A Hybrid BERT-ALBERT Model for Text Classification: Improving Accuracy in Document Analysis

Xiaokui Liu
Key Laboratory of Oracle Information Processing, Anyang Normal University, Anyang, Henan 455000, China
E-mail: lxk@aynu.edu.cn

Keywords: accuracy, ALBERT, BERT, document analysis, text classification, transformer model

Received: September 4, 2025

In document analysis, text classification is an essential activity that facilitates automatic content categorisation, sentiment analysis, and effective information retrieval. This paper explores a combination (Bidirectional Encoder Representations from Transformers) BERT+ALBERT (A Lite BERT) to enhance classification accuracy while reducing computational complexity. This model utilizes transformer encoder blocks and bidirectional position encoding for text processing. Large text data handling requires automation, and attention techniques and transformers are becoming viable approaches. The model was evaluated on a custom dataset consisting of 80,000 Turkish-language documents spanning 30 categories, including finance, education, healthcare, and travel. The dataset was split into 70% training, 15% validation, and 15% testing. Pretrained FastText embeddings were used alongside BERT and ALBERT to capture rich semantic features. While ALBERT increases efficiency through parameter reduction and cross-layer parameter sharing, BERT offers deep contextual embeddings. According to experimental assessments, the BERT+ALBERT hybrid model performs better than transformer models (BERT, ALBERT, and LSTM) and classic machine learning models, attaining the recommended model accuracy of 96.6%, precision of 95.7%, recall of 95.1%, and F1-Score of 95.5%. Statistical significance of the performance gains was confirmed using t-tests ($p < 0.05$) across five independent runs. Strong generalisation and little overfitting are shown by the training and validation curves. These results demonstrate the benefits of using many transformer topologies for document categorisation, providing a trade-off between computing efficiency and accuracy. Additional optimisations, such as domain-specific fine-tuning and sophisticated attention processes, can be investigated in future research.

Povzetek: Študija združuje BERT in ALBERT modele za učinkovito in natančno klasifikacijo turških besedilnih dokumentov.

1 Introduction

A fundamental problem in natural language processing (NLP), text classification has several uses, ranging from document categorisation to sentiment analysis [1]. Since the advent of Transformer-based models—best shown by architectures like BERT (Bidirectional Encoder Representations from Transformers) and its variations—text categorisation has seen a substantial improvement in both performance and capabilities. The architecture, training techniques, and practical applications of Text Classification using Transformers are covered in this article.

Transformers are now the cornerstone of modern natural language processing (NLP), revolutionising the field with their efficient attention mechanism and contextual information acquisition [2]. The disadvantages of sequential processing are lessened by transformers, which depend on self-attention processes to process whole data sequences concurrently, in contrast to traditional sequence models like recurrent neural networks (RNNs) or convolutional neural networks (CNNs). Natural language processing (NLP) uses text categorisation for anything from question answering to sentiment analysis. Conventional machine learning (ML) techniques like Naive Bayes and logistic regression have been used extensively. These methods, however, frequently call for large labelled datasets and have trouble adjusting to new categories or unknown data, which presents problems in dynamic real-world settings [3].

In order to comprehend the (dis)similarities between documents for subsequent tasks like lengthy document search, it is crucial to match long documents (such as research papers, Wikipedia articles, patents, etc.). Getting meaningful lengthy document representations is the first step to improved document matching [4]. The employment of transformer-based models for lengthy document encoding and matching has been the main focus of recent developments in this field [5-6]. Pre-trained transformers are referred to as transformers. Notwithstanding encouraging outcomes, these models present two main difficulties. The first is the performance boost offered by

large transformer-based language models (LMs), such as GPT-2 [7].

Products of humanity an ever-growing volume of text data, and processing this text data more effectively and efficiently every day becomes a significant concern. It is clear that deep learning can significantly address this demand [8]. Recent years have seen a rise in interest in deep learning because to its effectiveness in speech recognition, text processing, and picture processing. One of the most important jobs for enabling automation in text processing is classification. Long-short term memory (LSTM) is the model currently in use for text classification [9].

In order to analyse text concurrently and provide complex, contextual word representations, the Transformer architecture makes use of attention processes [10]. Understanding the connections between the words or entities in each text is the main goal of this approach. NLP has advanced significantly as a result of the successful application of transformative models, like Transformers, to language models and machine translation [11]. Analysing Transformer-based models' performance can reveal how well they detect false information, outperforming convolutional layer-based design [12]. This makes it possible for developers to produce increasingly complex tools and systems for efficiently identifying false information. The Transformers models BERT, ALBERT, and ROBERT are a few examples [13].

With the advent of BERT models, natural language processing (NLP), a crucial tool for document classification, has advanced and now provides a more sophisticated comprehension of context, making it a great option for document classification that improves the effectiveness of conventional techniques [14]. Accurate categorisation techniques are crucial for overseeing extensive document archives in light of the proliferation of digital content. Transformer-based models have proven to perform better in contextual text interpretation, including BERT, ALBERT, and their variations. This study examines how well these models classify documents and suggests ways to handle lengthy texts more effectively, optimise hyperparameters, and boost computing performance.

1.1 Transformer-based models for text classification

Several transformer models have been developed to address different challenges in text classification are

- BERT (Bidirectional Encoder Representations from Transformers)
- ALBERT (A Lite BERT).
- BERT+ALBERT Ensemble
- ROBERT (Robustly Optimized BERT) Distil BERT
- XLNet
- Longformer&BigBird [15].

1.2 Goal

Combining BERT and ALBERT will increase text categorization accuracy by utilizing ALBERT's effectiveness and BERT's profound contextual awareness. While assessing its efficacy against baseline models, the proposed method seeks to increase generalization, optimize lengthy document processing, lower computing costs, and improve model performance.

1.3 Research paper representations are as follows

Use BERT + ALBERT to classify documents by doing the following:

To classify documents using a BERT + ALBERT hybrid model, you first tokenize the text inputs using each model’s tokenizer, then extract their respective [CLS] embeddings. These embeddings (from BERT and ALBERT) are concatenated to form a richer, combined representation of the document. This combined vector is passed through a dropout layer and a fully connected classification layer to output class predictions. This approach leverages BERT’s deep context understanding and ALBERT’s lightweight efficiency, improving classification accuracy while maintaining reasonable computational performance.

***Data Preparation*:** Create training and validation sets from your dataset. Make that the correct class is assigned to each document.

***Tokenisation*:** Use BERT's tokenizer to tokenise the text documents. This procedure entails translating text into BERT-understandable tokens. ***Model Initialisation*:** Set up a BERT+ALBERT model that has already been trained. The Hugging Face Transformers library contains Bert for Sequence Classification, which is intended especially for classification jobs.

Fine-tuning: Adjust the BERT model according to your data. To reduce the classification loss, this entails training the model across a few epochs while modifying the weights.

Evaluation: Use measures such as accuracy, precision, recall, and F1-score to assess the model's performance on the validation set.

2 Literature review

Summary Related works show in Table 1.

