Knowledge-Based Systems 238 (2022) 107872

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

Hierarchical BERT with an adaptive fine-tuning strategy for document classification ☆

Jun Kong, Jin Wang *, Xuejie Zhang

School of Information Science and Engineering, Yunnan University, Kunming, China

ARTICLE INFO

Article history:
Received 30 January 2021
Received in revised form 6 November 2021
Accepted 2 December 2021
Available online 10 December 2021

Keywords:
Document classification
Hierarchical BERT
Adaptive fine-tuning strategy
Pretrained language model

ABSTRACT

Pretrained language models (PLMs) have achieved impressive results and have become vital tools for various natural language processing (NLP) tasks. However, there is a limitation that applying these PLMs to document classification when the document length exceeds the maximum acceptable length of the PLM since the excess portion is truncated in these models. If the keywords are in the truncated part, then the performance of the model declines. To address this problem, this paper proposes a hierarchical BERT with an adaptive fine-tuning strategy (HAdaBERT). It consists of a BERT-based model as the local encoder and an attention-based gated memory network as the global encoder. In contrast to existing PLMs that directly truncate documents, the proposed model uses a part of the document as a region, dividing input document into several containers. This allows the useful information in each container to be extracted by a local encoder and composed by a global encoder according to its contribution to the classification. To further improve the performance of the model, this paper proposes an adaptive fine-tuning strategy, which dynamically decides the layers of BERT to be fine-tuned instead of fine-tuning all layers for each input text. Experimental results on different corpora indicated that this method outperformed existing neural networks for document classification.

1. Introduction

With the rapid development of the Internet, a variety of social media texts are growing explosively. Classifying such a large volume of documents is essential to make them more manageable and, ultimately, obtain valuable insights that are related to politics, business and manufacturing. Due to the difficulty and inefficiency of human agents in managing the incoming volume of text, document classification techniques can automatically assign each document correctly to one or more categories based on its content and features. It has a wide range of applications in the fields of news topic classification [1], sentiment analysis [2,3], subject labeling of academic papers [4], and text summarization [5].

Practically, document classification has its own internal structural properties that distinguish it from sentence classification. Documents are often composed of multiple sentences and contain more words than conventional sentences. Due to the use of rhetoric, important information in the document may be distributed in different local positions. There are more complex and ambiguous semantic relationships between sentences, making document classification a challenging task.

Deep learning models have made impressive progress in various NLP tasks. Based on distributed word representation, several neural networks have been proposed for document classification, including convolutional neural networks (CNNs) [6,7], recurrent neural networks (RNNs) [8,9], gated recurrent unit (GRU) networks [10] and long short-term memory (LSTM) networks [11]. CNNs have been successful in computer vision and have also been used in document classification. Assuming that each token in a document does not contribute equally to the classification, self-attention [12] and dynamic routing [13,14] a process to automatically align text and emphasize important tokens have been proposed, further improving the performance of CNNs and GRU and LSTM networks. Moreover, hierarchical attention network [15] been proposed for document classification. It performed semantic modeling with GRU at the sentence level and the document level, respectively. However, this can lead to loss of the syntactic dependencies of tokens in the context of the sentence.

Recently, pretrained language models, including BERT [16], ALBERT [17] and RoBERTa [18], have successfully completed various NLP tasks. These PLMs eliminate the need to build a model from scratch when dealing with downstream tasks and can adopt

☆ The code for this paper is availabled at: https://github.com/JunKong5/HAdaBERT.

* Corresponding author.
E-mail addresses: kongjun@mail.ynu.edu.cn (J. Kong), wangjin@ynu.edu.cn (J. Wang), xjzhang@ynu.edu.cn (X. Zhang).
URLs: https://www.ise.ynu.edu.cn/teacher/973 (J. Wang), https://www.ise.ynu.edu.cn/teacher/711 (X. Zhang).

