications. It is also critical to evaluate the model across different languages and domains to determine its generalizability and adaptability to various contexts. The findings from this research open up exciting possibilities for future investigations and developments in abstractive text summarization and categorization. The implications are significant, offering a roadmap for subsequent research that could redefine the standards and capabilities of current text processing technologies.

References

[1] K. Chowdhary and K. Chowdhary, "Natural language processing," *Fundamentals of artificial intelligence*, pp. 603–649, 2020.

[2] D. Khurana, A. Koli, K. Khatter, and S. Singh, "Natural language processing: State of the art, current trends and challenges," *Multimedia tools and applications*, vol. 82, no. 3, pp. 3713–3744, 2023.

[3] A. Sarker, R. Ginn, A. Nikfarjam, K. O'Connor, K. Smith, S. Jayaraman, T. Upadhaya, and G. Gonzalez, "Utilizing social media data for pharmacovigilance: a review," *Journal of biomedical informatics*, vol. 54, pp. 202–212, 2015.

[4] P. Yin, B. Deng, E. Chen, B. Vasilescu, and G. Neubig, "Learning to mine aligned code and natural language pairs from stack overflow," in *Proceedings of the 15th international conference on mining software repositories*, 2018, pp. 476–486.

[5] Z. Abedjan, N. Boujemaa, S. Campbell, P. Casla, S. Chatterjea, S. Consoli, C. Costa-Soria, P. Czech, M. Despenic, C. Garattini *et al.*, *Data science in healthcare: Benefits, challenges and opportunities*. Springer, 2019.

[6] M. Han, X. Zhang, X. Yuan, J. Jiang, W. Yun, and C. Gao, "A survey on the techniques, applications, and performance of short text semantic similarity," *Concurrency and Computation: Practice and Experience*, vol. 33, no. 5, p. e5971, 2021.

[7] M. Sussna, "Word sense disambiguation for free-text indexing using a massive semantic network," in *Proceedings of the second international conference on Information and knowledge management*, 1993, pp. 67–74.

[8] A. Celikyilmaz, E. Clark, and J. Gao, "Evaluation of text generation: A survey," *arXiv preprint arXiv:2006.14799*, 2020.

[9] R. Dale, H. Moisl, and H. Somers, *Handbook of natural language processing*. CRC press, 2000.

[10] V. Sze, Y.-H. Chen, T.-J. Yang, and J. S. Emer, "Efficient processing of deep neural networks: A tutorial and survey," *Proceedings of the IEEE*, vol. 105, no. 12, pp. 2295–2329, 2017.

[11] V. Mittal, D. Gangodkar, and B. Pant, "Exploring the dimension of dnn techniques for text categorization using nlp," in *2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS)*. IEEE, 2020, pp. 497–501.

[12] A. Graves and A. Graves, "Long short-term memory," *Supervised sequence labelling with recurrent neural networks*, pp. 37–45, 2012.

[13] R. Dey and F. M. Salem, "Gate-variants of gated recurrent unit (gru) neural networks," in *2017 IEEE 60th international midwest symposium on circuits and systems (MWSCAS)*. IEEE, 2017, pp. 1597–1600.

[14] S. Dumais, J. Platt, D. Heckerman, and M. Sahami, "Inductive learning algorithms and representations for text categorization," in *Proceedings of the seventh international conference on Information and knowledge management*, 1998, pp. 148–155.

[15] P. Liu, X. Qiu, and X. Huang, "Recurrent neural network for text classification with multi-task learning," *arXiv preprint arXiv:1605.05101*, 2016.

[16] Z. Ren, Q. Shen, X. Diao, and H. Xu, "A sentiment-aware deep learning approach for personality detection from text," *Information Processing & Management*, vol. 58, no. 3, p. 102532, 2021.

[17] M. M. Lopez and J. Kalita, "Deep learning applied to nlp," *arXiv preprint arXiv:1703.03091*, 2017.

[18] C.-F. Tsai, "Bag-of-words representation in image annotation: A review," *International Scholarly Research Notices*, vol. 2012, 2012.

[19] K. W. Church, "Word2vec," *Natural Language Engineering*, vol. 23, no. 1, pp. 155–162, 2017.

[20] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1532–1543.

[21] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[22] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized bert pretraining approach," *arXiv preprint arXiv:1907.11692*, 2019.

[23] L. Gan, Z. Teng, Y. Zhang, L. Zhu, F. Wu, and Y. Yang, "Semglove: Semantic co-occurrences for glove from bert," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 30, pp. 2696–2704, 2022.

[24] Y. Zhang, R. He, Z. Liu, L. Bing, and H. Li, "Bootstrapped unsupervised sentence representation learning," in *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, 2021, pp. 5168–5180.

[25] X. Cheng, "Dual-view distilled bert for sentence embedding," in *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2021, pp. 2151–2155.

[26] A. Saeed, T. Ozcelebi, and J. Lukkien, "Multi-task self-supervised learning for human activity detection," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 3, no. 2, pp. 1–30, 2019.

[27] A. Saeed, F. D. Salim, T. Ozcelebi, and J. Lukkien, "Federated self-supervised learning of multisensor representations for embedded intelligence," *IEEE Internet of Things Journal*, vol. 8, no. 2, pp. 1030–1040, 2020.

[28] D. Hendrycks, M. Mazeika, S. Kadavath, and D. Song, "Using self-supervised learning can improve model robustness and uncertainty," *Advances in neural information processing systems*, vol. 32, 2019.

[29] L. Shen, H. Jiang, L. Liu, and Y. Chen, "Frequency-aware dimension selection for static word embedding by mixed product distance," *arXiv preprint arXiv:2305.07826*, 2023.

[30] R. Rao, S. Sharma, and N. Malik, "Automatic text summarization using transformer-based language models," *International Journal of System Assurance Engineering and Management*, pp. 1–7, 2024.

[31] A. Saxena and A. Ranjan, "Improving sequence-to-sequence models for abstractive text summarization using meta heuristic approaches," *arXiv preprint arXiv:2403.16247*, 2024.

[32] A. F. Agarap, "Deep learning using rectified linear units (relu)," *arXiv preprint arXiv:1803.08375*, 2018.

[33] D. Hendrycks and K. Gimpel, "Gaussian error linear units (gelus)," *arXiv preprint arXiv:1606.08415*, 2016.

[34] D. Greene and P. Cunningham, "Practical solutions to the problem of diagonal dominance in kernel document clustering," in *Proc. 23rd International Conference on Machine learning (ICML'06)*. ACM Press, 2006, pp. 377–384.

Krishna Dheeravath B.Tech (CSE), M.Tech (CSE), is a Research scholar at Jawaharlal Nehru Technological University, Ananthapur, in the department of Computer Science and Engineering. He has 18+ years of relevant work experience in academics and is a lifetime member of ISTE. His research areas of interest are data mining, machine learning, artificial intelligence, deep learning, neural networks, and data analytics. He has attended seminars, international conferences and workshops. He has published more than ten research papers in international journals.

Dr. S Jessica Saritha is working as an Assistant Professor at Jawaharlal Nehru Technological University, Ananthapur, in the department of computer science and engineering. She has 20+ years of relevant work experience in academics. Her research areas of interest are High-dimensional Datasets, Data mining, Data warehousing, Machine learning, Artificial intelligence, Fuzzy Logic, Clustering Algorithm and Cloud Computing. She has attended seminars, international conferences and workshops. She has published more than thirty research papers in international journals.