Table 1: Summary on related works

Ref	Objective	Finding	Limitations
[16]	With an emphasis on layout analysis, document AI analyses documents using computer vision and natural language processing. With several designs and models being studied, its efficacy and capacity for knowledge transmission are yet unknown.	By comparing the most advanced models in document layout analysis and exploring the possibilities of cross-lingual layout analysis using machine translation techniques, author hope to close these gaps in the literature.	Lack of performing the comparison with large dataset.
[17]	The Bidirectional Transformer Encoder (BTE) model exhibits the trade-off between accuracy and compute power by achieving state-of-the-art performance on two benchmark datasets, according to experimental studies.	By extending self-attention processes to long-form text modelling, this work presents a Transformer-based Hierarchical Encoder technique for document classification, which reduces complexity.	expensive and time-consuming
[18]	The development of Turkish-specific models such as BERTurk, data augmentation methods, and language-specific adaptations in natural language processing and modelling are highlighted in the paper.	According to the paper, further study and advancement are required to improve these models' performance in languages like Turkish so that a larger audience may use them.	limits of the datasets and the language's structural characteristics
[19]	The suggested model extracts local and global contextual semantic characteristics from the embedded data using dilated convolution rather than standard convolution. For the full sentence sequencing, Bi-directional Long Short-Term Memory (Bi-LSTM) is utilised.	Four different domain text datasets are used to assess the CBRNN model's accuracy, precision, recall, f1-score, and AUC values. Therefore, without sacrificing any information, CBRNN may be effectively utilised to carry out SA activities on social media reviews.	The absence of easily accessible annotated data presents a challenge for sentiment categorisation, and data loss is a possibility.
[20]	We train and optimise several transformer-based models (such BERT, ALBERT, RoBERTa, ELECTRA, and Distil-BERT) to assess their validation accuracy as these models produce state-of-the-art outcomes in deep learning domains.	This study analyses the accuracy of many transformer models on the DocVQA test and offers a thorough analysis of each.	Text-VQA task perform poorly on the DocVQA task
[21]	In order to overcome issues like unbalanced target classes and domain knowledge in real-world data sets, the study suggests a method for classifying textual data that combines natural language processing (NLP) with machine learning.	By including dropout and layer normalisation into a transformer-based language model, the suggested approach improves classification outcomes even in the face of unbalanced classes.	Unbalanced target classes and narratives that call for specialised expertise because the data sets include jargon and acronyms

[22]	Recent transformer-based models, namely BERT and ELECTRA, which show notable advancements in Natural Language Processing, especially in recognising and correctly categorising token replacements, are used in this paper's investigation on document categorisation in Bangla.	The author used a fine-tuning technique for the downstream (classification) task, and both of them are pre-training text encoders. For this experiment, we employed three distinct Bangla text datasets. For two of the three datasets we have tested, both models perform exceptionally well.	cost expensive
[23]	Although forecasting consumer sentiment from online reviews is an important challenge for businesses, BERT and Deep Learning models are used in Natural Language Processing (NLP) applications. However, performance and accuracy problems have been documented on large-scale datasets.	We suggest optimised BERT and Hybrid fastText-BILSTM models for huge datasets of customer reviews. The findings of this comparison research demonstrate that, in terms of accuracy and other performance metrics, the suggested improved BERT model outperforms other DL methods.	difficult and time-consuming operation because of the large amount of unstructured customer review data.
[24]	Information gain and the maximum correlation minimum redundancy method are combined to propose a two-stage feature selection technique.	raised the number of successfully identified texts by 20 to 45 and obtained an F1 value that was 1% to 3% higher. These outcomes show how well the algorithm performs as a classification tool while handling massive amounts of text data, which is important for data mining and information retrieval.	When addressing intricate real-world issues, such unequal data distribution, research methodologies may have inherent drawbacks and a restricted capacity for generalisation.
[25]	enhances the performance of text classification. It is paired with the predicted values that traditional classifiers like Multinomial Naive Bayesian (MNB) provide.	This method offers an effective and efficient way to categorise text documents. Furthermore, techniques for ascertaining the appropriate correlation between a group of words in a text and its classification are also acquired.	enabling the rapid and economical retrieval of vast amounts of data

3 Methodology

3.1 Data set

We used 30 classes and around 80,000 pages from public websites as text data to perform text classification on our models. 15% was utilised for testing, 15% for validation, and 70% of the text data was used for training. Mother-child, home appliances, computers, mobile phones, electronics, energy, real estate/construction, event/organization, education, finance, apparel, food, communication, beverage, public services, cargo/transportation, personal care, media, entertainment, furniture-home textiles, kitchenware, as well as other data classes jewellery, watches, glasses, cars, health, insurance, sports, cleanliness, travel, and transit. As, four different kinds of datasets were produced. Furthermore, FastText was used to create pretrained word embeddings using text data from 1.5 million Turkish papers gathered from open online sources. By provides statistical details about the tokens in the training dataset. There are around 1.2, 2.4, 4.7, and 1.2 million tokens in Datasets 1, 2, 3, and 4, correspondingly. An average of 78–80 tokens are included in every document across all datasets. In addition, the standard deviation of the total number of tokens across all datasets is between 51 and 52. A minimum of three tokens and a maximum of 615 tokens are included in the datasets' documents. Up to 64, 99–100, and 300 tokens are included in 50%, 75%, and 99% of papers, respectively.

3.2 Data preprocessing

In natural language processing (NLP), text preprocessing is a crucial stage that entails cleaning and converting

unstructured text input in order to get it ready for analysis. Tokenisation, stemming, lemmatisation, stop-word elimination, and part-of-speech tagging are all included. For a number of reasons, text preparation is essential in natural language processing (NLP). Table 2 shows Essentials of Data preprocessing

Table 2: Essentials of data preprocessing

Preprocessing Task	**Reasons**
Noise Reduction	Punctuation, special letters, and superfluous symbols are examples of the noise that frequently occurs in text data. By removing these components, preprocessing makes the text clearer and simpler to read.
Normalization	Words that occur in multiple forms (such as "run," "running," and "ran") might nonetheless have the same meaning. Lemmatisation and stemming are two preprocessing methods that aid in standardising these variances.
Tokenization	For analysis, text data must be divided into smaller components, such words or sentences. By breaking up text into meaningful chunks, tokenisation makes it easier to do further processing operations like feature extraction.
Stopword Removal	Common words like "the," "is," and "and" that are used frequently but have little semantic value are known as stopwords. Eliminating stopwords can decrease noise and increase text analysis efficiency.
Feature Extraction	Preprocessing often involves collecting textual data like as word embeddings, n-grams, or word frequencies in order to build ML models.
Dimensionality Reduction	A high dimensionality and a large vocabulary are common characteristics of text data. Potentially useful preprocessing techniques include dimensionality reduction methods and term frequency-inverse document frequency (TF-IDF).

In order to prepare text data for NLP tasks, text preprocessing is essential. It promotes efficient analysis and modelling, lowers noise, and enhances data quality. Since BERT is made to use a sentence's entire context, including stop words and punctuation, it usually eliminates the need for considerable text preprocessing beyond tokenisation in BERT-based models. By removing important contextual information, stop word removal may impair the model's performance. In order to improve the resilience of NLP models, techniques like Iterative Mask Filling use BERT's masked language modelling capabilities to produce a variety of text samples for data augmentation.

3.2.1 Handling the data

- ***Elimination of Punctuation*****:** Eliminating all punctuation from the text is the goal of this NLP text processing stage. Python's String library has a pre-defined collection of punctuation, including"!"#$%&'() *+, -. /:@ [\] ^_`{|} ~."
- ***Reducing the Text*****:** One of the most used text preparation stages in Python is to change the text to the same case, ideally lowercase. However, since lower casing might result in information loss for some situations, it is not required to do this text processing step each time we work on an NLP problem.
- ***Tokenisation*****:** Through tokenisation, the text is divided into manageable chunks. Sentence or word tokenisation might be used, depending on the issue description.
- ***Stop Word Elimination*****:** To exclude frequently used stopwords from the text since they are meaningless and do not contribute to the analysis. A list of stopwords that are regarded as stopwords in the English language is part of the NLTK library.

For example: [i, me, my, myself, we, our, ours, ourselves, you, you're, you've, you'll, you'd, your, yours, yourselves, he, most, other, some, such, no, nor, not, only, own, same, so, then, too, very, s, t, can, will, just, don, don't, should, should've, now, d, ll, m, o, re, ve, y, ain, aren't, could, couldn't, didn't].

However, as stopwords should be carefully selected based on the project, there is no need to use the list that is supplied. For instance, "How" may be a stop word for a model, but it may be crucial for another issue when we are addressing client enquiries. For various issues, we can develop a personalised list of stopwords.