https://doi.org/10.1016/j.knosys.2021.107872
0950-7051/

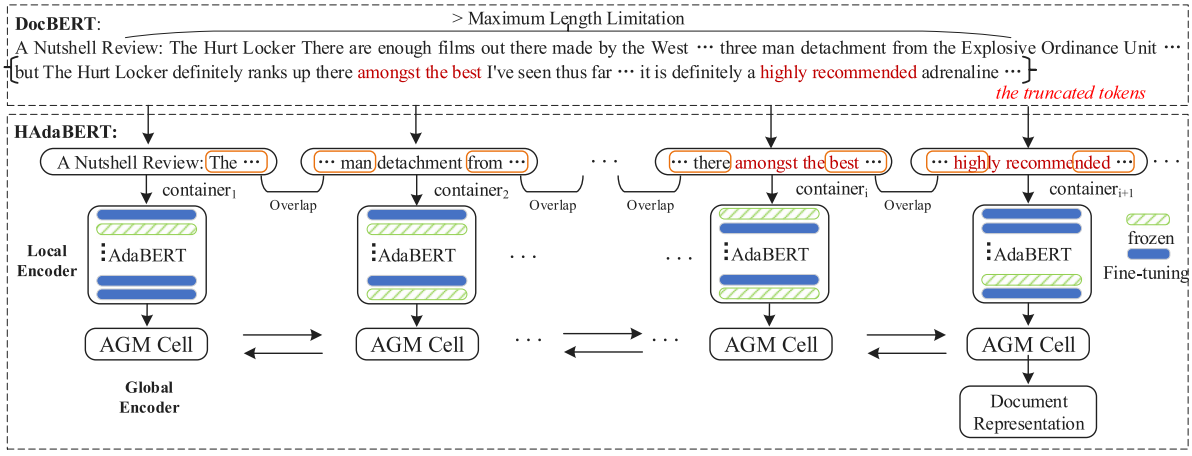

Fig. 1. An example from DocBERT (truncated to 512 as input). The key information of document classification is in the truncated part. These terms will never appear with BERT in the processing range and therefore cannot be classified correctly. Our method effectively captures local and global information for classification.

transformers [19] and self-attention mechanisms [12] to learn high-quality contextual representations of the texts using transfer learning. Typically, PLMs are first fed a large amount of unannotated data and are trained by either a masked language model or next sentence prediction to learn the usage of various words and how the language is written in general. Then, the models are transferred to another task where they are fed another smaller task-specific dataset. For document classification, the input sequence can be very long, but the maximum input length of the BERT model is limited in most cases [20]. Once the documents exceed the preset maximum input length, the excess part of these documents are directly truncated. Fig. 1 shows an example of the DocBERT model that truncates the input documents to 512 to fine-tune the BERT and predict document categories [20]. As shown in Fig. 1, the strongly subjective phrase *amongst the best* and *highly recommended* will be truncated, which may lead to an incorrect classification of the model.

Another obvious issue of applying BERT to document classification is computational consumption. That is, as the length of the input sequence increases, the demand for computational resources increase dramatically since the time complexity for fine-tuning each transformer layer is exponential with respect to the input length. Recent studies have indicated that fine-tuning all layers (usually 12 layers in BERT) does not necessarily lead to the best performance but increases the training costs [21]. Based on this, one intuitive method is to select the layers that should be fine-tuned in either the training or inferring phase. However, existing works use only a manual strategy for each sample [22] and fail to dynamically change the layer selection strategy for different input samples.

In this paper, the hierarchical BERT model with an adaptive fine-tuning strategy was proposed to address the aforementioned problems. As shown in Fig. 2, the HAdaBERT model consists of two main parts to model the document representation hierarchically, including both local and global encoders. Considering a document has a natural hierarchical structure, i.e., a document contains multiple sentences, and each sentence contains multiple words, we apply the local encoder to learn the sentence-level features while the global encoder to compose these features as the final representation. In contrast to existing PLMs directly truncating the document to the maximum input length, the proposed HAdaBERT uses part of the document as a region. Instead of using one sentence in each container for the local encoder, we introduced a container division strategy to get effective local information such that the syntactic dependencies are preserved. Then, the key and useful information in the containers can be extracted by the local encoder. Then, the global encoder learns to sequentially compose the syntactic relationships between the containers with an attention-based gated memory network. By using both encoders in a hierarchical architecture, the model effectively captures local information and long-term dependencies in long documents. Furthermore, an adaptive fine-tuning strategy for each input sample is proposed to apply a policy network that adaptively selects the optimal fine-tuned layers of BERT to improve model performance during training and inference. Such a strategy is an automatic and general method that can be extended to other pretrained language models.

The empirical experiments were conducted on several corpora, including the AAPD, Reuters, IMDB, and Yelp-2013 datasets. The comparative results show that the proposed HAdaBERT model outperformed several existing neural networks in document classification. In addition, the model can effectively address the limitations of previous methods for document classification. It is also competitive with existing models for learning the representations of short sentences. Another observation is that the adaptive fine-tuning strategy achieves the best performance improvement by dynamically selecting layers for fine-tuning.

The remainder of this paper is organized as follows. Section 2 presents and reviews the related works on document classification. Section 3 describes the proposed hierarchical BERT model and the strategy of adaptive fine-tuning. Comparative experiments are conducted in Section 4. Finally, conclusions are drawn in Section 5.

2. Related works

Document-level classification is a fundamental and challenging task in natural language processing. This section presents a brief review of existing methods for document-level classification, including conventional, hierarchical and pretrained neural networks.

2.1. Conventional neural networks

The deep neural network models commonly used for document classification tasks are CNNs [23–25] and RNNs [9,26]. These models represent documents as distributed representations with semantic and syntactic information. Due to the great success in the field of computer vision and the parallel convolutional computation, CNNs have been successfully applied to text classification. Kim et al. [6] proposed CNNs for document classification, using a convolutional layer with multiple dimensions for local

feature information extraction and then using maxpooling to select the features with max values on text sequences. Using CNNs, Zhang et al. [7] applied character-level representation as a raw signal and used a one-dimensional convolutional neural network to process it. There is no need for learning semantic information at the word level when processing document data. Liu et al. [27] extended CNNs with a dynamic maximum pooling layer and an additional full-connection layer for multilabel document classification. These CNNs can extract local *n*-gram features of different dimensions. However, the limitation of the convolutional kernel size may lose the contextual information of the text in modeling the representation of long text sequences.

Recurrent models, such as RNNs and GRU and LSTM networks, can capture long-term dependencies in a text sequence. In RNNs [26,28], the output of a neuron relies on the hidden state of the previous step and the input of the current step. However, the problem of gradient explosion/vanishing in RNNs may cause the model to fail to capture long-term dependency. To address this issue, LSTM [29] and GRU [10] networks use memory cells that allow explicit memory update and delivery. Both models learn contextual information that are beneficial for capturing the semantics of long texts. Furthermore, Tai et al. [11] proposed a tree-LSTM network that incorporates syntactic information according to a dependency parse tree. Zhou et al. [30] proposed a recurrent CNN (RCNN) model that extends existing convolutional layers as recurrent architecture. Yang et al. [4] proposed a sequence-to-sequence model to capture the intrinsic relationship between multiple labels for multilabel document classification. In addition, memory networks [31–33] introduce an external memory similar to the memory cells of a LSTM network, which is capable of storing information for long texts.

Based on the assumption that not all words contribute equally to the final classification, an attention mechanism [12] has been introduced to select important information and thus improve the performance of models. Based on self-attention, CNNs and GRU and LSTM networks [34–36] can better perform multiple NLP tasks. The attention mechanism captures important information about the text to better learn the text representation, and it can work with various neural encoders well. Considering this approach, Zheng et al. [37] proposed a self-interactive attention-based mechanism for document classification. To further enhance the attention mechanism, Sabour et al. [13] proposed a capsule network that assigns a weight by measuring the agreement between input and output representations. There are two limitations with the attention mechanism. First, it cannot perceive the position information, leading to the inability to learn high-level lingual features, e.g., a turn of expression or progressive expression, within sentences. Additionally, it may have difficulties emphasizing keywords in very long documents when the number of other less important words in the context increases because the weights of the keywords assigned via the attention mechanism become diluted and difficult to leverage during model training.

2.2. Hierarchical neural networks

Recently, hierarchical neural networks [15,38] have been widely used for document classification. To obtain the syntactic information between sentences, Tang et al. [38] designed a hierarchical architecture to learn document representation using a CNN and LSTM network with word embeddings and then adopted a GRU network to encode syntactic information within documents. Using a similar idea, Xu et al. [39] proposed a cached LSTM model that applied a group of LSTM cells with different forgetting gates. The gates assigned high forgetting rates can learn local features, while gates with low forgetting rates can capture global features. For predicting affective ratings, Wang et al. [40,41] proposed a tree-structured regional CNN-LSTM model to incorporate syntactic information from a dependency parse tree.