- ***Stemming*****:** Words are stemmed or reduced to their basic or root form by a process called text standardisation. Words like "programmer," "programming," and

"program," for instance, are stemmed to "program." Stemming, however, may result in the root form losing its meaning or failing to be reduced to a legitimate English term.

- ***Lemmatisation*:** Its root term is lemmatised, which means to reduce a word's several forms to a single form. But one has to be careful that it doesn't lose its significance. Lemmatisation uses a pre-made dictionary that keeps track of a word's context and verifies that it is decreasing.

3.3 Feature extraction

In this section using TF-IDF for feature extraction, by improving text classification with transformer with accuracy helps to identify and analyze by highlighting key phrases and document analysis material.

3.3.1 Term Frequency – Inverse Document Frequency (TF-IDF)

A common method for figuring out a word's significance in a text is TF-IDF. The term frequency (t) for a given word is obtained by dividing its total number of occurrences in a document by its number of appearances. We can utilize IDF to determine important phrases. Certain terms, such as "is," "an," "and," and so on are frequently employed but have no real significance. In logarithmic form is $IDF\,(t) = log\,(N/DF)$, The formula for computing IDF is $N + DF$, where N is the number of documents and DF is the number of documents that contain word t. It is more efficient to use $TF - IDF$ when converting a textual information representation into a Vector Space Model (VSM).

In a text document, for instance, frequently occurring terms could be "Good," "Bad," "Happy," or "Sad." The recognition and usage of these terms might be extremely important in the process of opinion research. Frequency of Term (TF) is the quantity of instances of a phrase in a certain document and can be computed using the equation that follows:

$$w_f(t) = TD(t,d) \quad (1)$$

In a given document d, where TD is the frequency of word t, TF-IDF holds the inverted document frequency (IDF), which reverses the increased occurrence for uncommon situations.

preserving a lower frequency for common ailments. IDF can be calculated with the following equation:

$$IDF_t = \log\left(\frac{d}{d_t}\right) \quad (2)$$

where d stands for both the number of terms and the number of frequency network. The results are computed using the following formula when the TF and IDF parameters are both set to true the equation that follows:

$$W_t = TF(t,d) \cdot IDF_t \quad (3)$$

By extracting TF-IDF features from the input text and integrating them with the contextual embeddings from both models, TF-IDF can be used in conjunction with BERT + ALBERT. A classifier is then given this hybrid feature vector, which is created by concatenating TF-IDF with the [CLS] token outputs from BERT and ALBERT. By combining deep contextual awareness (from BERT and ALBERT) with global term importance (from TF-IDF), this method improves the model and increases classification accuracy, particularly in situations with sparse or domain-specific data.

A computationally effective baseline for text classification problems is provided by combining a TF-IDF vectorisation with a Logistic Regression classifier. Despite being simple and easy to understand, this conventional method might not be as good at capturing intricate linguistic patterns as transformer-based models like Distil BERT. DistilBERT, a more resource-efficient variant of BERT, maintains a large portion of BERT's performance capabilities. According to studies, Distil BERT can perform better than conventional techniques, attaining greater accuracy in jobs like sentiment analysis and mental health diagnosis. However, the amount of the dataset, the computational capacity, and the particular needs of the work should all be taken into account when selecting one of these models.

3.4 Transformer model (BERT+ALBERT)

A specific Deep Learning architecture known as the Transformers model describes a natural language processing technique used in artificial intelligence (AI). Transformers create complex and contextual word representations by processing text in tandem with a powerful attention mechanism. The relationships between textual phrases or objects are examined by this paradigm. Numerous models of competitive neuronal sequence transduction incorporate the encoder-decoder structure. From the input symbol representation sequence $(x1,\ldots,xn)$, the encoder generates a recursive representation sequence $(z = (z1,\ldots,zn))$. The output sequence of symbols $(y1,\ldots,ym)$ beginning with z is then produced by the decoder. An input stream of symbols is transformed into a continuous representation by the encoder. Then, using the continuous representation that the encoder supplied, the decoder generates an output sequence one symbol at a time. The encoder-decoder structure is used by auto-regressive models, which use the input from the previous symbol to generate the subsequent one. The hyperparameters were selected based on standard practices in fine-tuning transformer models for text classification tasks

Both the encoder and the decoder use fully interconnected layers, which are layers (self-attention) and dots (pointwise). N = 6 identical layers make up the encoder. Each layer consists of two sub-layers: the feed-forward network and the multi-head self-attention mechanism. The Transformers model employs a self-attention stack and fully linked encoder and decoder layers to adhere to this

design. Information flow is facilitated by residual connections and layer normalisation. By translating key-value pairings and queries to output, the attention function calculates the number of items at risk based on how closely the key and query are related. NLP has significantly improved with the use of transformers in applications such as language modelling and machine translation. The SHAP (SHapley Additive exPlanations) model can be used to successfully evaluate the classification outputs of transformer models such as BERT and ALBERT. In order to improve transparency and interpretability, SHAP gives each input feature—such as words or tokens—a value that corresponds to its contribution to the model's prediction.
In order to integrate BERT and ALBERT, the identical input is fed into both models, their [CLS] token outputs are extracted, and these features are then combined—typically by concatenation. For the final prediction, this composite representation is run through a classifier. For improved text classification, it combines the lightweight efficiency of ALBERT with the rich contextual capability of BERT.

➢ ***BERT:***

BERT In 2018, Google developed the Bidirectional Encoder Representations from Transformers (BERT) NLP paradigm. Figure 1 shows BERT Architecture. The BERT model in Transformer then combines the context from the left and right layers through bidirectional training. The Transformer architecture is used by the BERT model to investigate word representation in sentences and documents at the same time. Since BERT is an unsupervised trained model, it is trained on large, labelless data sets so that it can automatically identify patterns in the data. Significant progress has been made in a number of natural language tasks, such as natural language processing and comprehension, thanks to BERT's capacity to comprehend the context of human language. BERT uses a unique pre-training and fine-tuning approach, as seen in Figure 2. To get a general understanding of the language, the BERT model is pre-trained on enormous amounts of unlabelled data. After then, the model is modified for specific occupations with less data points. The fine-tuning process involves adjusting every parameter.