To extend the self-attention mechanism into a hierarchical architecture, Yang et al. [15] proposed a hierarchical model with an attention mechanism at both the sentence and document levels. Yin et al. [42] proposed a hierarchical interactive attention-based model to construct the document representation by taking the task as a comprehension problem. Additionally, for product reviews, several works proved that it is useful to incorporate sentiment information with extra features, such as user and product messages, for final polarity classification [43–45].

2.3. Pretrained neural networks

By using the self-attention mechanism, transformer [19] do not directly use the sequential structure of a GRU and LSTM network but allow the model to be trained in parallel. To further improve transformer, Gong et al. [46] proposed a multi-head attention algorithm and hierarchical structure introduced into the self-attention mechanism. However, the transformer may ignore the local dependency and the temporal relationship of the text. As the length of the sequence increases, the storage and calculation complexity of the transformers increase rapidly.

Based on the transformer, several pretrained language models (PLMs) have been proposed for text classification [16,47]. The basic idea is to use a large amount of unlabeled data to pretrain the language model through either a masked language model or next sentence prediction to learn both syntactic and semantic information by using unsupervised training. Then, the trained model can be transferred to another task where it is fed another smaller task-specific dataset. Using this approach, Adhikari et al. [20] first introduced the BERT model [16] with a truncating strategy as DocBERT for document classification, which achieved good performance. Liu et al. [18] proposed RoBERTa, which uses a quick fine-tuning BERT training procedure, to successfully use segment-wise predictions to conduct document classification and showed improved results. Lan et al. [17] proposed the ALBERT model, which introduces two parameter-reduction techniques of factorized embedding parameterization and cross-layer parameter sharing to lower the memory consumption and increase training speed.

For document classification, the existing PLMs suffer from input length limitations. Using a truncation strategy, the model loses part of the information for classification, which can have profound effects when the important content falls in the truncated parts, and the performance of the document classification model will decrease. Another issue is that both pretraining and fine-tuning of the BERT-based model incur considerable computing resource costs. With increasing input length, the consumed resources increase dramatically and attention weight is diluted.

3. Hierarchical BERT with adaptive fine-tuning

In this section, the hierarchical BERT with an adaptive fine-tuning strategy (HAdaBERT) model is described in detail. Fig. 2 shows the overall architecture of the proposed model, which consists of two main parts to model the document representation hierarchically, including both local and global encoders. The input document is first divided into several containers. In each container, a BERT-based model is fine-tuned to extract high-quality local features. Taking the sequential local features as input, an attention-based gated memory network (AGM) is applied as a global encoder to learn the long-term dependencies between containers. The proposed HAdaBERT model can effectively capture both local and long-range information for document classification. To efficiently fine-tune BERT in the local encoder, we

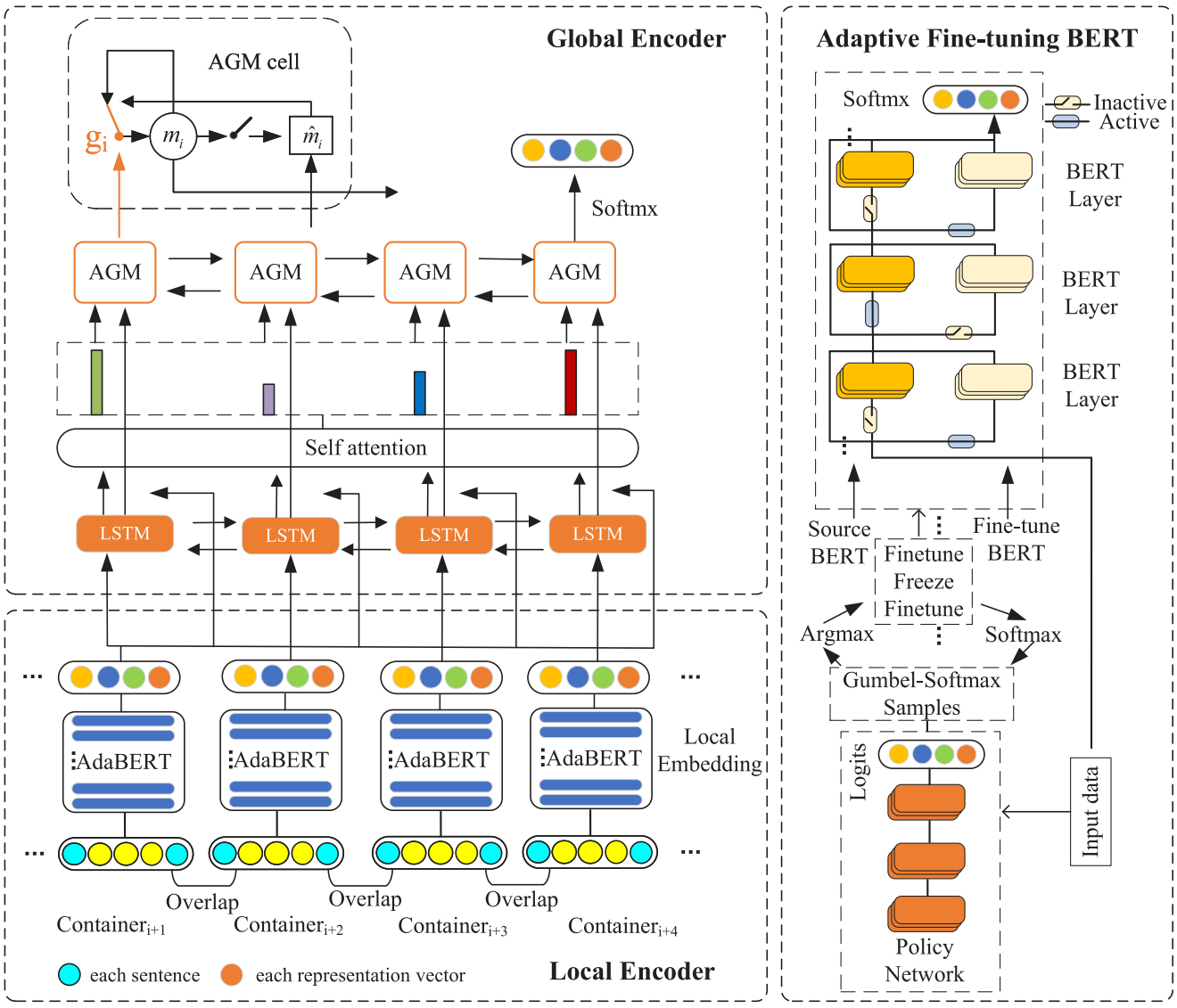

Fig. 2. The overall architecture of the proposed hierarchical BERT model with adaptive fine-tuning strategy.

introduce a policy network to adaptively decide the layers that should be selected for fine-tuning. Specific details of the different structures and components are described in the following subsections.

3.1. Local encoder

For each given document, the hierarchical BERT model uses part of the document as a region and divides the document into v containers, denoted as $D = [r_1, r_2, \ldots, r_v]$. Each container includes a sequence of tokens, $r_i = [x_{i1}, x_{i2}, \ldots, x_{ic}]$, where c is the capacity of the container. In an intuitive region division strategy, each individual sentence in the document is taken as a region. For example, a document that consists of three sentences has three regions. However, this simple strategy is very imbalanced given a large sentence length margin, where both very long and very short sentences appear in the document. Another simple approach is to divide the document into regions of fixed length . Such a strategy destroys the syntactic relationship within the document, which may lead to a decline in performance.

To address this issue, we introduced a container division strategy. For implementation, several containers with fixed capacity c were used to load sentences based on the idea that a container should load as many sentences as possible. We successively put the sentences into a container until the text length exceeded capacity c. Then, the sentence sequences in this container is padded to c with zero values. Notably, we set up an overlap sentence between each adjacent container. That is, the last sentence of the previous container and the first sentence of the next container are the same. This can effectively link these two containers such that they are not independent of each other and the syntactic dependencies are preserved.