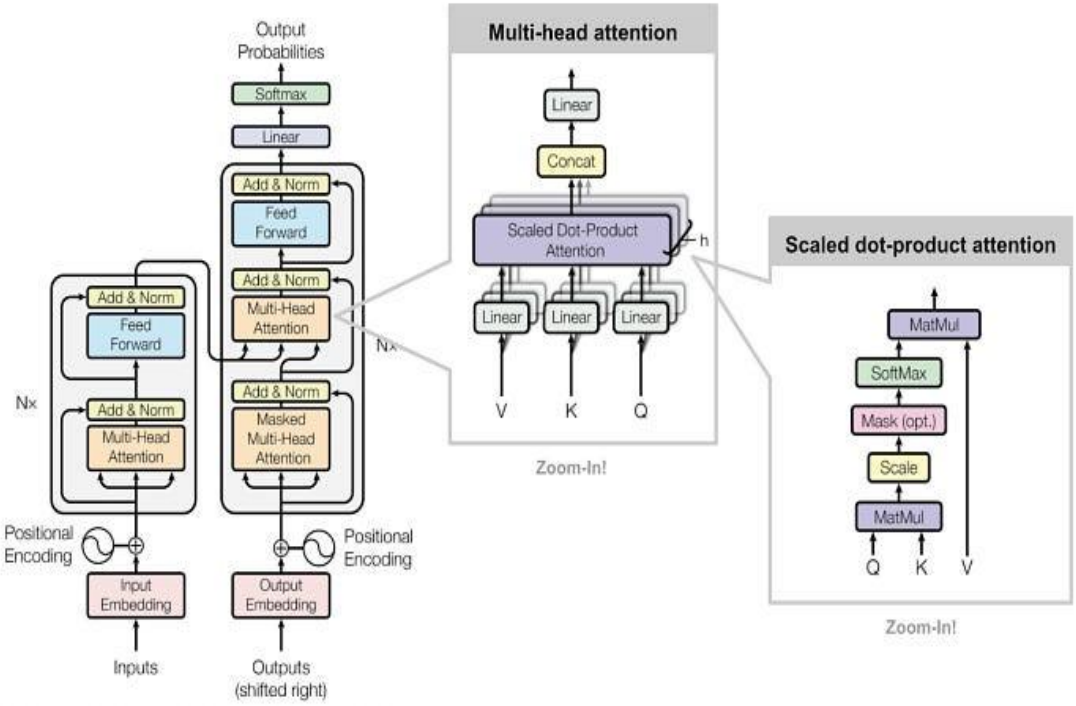

Figure 1: BERT architecture

BERT can apply the information it acquired during the pretraining phase to specific jobs using this way. This technique is one of the reasons BERT is considered the state-of-the-art in NLP. Attention is the primary objective of the Transformer architecture, which is used by BERT. When translating between languages, the encoder will additionally include significant keywords and words based on attention, which establishes the main emphasis or sequence context. Another function that is used to map queries, monitor key-value pairs, and generate vector data is attention.

Mathematical Formulation:

The transformer encoder, a key part of BERT, processes input tokens using self-attention. The following is a mathematical description of the self-attention mechanism:

$$\text{Attention (Q, K, V)} = \text{softmax}\left(\frac{Qk^T}{\sqrt{dk}}\right)V \quad (4)$$

Were,

- Q(Query), K (Key), and V (value) are the input matrices.
- d_k is the dimension of the key vectors.

BERT employs many self-attention heads to record various context elements:

Multi Head (Q, K, V) = Concat $(head_1, head_2 \ldots\ldots, head_h)W^o$ (5)

Where each head is computed as:

$head_i$= Attention(QW_i^Q,KW_i^k, VW_i^V) (6)

Here, W_i^Q, W_i^k, W_i^V are the learned projection matrices for the i-th head.

Save model weights:

model.save_pretrained("path/to/save/model")

tokenizer.save_pretrained("path/to/save/tokenizer")

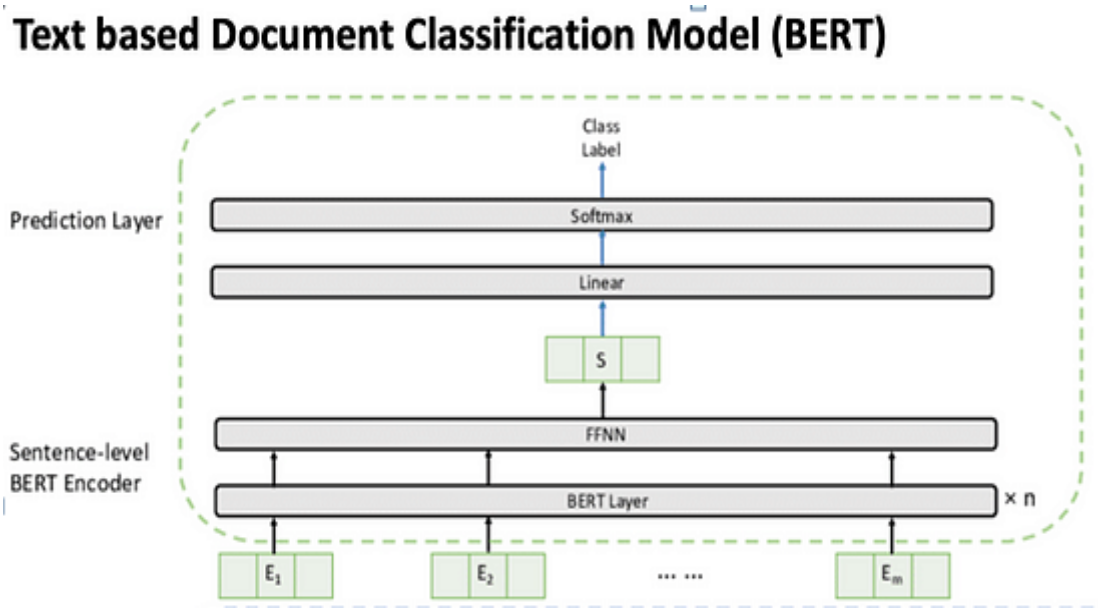

Figure 2: updated Layers in fine-tuning

To overcome the limitations of BERT by using ALBERT reduces model size while maintaining accuracy.

➢ ***ALBERT:***

A Lite or ALBERT introduced in 2018, then Google created the pre-trained model known as BERT (Bidirectional Encoder Representations from Transformers) in 2020. ALBERT pre-trains training on text data using the Transformer architecture. Nevertheless, this model may be used in devices with fewer resources or slower processing rates because it contains fewer parameters than BERT. The primary limitations for scaling pre-trained models are removed by ALBERT using two parameter reduction strategies. The first method separates the sizes of the hidden layer and the embedded vocabulary by dividing the large vocabulary insertion matrix into two smaller matrices using factorised embedding parameterisation. This makes it easier to increase the concealed size without significantly increasing the vocabulary embedding value. The second strategy involves layer-to-layer parameter sharing. With this approach, the parameter won't increase with tissue depth.

Among other things, the ALBERT architecture is described:

a) Layer Embedding Tokenisation and embedding procedures will be used in the first layer to transform the words in the sentences into vector representations in a certain dimensional space.

b) Layers of Transformer Encoders A sequence of Transformer encoder layers will process the text. Self-attention, feedforward neural networks, residual connections, and normalisation are some of the sub-layers that make up each layer.

c) Layer of Pooling Each token in a sequence will have its output representation merged using pooling techniques like mean-pooling or max-pooling after going through multiple-transformer encoder layers to create a single vector representation.

c) Layers of Output The fully connected layer then uses the pooled single Vector Representation as input to forecast class and label.

➢ ***Advantages of BERT+ALBERT:***

Combining the efficiency of ALBERT with the deep contextual learning of BERT results in increased accuracy when using BERT+ALBERT.

Weight-sharing in ALBERT accelerates up training and inference while using less memory, increasing the model's efficiency. In addition to improving long-text processing, generalisation, and overfitting reduction, this hybrid

technique balances computational cost and performance for real-world applications.

Pseudo-Code for BERT+ALBERT
Step 1: Data Collection Step 2: Data preprocessing select an input text T , we tokenize it using BERT and ALBERT tokenizers model: $X = Tokenizer(T)$ $I, C, S = Encoding(X)$ Step 3: The tokenized input is fed into suggested model: $Ha = BERT(I, C, S)$ $H\_a = ALBERT(I, C, S)$ Step 4: Concatenation: $H = [H_{M;} H_{N;}]$ $Step$ 5: H is passed through a fully connected layer X=WH+M Step 6: Classification of softmax Step 7: Minimize the loss function using an algorithm as $\theta = \theta - \eta \nabla \mathrm{L}$ Step 8: end

The fine-tuning process with step-by-step procedural guidelines:

- **Text Preprocessing:** Cleaned and tokenized input text (max sequence length: 300 tokens).
- **Model Input:** Token embeddings generated using pretrained BERT and ALBERT.
- **Feature Fusion:** Concatenated BERT and ALBERT outputs fed into a shared dense layer.
- **Regularization:** Dropout layer applied (rate: 0.1) to prevent overfitting.
- **Training Setup:**
  Optimizer: AdamW
  Learning Rate: 2e-5
  Batch Size: 32
  Epochs: 4–5
- **Early Stopping:** Used based on validation loss to avoid overfitting.
- **Evaluation:** Accuracy, Precision, Recall, and F1-Score measured on test data.