To extract the local features in each container, the pretrained language model BERT [16] was used. It achieved impressive performance for various NLP tasks. BERT consists of multiple layers of bidirectional transformer encoders [19] and is pretrained by unsupervised learning with either a masked language model (with a masked ratio of 15%) or next sentence prediction. In particular, the uncased BERT-based[1] model was used, which contains 12 layers of transforms with a hidden size of 768. In each container, two special symbols, i.e., [CLS] and [SEP], were first added to the beginning and end of container r_i, respectively. Then, we fed each container into a BERT model to obtain the local representation

[1] https://github.com/huggingface/transformers.

$t_i \in \mathbb{R}^{d_t}$, denoted as follows:

$$t_i = f_{\text{BERT}}([x_{i1}, x_{i2}, \ldots, x_{ic}]; \theta_{\text{BERT}}) \tag{1}$$

where $x_{i1}, x_{i2}, \ldots, x_{ic}$ is the input sequence of tokens in the ith container; θ_{BERT} is the trainable parameter of the BERT model, which is shared for all containers and then fine-tuned during model training; and d_t=768 is the dimensionality of the local representation.

3.2. Global encoder

To sequentially encode all local representations into a document representation, an attention-based gated memory network was used to capture long-range dependencies across containers and model the global information. Taking all local representations $[t_1, t_2, \ldots, t_v]$ as input, a bidirectional LSTM (BiLSTM) network was first used to map the local representations to hidden states, denoted as

$$\begin{aligned} \overrightarrow{h}_i &= \text{LSTM}(\overrightarrow{h}_{i-1}, t_i) \\ \overleftarrow{h}_i &= \text{LSTM}(\overleftarrow{h}_{i+1}, t_i) \\ h_i &= [\overrightarrow{h}_i; \overleftarrow{h}_i] \end{aligned} \tag{2}$$

where $\overrightarrow{h}_i \in \mathbb{R}^{d_h}$ and $\overleftarrow{h}_i \in \mathbb{R}^{d_h}$ are the forward and backward hidden states of the local representation t_i, respectively, d_h is the dimensionality of the hidden states, and [;] is a concatenate operation. To enhance the information of each local representation, we add residual connections between the input and output of BiLSTM, denoted as

$$k_i = h_i \oplus t_i \tag{3}$$

where k_i is a local representation and $\oplus$ is an elementwise sum operation.

Considering that not all words and sentences contribute equally to the final classification, we used an AGM layer to sequentially select the important information and integrate it into a memory to obtain long-distance relationships and preserve contextual information. To automatically select the important information, we used self-attention mechanism to learn the weights for each local representation k_i, which is calculated as

$$e_i = \tanh(W_e k_i + b_e) \tag{4}$$

$$\alpha_i = \frac{\exp(e_i)}{\sum_{j=1}^{|D|} \exp(e_j)} \tag{5}$$

where W_e and b_e denote the weight and bias associated with the attention layer, respectively, and α_i is the weight assigned to the ith local representation.

The AGM layer has recurrent architecture, which is somewhat similar to a memory network. It uses a memory vector, $m \in \mathbb{R}^{d_m}$, to record the important information in the local representations and a gate g to control the information that is relevant to keep or to forget during training. The calculation of the AGM layer is as follows:

$$g_i = \sigma(W_g k_i + U_g m_{i-1}) \tag{6}$$

$$\hat{m}_i = \tanh(W_h k_i + g_i \odot (U_h m_{i-1})) \tag{7}$$

$$m_i = (1 - \alpha_i) \odot m_{i-1} + \alpha_i \odot \hat{m}_i \tag{8}$$

where W_g, U_g, W_h and U_h are the trainable parameters with the AGM layer, σ and tanh are the sigmoid and tangent activation functions, respectively, and m_{i-1} and $\hat{m}_i$ denote the previous state and the current candidate state, respectively. Gate g_i determines how much past information needs to be forgotten, while weight α_i is used to control how much past information is retained and how much new information is absorbed.