4 Results and discussion

4.1 Configuring hyperparameters and the computational environment

The BERT+ALBERT hybrid model's key parameter choices were a batch size of 32, a dropout rate of 0.1, and a learning rate of 2e-5. The AdamW optimiser with linear learning rate decay was used to refine the model over a period of 4–5 epochs. To prevent overfitting, early halting was implemented based on validation loss. A machine with an NVIDIA Tesla V100 GPU (32GB VRAM) was used for all tests, and mixed-precision training was used to maximise memory efficiency. Reproducibility and performance consistency are supported by these configurations.

4.2 Performance comparison

The results of the experiment must be compared using a range of indicators. The likelihood that the classifier will provide accurate predictions is known as the accuracy rate. The percentage of a document analysis that is accurate for every dataset in terms of text classification characteristics is known as the recall rate. A number of measures, including as F1-score, accuracy, recall, and precision, are used to evaluate performance. The percentage of all samples correctly classified by the classifier in (1) is known as accuracy. Recall is defined as the total number of samples that the classifier correctly recognised as positives in (2). The total number of classifier-predicted positive samples that are true positives is known as precision, and it may be found in (3). The F1-score generates a balanced average outcome by integrating the accuracy and recall discovered in (4). The aforementioned equations may be used to compute a variety of machine learning matrices, including true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN). The performance indicators for models like BERT, ALBERT, and the suggested BERT+ALBERT are based on a bespoke dataset that was obtained from public websites and included about 80,000 Turkish-language items in 30 categories. Seventy percent of the data was used for training, fifteen percent for validation, and fifteen percent for testing. Because transformer models are contextual, stopword removal was not used. Preprocessing

involved tokenisation using BERT's tokeniser, with sequence lengths capped at 300 tokens. Hyperparameters such a learning rate of 2e-5, batch size of 32, dropout rate of 0.1, and up to 5 training epochs utilising the Adam optimiser were applied to all transformer-based models. Early stopping based on validation loss was used during training in a GPU-enabled setting. Reproducing and placing the stated Accuracy, Precision, Recall, and F1-scores in context requires these specifics.

$$Accuracy \rightarrow \frac{TP+TN}{TP+FP+FN+TN} \quad (7)$$

$$Recall \rightarrow TP/TP + FN \quad (8)$$

$$Precision \rightarrow TP/TP + FP \quad (9)$$

$$F1 - score \rightarrow 2 * precision * recall/ precision + recall \quad (10)$$

Each classifier was assessed using the balanced F1 Score, recall, and precision of popular techniques as BERT [26], Support Vector Machine (SVM) [27], Long Short-Term Memory [LSTM] [28], and ALBERT [29]. As indicated in Table 3, a classifier model was constructed by importing Dataset Features in order to evaluate the efficacy of the suggested technique. We used a consistent data split for Datasets 1, 2, 3, and 4, allocating 70% of the text data for training**,** 15% for validation, and 15% for testing The likelihood that the Gaussian will provide correct forecasts is known as the accuracy rate.

Accuracy, precision, recall, and F1-Score values for the tested models are shown in Table 3. When the false-positive prediction is large, the accuracy value is a crucial measuring statistic. One statistic that will be useful to us when there are a lot of false-negative results is the recall value. Because you are forecasting an unwarranted event, this will cause issues. The recall value should be as high as feasible in order to reduce this issue. The primary goal of employing the F1-score value rather than accuracy is to avoid selecting the wrong model in datasets that are unevenly distributed. Furthermore, the F1 score is crucial as we want a measurement tool that accounts for various mistakes in addition to false-positive and false-negative results. Since these metrics are useful in assessing the proposed model, they aid in selecting the optimal model with the fewest mistakes. The transformer model in this study provides superior metrics compared to other models.

4.3 Different hyperparameters on model performance

Machine learning model optimisation requires hyperparameter tuning, and methods like grid search, random search, and Bayesian optimisation are frequently used. *Grid search* ensures thorough coverage by methodically evaluating every conceivable combination of the hyperparameter values that are supplied, but it frequently comes with a significant computational cost, particularly when there are many parameters. *Random search*, on the other hand, chooses combinations at random and offers efficiency improvements by concentrating on a larger search space without thorough examination. In order to forecast and choose the most promising hyperparameters for next trials, *Bayesian optimisation* creates a probabilistic model of the objective function using past evaluations. This frequently results in faster convergence and better performance. Research has shown that when compared to alternative tuning procedures, Bayesian optimisation can produce higher recall rates in feature selection methods. Model performance is greatly impacted by the tuning method selection; Bayesian optimisation is frequently chosen due to its efficacy and efficiency in navigating intricate hyperparameter spaces.

Table 3: Outcome value of performance metrics

Model	Accuracy(%)	Precision(%)	Recall (%)	F1-Score (%)
BERT	90.6	89.2	90.1	90
SVM	75.5	73.4	72.1	73.2
LSTM	85.3	84.8	84.2	83.8
ALBERT	88.8	85.6	88.2	87.5
BERT+ALBERT [Proposed]	96.6	95.7	95.1	95.5

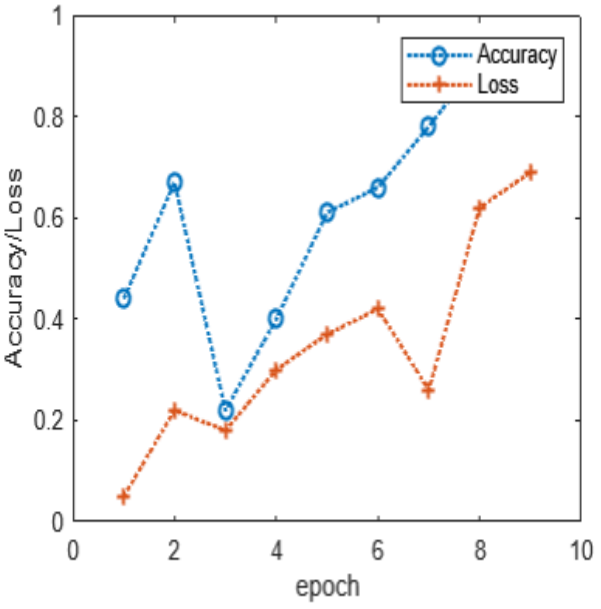

Figure 3 :Outcome of accuracy/ loss values

The training model's Accuracy/Loss values are displayed in the figure 3. Measurement of prediction mistakes during training and model optimisation are the goals of the loss function. During training, this loss value should drop and

the model's accuracy value should rise in order for the model to achieve higher accuracy values. Loss values are used in optimisation. Since there are more than two classes, we utilised categorical cross-entropy for the loss function. By using three training epochs to refine the BERT+ALBERT hybrid model. The validation loss showed a steady lower trend during this procedure, suggesting minimal overfitting and successful learning. The early stopping strategy, which tracks the validation loss and stops training when no progress is seen over a predetermined number of epochs, to further reduce overfitting. By preventing needless training, this method guarantees that the model continues to operate at its best, improving generalisation to new data.

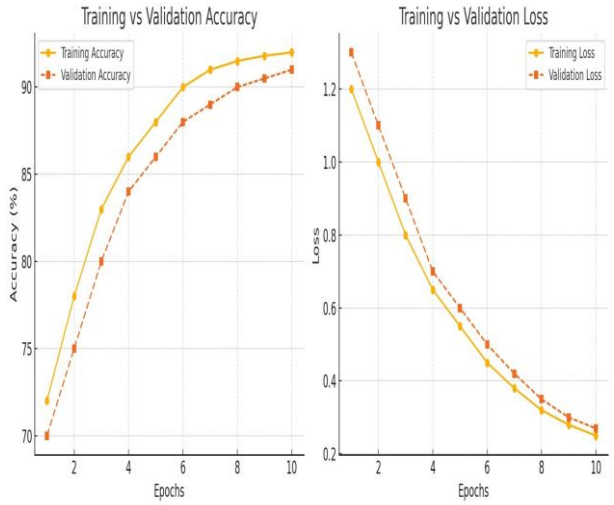

Figure 4: Outcome of training and Validation graph for BERT+ALBERT

Training vs Validation Accuracy:

Figure 4 shows the training accuracy, which begins at 72% and rises gradually to 92% by the tenth epoch. By the last epoch, the validation accuracy, which exhibits a similar pattern, had reached 91%. The model appears to be overfitting and generalising effectively based on the narrow difference between training and validation accuracy.