For implementation, we applied two independent AGM layers to model both forward and backward information about the sequence at every step. The final memory vector is a concatenation of both directions of AGMs, which is able to preserve information from both the past and future. We concatenated the last memory vector of the forward direction and the first memory vector of the backward direction to generate the document representation o for the final classification,

$$o = [\overrightarrow{m}_v; \overleftarrow{m}_1] \tag{9}$$

Given the training dataset $\{D_n, y_n\}_{n=1}^{N}$, the classification is a one-layer MLP with a *softmax* function. The loss function has categorical cross-entropy, defined as

$$\hat{y}_n^c = softmax(\text{MLP}(o_n)) \tag{10}$$

$$\mathcal{L} = -\sum_{n=1}^{N} \mathbb{I}(y_n) \circ \log(\hat{y}_n^c) \tag{11}$$

where y_n and $\hat{y}_n^c$ denote the gold label and the predicted probability of sample n, respectively, and N and C are the number of training samples and classes, respectively. $\mathbb{I}(y)$ denotes a one-hot vector with the yth component being one, and $\circ$ represents the elementwise multiplication operation.

3.3. Adaptive fine-tuning strategy

As mentioned above, fine-tuning all layers in BERT does not necessarily result in the best performance. Therefore, we proposed using a policy network to adaptively select the layers in BERT that should be fine-tuned or frozen during training. The BERT model in Eq. (1) consists of L-layers of T_i transformers, denoted as

$$f_{\text{BERT}} = [T_1, T_2, \ldots, T_L] \tag{12}$$

Taking container r as input, the policy network learns a unique adaptive fine-tuning strategy per the training example, which is a sequence of policies, denoted by

$$\text{Policy}(r) = \{s_1, s_2, \ldots, s_L\} \tag{13}$$

where $s_i \in \{0, 1\}$ is the policy for the ith layer of the BERT-based model. The fine-tuning strategy can be considered as binary classification task. That is, the ith layer is fine-tuned if s_i is equal to 1 and frozen if 0. For implementation, we freeze the original layer T_i in BERT and create a new trainable layer, $\hat{T}_i$, which is cloned from T_i. With the additional layer $\hat{T}_i$, the output of the ith layer is computed as follows:

$$d_i = s_i \hat{T}_i(d_{i-1}) + (1 - s_i) T_i(d_{i-1}) \tag{14}$$

where d_i is the output representation of the ith BERT layer. The architecture of the policy network is a neural network model with three-layer transformers and a dense output layer using *softmax* activation. Taking the embeddings of container r as input, the policy network is computed as

$$e_i = \text{Transformer}(r) \tag{15}$$

where e_i is a two-element vector. The execution strategy s_i for the ith layer is calculated by

$$s_i = \arg\max(e_i) \tag{16}$$

where argmax is a process of max-sampling e_i by the maximum value. s_i = 0 refers to the ith layer should be frozen, s_i = 1

Table 1
Statistics of AAPD/Reters/IMDB and Yelp-2013 datasets.

Dataset	Train	Dev	Test	Max/doc	Average/doc	Classes
AAPD	53840	1000	1000	598	167.3	54
Reuters	5827	1943	3019	1910	144.3	90
IMDB	108670	13432	13567	1988	393.8	10
Yelp-2013	62522	7773	8671	1643	212.2	5

represents the ith that should be fine-tuned, and e_i is the logit distribution corresponding to the output of the policy network. Unfortunately, parameter s_i is discrete and therefore the network is nondifferentiable and difficult to optimize using backpropagation. In this paper, the Gumbel-Softmax [48] is used to address this issue.

Gumbel-Softmax is a method of sampling from a discrete distribution in a form that allows a differentiable, approximate sampling of the discrete distribution to be defined. Gumbel-Softmax converts discrete values into continuous values, thus enabling the computation of gradients and the optimization of the policy network when backpropagating in an end-to-end fashion. The specific Gumbel-Softmax implementation is as follows:

$$s_i = softmax(\frac{\log(e_i) + G_i}{\tau}) \tag{17}$$

$$G_i = -log\ (-log\ (u_i)) \tag{18}$$

where G_i is a random variable with an independent homogeneous standard Gumbel distribution, which is called Gumbel noise, and the noise is used to make the obtained result invariant, and $u_i \sim U\ (0, 1)$ is sampled from a uniform distribution. The smaller that parameter τ is in this equation, the closer the one-hot vector s_i is. Then the sample values of e_i can be derived from s_i.