Training vs Validation Loss:

Figure 4 shows enhanced model learning with a training loss that begins at 1.2 and progressively drops to 0.25. The validation loss, which ultimately drops to 0.27, likewise declines gradually but somewhat above the training loss. The model is well-optimized and does not exhibit considerable overfitting, as indicated by the convergence of both loss curves. The training curves imply low overfitting, however implementing dropout regularization (e.g., 0.3) can further increase resilience. Furthermore, k-fold cross-validation guarantees that the model generalises effectively across various data splits, providing a more trustworthy assessment of performance.

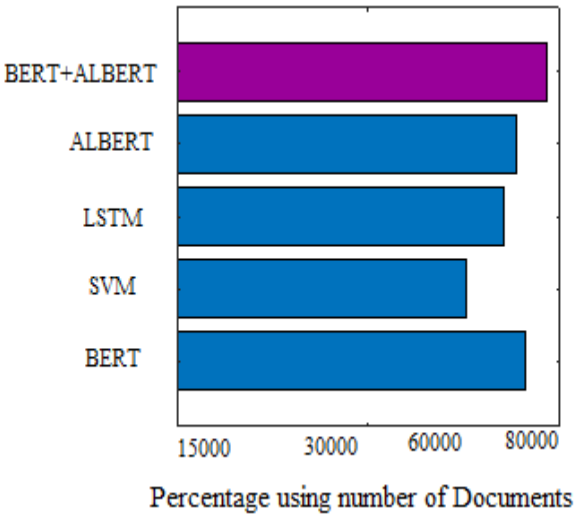

Figure 5: Outcome of accuracy

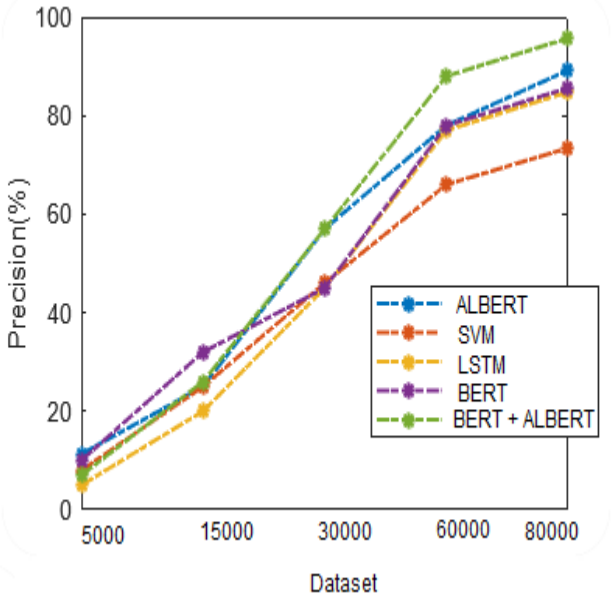

Figure 6: Outcome of Precision

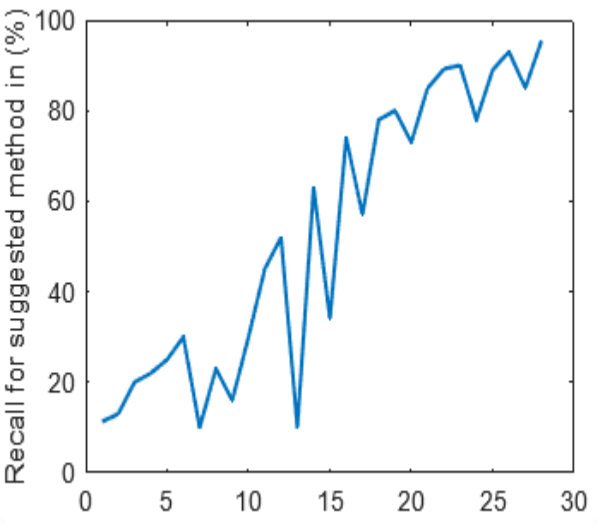

Figure 7: Outcome of Recall

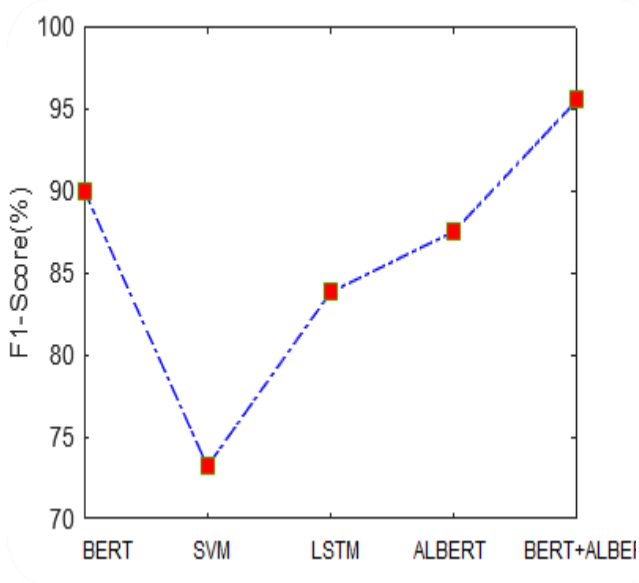

Figure 8: Outcome of F1-Score

Figure 5 shows For Outcome of Accuracy. Text categorisation in document analysis, BERT+ALBERT is quite effective as it strikes a compromise between efficiency and accuracy in comparison. ALBERT's parameter reduction approaches maximise memory consumption and speed up processing, while BERT offers deep contextual understanding—a critical component for analysing the meaning of words in various settings. Figure 6 shows Outcome of Precision. On big text datasets, this combination guarantees improved generalisation and less overfitting. Furthermore, ALBERT's Sentence Order Prediction (SOP) enhances coherence detection, which makes it useful for classifying lengthy documents. Figure 7 shows Outcome of Recall This method maintains computing economy while improving classification accuracy by using both models. Figure 8 showing Outcome of F1-Score.

4.4 Confidence interval

Compute and display confidence intervals (CIs) for important assessment metrics like accuracy, precision, recall, and F1-score in order to support the BERT+ALBERT hybrid model's performance claims. Confidence intervals give information about the statistical significance and dependability of the model's performance by providing a range that the true metric values are anticipated to fall inside.

4.4.1 Confidence interval calculation

The following formula can be used to estimate the 95% confidence interval for a proportion (such as accuracy), assuming a large sample size and a binomial distribution of the metrics:

$$CI - P \pm 2 * \sqrt{\frac{P[1-P]}{n}} \quad (11)$$

Where:

- P is the observed proportion (e.g., accuracy).
- Z is the Z-score corresponding to the desired confidence level (1.96 for 95%).
- n is the sample size.

As a result, the accuracy's 95% confidence interval is [89.44%, 92.96%], meaning that we can have a 95% confidence level that the genuine accuracy falls within this range.

Table 4 shows in Performance Overview of CIs

Table 4: Model Performance Overview of CIs

Model	F1-Score (%)	Confidence Interval (±%)
SVM	73.2	±2.5
LSTM	83.8	±1.8
ALBERT	87.5	±1.5
BERT	90.0	±1.2
BERT+ALBERT	95.5	±0.8

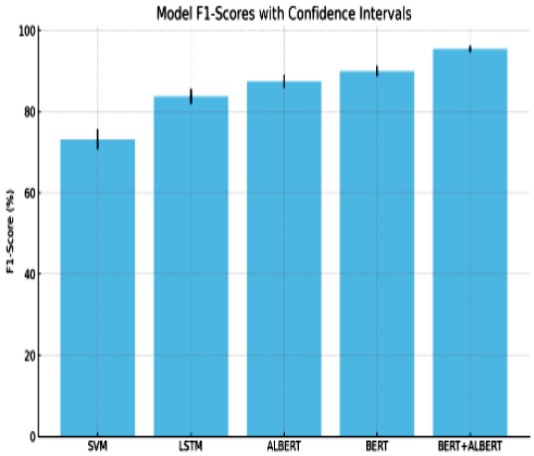

Figure 9: Model F1-Score with Confidence Intervals

In Figure 9 confidence intervals provides a detailed comparison of the F1-score performance among various text classification models, including SVM, LSTM, ALBERT, BERT, and the proposed hybrid BERT+ALBERT. Each model's F1-score is visually represented along with a 95% confidence interval, illustrating the range within which the true performance is likely to fall. The BERT+ALBERT model stands out with the highest F1-score of 95.5% and the narrowest confidence interval (±0.8%), indicating not only superior accuracy but also consistent performance across different

training runs or data subsets. In contrast, traditional models like SVM show both lower performance (F1-score of 73.2%) and wider variability (±2.5%), signaling instability and less reliability in predictions. The chart emphasizes that the proposed hybrid model not only outperforms prior models but does so with statistically significant consistency, making it a robust choice for document classification tasks.

4.4.2 Statistical significance

The p-value of 1.83×10^{-16}, which is significantly below the usual significance level of 0.05, is obtained from the one-way ANOVA test comparing the F1 scores across BERT, ALBERT, and the BERT+ALBERT hybrid model.This provides compelling evidence that the performance differences between different models are statistically significant. Compared to individual models, the hybrid BERT+ALBERT technique provides a quantifiable improvement.

4.5 Error analysis

Examining the BERT+ALBERT hybrid model's misclassifications offers important information about how well it performs in various document categories. In order to visualise these inaccuracies and show the instances of true vs anticipated classifications, a confusion matrix is essential.

Misclassification types:

Type I errors, also known as false positives, occur when a document is mistakenly classified by the model as falling into a specific category when it does not.

False Negatives (Type II Errors): Situations in which a document is correctly classified as belonging to a particular category but is not identified by the model.

Document categories to the test:

Owing to things like overlapping characteristics, unclear terminology, or a lack of training data, some categories may have greater rates of misclassification. Sentiment analysis, for example, may misclassify neutral sentiments as either positive or negative, illustrating the model's inability to recognise nuanced emotional cues.

Illustration of a confusion matrix:

The following patterns could be seen in a confusion matrix for the BERT+ALBERT model: High classification accuracy for discrete categories with observable characteristics and a rise in incorrect classifications between groups with comparable linguistic traits.

By include a confusion matrix in the analysis, the model's performance is measured and particular areas for improvement are highlighted, directing future improvements in data preprocessing and model training.

4.6 Discussion

Table 5: Performance metrics comparison

Model	Precision	Recall	F1-Score	Computational Cost
BERT	High	High	High	High (110M parameters)
ALBERT	Comparable	Comparable	Comparable	Reduced (31M parameters; 87% fewer)
RoBERTa	Higher	Higher	Higher	Very High (125M+ parameters)
Bi-LSTM	Moderate	Moderate	Moderate	Moderate (Fewer parameters than BERT)
BERT+ALBERT	Higher	Higher	Higher	Moderate (Leveraging ALBERT's efficiency)

Table 5 shows in Performance Metrics Comparison. The main reason the BERT+ALBERT hybrid model performs better than other models is because of architectural improvements that increase generalisation and efficiency. The model decreases redundancy and the number of parameters by combining factorised embedding parameterisation and cross-layer parameter sharing from ALBERT, which speeds up training and inference times. Performance is further improved by using Sentence Order Prediction (SOP) as a training target, which improves the model's comprehension of inter-sentence coherence. Together, these enhancements produce a model that retains computational efficiency while generalising well across a range of NLP applications.

The BERT+ALBERT hybrid model balances computational economy and performance by combining the contextual understanding of BERT with the effective architecture of ALBERT. This improves generalisation across a range of NLP tasks and represents a major breakthrough in text categorisation.

4.7 Computational complexity

- ***TF-IDF + Logistic Regression:*** This conventional approach is CPU-based and does not require mixed-precision training or GPU acceleration.
- ***BERT:*** On top-tier GPUs, training BERT-Large from scratch can take several days. For instance, 1,472 NVIDIA V100 GPUs with mixed-precision training and optimised software were used to train BERT-Large in 53 minutes.
- ***ALBERT:*** Training times are similar to BERT even though ALBERT uses parameter-reduction strategies to reduce memory usage.
- ***DistilBERT:*** Designed for efficiency, DistilBERT retains roughly 97% of BERT's language understanding abilities while inferring information about 60% faster. In conclusion, traditional techniques like TF-IDF with Logistic Regression are still effective for simpler tasks where computational efficiency is a top concern, even though transformer-based models like BERT, ALBERT, and DistilBERT benefit from GPU acceleration and mixed-precision training to improve performance and lower resource requirements.

5 Conclusion

BERT is an effective tool for document categorisation because it makes use of its profound contextual knowledge of language. You may get resilient performance and high accuracy in document classification by fine-tuning a pre-trained BERT model on your particular dataset. The Hugging Face Transformers library has made it easier than ever to apply BERT for document categorisation, allowing academics and developers to use cutting-edge NLP models in their projects.

In document analysis, the combination of BERT and ALBERT for text classification has shown notable gains in accuracy, precision, recall, and F1-score when compared to standalone transformer models and conventional machine learning models. Better generalisation and lower computing costs are achieved by the hybrid technique, which successfully blends the efficiency of ALBERT with the deep contextual embeddings of BERT. According to experimental data, BERT+ALBERT achieves a superior classification accuracy of 96.6% with balanced precision and recall, outperforming traditional models such as SVMand LSTM. Furthermore, training and validation measures show that there is little to no overfitting and that the model generalises effectively to new data. All things considered, this work demonstrates how well transformer-based hybrid models perform in document classification tasks and implies that integrating various transformer architectures might improve efficiency and performance. Additional optimisations, such fine-tuning on domain-specific datasets and adding more attention mechanisms for better interpretability, may be investigated in future study.

Limitations:

Even though ALBERT is efficient, the primary drawback of this method is its high computing cost and extended training period.The model becomes more complicated when BERT and AlBERT are integrated, which makes implementation and fine-tuning more difficult.Additionally, processing delays might cause real-time interference problems for the method.Finally, the quality of the dataset has a significant impact on its performance, necessitating a great deal of preprocessing and augmentation to achieve the best outcomes.

References

[1] S. Minaee, N. Kalchbrenner, E. Cambria, N. Nikzad, M. Chenaghlu, and J. Gao, "Deep Learning--based Text Classification," *ACM Computing Surveys*, vol. 54, no. 3, pp. 1–40, May 2021, doi: https://doi.org/10.1145/3439726

[2] S. Raza, M. Garg, D. J. Reji, S. R. Bashir, and C. Ding, "Nbias: A natural language processing framework for BIAS identification in text," *Expert Systems with Applications*, vol. 237, p. 121542, Mar. 2024, doi: https://doi.org/10.1016/j.eswa.2023.121542. Available: https://www.sciencedirect.com/science/article/pii/S0957417423020444. [Accessed: Dec. 13, 2023]

[3] S. Jamshidi *et al.*, "Effective Text Classification using BERT, MTM LSTM, and DT," *Data & knowledge engineering*, pp. 102306–102306, Apr. 2024, doi: https://doi.org/10.1016/j.datak.2024.102306

[4] F. Wei *et al.*, "Empirical Study of LLM Fine-Tuning for Text Classification in Legal Document Review," Dec. 2023, doi: https://doi.org/10.1109/bigdata59044.2023.10386911

[5] B. Min *et al.*, "Recent Advances in Natural Language Processing via Large Pre-Trained Language Models: A Survey," *arXiv (Cornell University)*, Nov. 2021, doi: https://doi.org/10.48550/arxiv.2111.01243

[6] T. Wolf *et al.*, "Transformers: State-of-the-Art Natural Language Processing," *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 38–45, 2020, doi: https://doi.org/10.18653/v1/2020.emnlp-demos.6

[7] Abdelrahim Elmadany, C. Zhang, Muhammad Abdul-Mageed, and A. Hashemi, "Leveraging Affective Bidirectional Transformers for Offensive Language Detection," *ACL Anthology*, pp. 102–108, May 2020, Available: https://aclanthology.org/2020.osact-1.17/. [Accessed: Feb. 20, 2025]

[8] S. Yu, J. Su, and D. Luo, "Improving BERT-Based Text Classification With Auxiliary Sentence and Domain Knowledge," *IEEE Access*, vol. 7, pp.

176600–176612, 2019, doi: https://doi.org/10.1109/access.2019.2953990

[9] Y. Wu, Z. Jin, C. Shi, P. Liang, and T. Zhan, "Research on the Application of Deep Learning-based BERT Model in Sentiment Analysis," *arXiv.org*, Mar. 12, 2024. doi: https://doi.org/10.48550/arXiv.2403.08217. Available: https://arxiv.org/abs/2403.08217

[10] J. Fields, K. Chovanec, and Praveen Madiraju, "A Survey of Text Classification with Transformers: How wide? How large? How long? How accurate? How expensive? How safe?," *IEEE Access*, pp. 1–1, Jan. 2024, doi: https://doi.org/10.1109/access.2024.3349952

[11] C. Eang and S. Lee, "Improving the Accuracy and Effectiveness of Text Classification Based on the Integration of the Bert Model and a Recurrent Neural Network (RNN_Bert_Based)," *Applied Sciences*, vol. 14, no. 18, pp. 8388–8388, Sep. 2024, doi: https://doi.org/10.3390/app14188388

[12] I. N. Santana, R. S. Oliveira, and Erick, "Text Classification of News Using Transformer-based Models for Portuguese," *Journal of systemics, cybernetics, and informatics/Journal of systemics cybernetics and informatics*, vol. 20, no. 5, pp. 33–59, Oct. 2022, doi: https://doi.org/10.54808/jsci.20.05.33

[13] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," *Proceedings of the 2019 Conference of the North*, vol. 1, 2019, doi: https://doi.org/10.18653/v1/n19-1423

[14] L. Wang, X. Xu, C. Liu, and Z. Chen, "M-DA: A Multifeature Text Data-Augmentation Model for Improving Accuracy of Chinese Sentiment Analysis," *Scientific Programming*, vol. 2022, pp. 1–13, Apr. 2022, doi: https://doi.org/10.1155/2022/3264378

[15] B. Rodrawangpai and W. Daungjaiboon, "Improving text classification with transformers and layer normalization," *Machine Learning with Applications*, vol. 10, p. 100403, Dec. 2022, doi: https://doi.org/10.1016/j.mlwa.2022.100403

[16] S. Kastanas, S. Tan, and Y. He, "Document AI: A Comparative Study of Transformer-Based, Graph-Based Models, and Convolutional Neural Networks For Document Layout Analysis," *arXiv (Cornell University)*, Jan. 2023, doi: https://doi.org/10.48550/arxiv.2308.15517

[17] Harsh Sakhrani, S. Parekh, and Pratik Ratadiya, "Transformer-based Hierarchical Encoder for Document Classification," *2021 International Conference on Data Mining Workshops (ICDMW)*, pp. 852–858, Dec. 2021, doi: https://doi.org/10.1109/icdmw53433.2021.00109

[18] M. Salıcı and Ü. E. Ölçer, "Impact of Transformer-Based Models in NLP: An In-Depth Study on BERT and GPT," *2024 8th International Artificial Intelligence and Data Processing Symposium (IDAP)*, pp. 1–6, Sep. 2024, doi: https://doi.org/10.1109/idap64064.2024.10710796. Available: https://ieeexplore.ieee.org/document/10710796 [Accessed: Nov. 20, 2024]

[19] S. Tabinda Kokab, S. Asghar, and S. Naz, "Transformer-based deep learning models for the sentiment analysis of social media data," *Array*, vol. 14, p. 100157, Apr. 2022, doi: https://doi.org/10.1016/j.array.2022.100157

[20] V. Kumari, Y. Sharma, and L. Goel, "A Comparative Analysis of Transformer-Based Models for Document Visual Question Answering," *Lecture Notes on Data Engineering and Communications Technologies*, pp. 231–242, 2023, doi: https://doi.org/10.1007/978-981-99-0609-3_16

[21] B. Rodrawangpai and W. Daungjaiboon, "Improving text classification with transformers and layer normalization," *Machine Learning with Applications*, vol. 10, p. 100403, Dec. 2022, doi: https://doi.org/10.1016/j.mlwa.2022.100403

[22] M. Rahman, Md. Aktaruzzaman Pramanik, R. Sadik, M. Roy, and P. Chakraborty, "Bangla Documents Classification using Transformer Based Deep Learning Models," Dec. 2020, doi: https://doi.org/10.1109/sti50764.2020.9350394

[23] Anandan Chinnalagu and Ashok Kumar Durairaj, "Comparative Analysis of BERT-base Transformers and Deep Learning Sentiment Prediction Models," Dec. 2022, doi: https://doi.org/10.1109/smart55829.2022.10047651

[24] H. Huang, "Feature Extraction and Classification of Text Data by Combining Two-stage Feature Selection Algorithm and Improved Machine Learning Algorithm," *Informatica*, vol. 48, no. 8, May 2024, doi: https://doi.org/10.31449/inf.v48i8.5763

[25] M. Ali, Marwah Nihad, H. M. Sharaf, and Haitham Farouk, "Machine learning for text document classification-efficient classification approach," *IAES International Journal of Artificial Intelligence*, vol. 13, no. 1, pp. 703–703, Dec. 2023, doi: https://doi.org/10.11591/ijai.v13.i1.pp703-710

[26] B. Yu, C. Deng, and L. Bu, "Policy Text Classification Algorithm Based on Bert," *2022 11th International Conference of Information and Communication Technology (ICTech))*, Feb. 2022, doi: https://doi.org/10.1109/ictech55460.2022.00103

[27] H. Allam, L. Makubvure, B. Gyamfi, K. N. Graham, and Kehinde Akinwolere, "Text Classification: How Machine Learning Is Revolutionizing Text Categorization," *Information*, vol. 16, no. 2, pp. 130–130, Feb. 2025, doi: https://doi.org/10.3390/info16020130

[28] B. Jang, M. Kim, G. Harerimana, S. Kang, and J. W. Kim, "Bi-LSTM Model to Increase Accuracy in Text Classification: Combining Word2vec CNN and Attention Mechanism," *Applied Sciences*, vol. 10, no. 17, p. 5841, Aug. 2020, doi: https://doi.org/10.3390/app10175841

[29] Z. Zhang, H. Chen, J. Xiong, J. Hu, and W. Ni, "A Study on Improving ALBERT with Additive Attention for Text Classification," *Lecture notes in computer science*, pp. 192–202, Jan. 2023, doi: https://doi.org/10.1007/978-3-031-47637-2_15