Gumbel-Softmax sampling is used to optimize the policy network for backpropagation. The policy network and BERT are jointly trained to find the optimal combination of dynamic fine-tuning corresponding to each input text to improve the classification performance of the model.

4. Experiments

In this section, comparative experiments were conducted to evaluate the performance of hierarchical BERT with an adaptive fine-tuning strategy compared to several existing methods used for document classification.

4.1. Datasets

To evaluate the performance of our method, we conducted experiments on four document classification datasets, including AAPD, Reuters, IMDB and Yelp-2013. The statistics are shown in Table 1. **Train**, **Dev** and **Test** denote the train set, the development set and the test set, respectively. **Max/doc** and **Average/doc** denote the maximum number and the average number of words per document, respectively. **Classes** indicates the number of categories of the document.

- **AAPD** [4] is a multilabel document classification dataset. It consists of the abstracts of papers in the field of computer science and one or more corresponding discipline categories. The total number of disciplines is 54.
- **Reuters** [1] is a news dataset classified by news topics collected by Reuters. Each news article contains one or more topic categories, and the dataset contains 90 categories.
- **IMDB** [20] is a dataset of movie reviews obtained from the IMDB movie rating website, containing scores from 1 to 10.
- **Yelp-2013** [49] consists of several reviews from Yelp, with each review rated from 1 to 5 (higher ratings are better).

4.2. Evaluation metrics

The evaluation metrics in previous studies [4,20] were used to facilitate fair comparison with baseline models. That is, Accuracy (Acc) score was used for the multi categorization task, i.e., IMDB and Yelp-2013, while F_1-score was used for the multilabel task, i.e., AAPD and Reuters. Correspondingly, a higher accuracy or micro F_1-score indicates better prediction performance of the model. Further, z-test [50,51] was applied to determine whether the performance difference was statistically significant.

4.3. Baseline

The proposed HAdaBERT model is compared with several existing methods, including conventional, hierarchical and pre-trained neural networks. The implementation details of each method are described as follows:

- **CNNs** [6] **and LSTM** [29] **:** CNNs capture local n-gram features but lose long-distance dependency. LSTM networks introduce a gating mechanism to solve the problem of RNN gradient explosion.
- **BiLSTM** [52] **:** A bidirectional strategy is adopted for LSTM networks to learn both forward and backward hidden states to capture past and future information simultaneously.
- **ATT-LSTM** [12] **:** Considering that each token in the document contributes differently to the classification, the attention mechanism is introduced to further improve the prediction performance of the BiLSTM model.
- **SGM** [4] **:** An SGM model adopts sequence-to-sequence generation models for multilabel classification of documents to model the association information between labels.
- **NSC** [53] **:** An LSTM network is applied in a hierarchical architecture at both the sentence and document levels to learn document representations for the final classification.
- **HAN** [15] **:** The HAN model combines bidirectional LSTM networks with an attention mechanism at both the sentence and document levels simultaneously for document classification.
- **DocBERT** [20] **:** DocBERT truncates the document to the maximum length limitation and then fine-tune BERT for document classification.
- **RoBERTa** [18] **:** RoBERTa is a more fine-grained tuned version of the BERT model. More datasets and improved optimization functions are used in RoBERTa. For the part that exceeds the maximum length limitation, RoBERTa takes a similar truncation approach.
- **ALBERT** [17] **:** Both word embedding parameter factorization and hidden interlayer parameter sharing are used in ALBERT. The same truncation process is used to fine-tune ALBERT.
- **HAdaBERT :** The proposed hierarchical BERT model is also implemented for comparison. An adaptive fine-tuning strategy is used to further improve the model performance.

The training procedure of the HAdaBERT model is divided into two separate parts: local and global encoders. We used the uncased BERT-based model for the local encoder. The optimizer is Adam, which applies a warmup strategy with a weight decay of 0.01 during training. For the local encoder, the learning rates of BERT and the policy network are 2e−5 and 2e−4, respectively. The batch size is 32. For the global encoder, the learning rate is 2e−4. The batch size is 32 in IMDB and Yelp-2013 and 128 in AAPD and Reuters. The output dimensionality (d_t) of the hidden representation in the PLMs is 768.

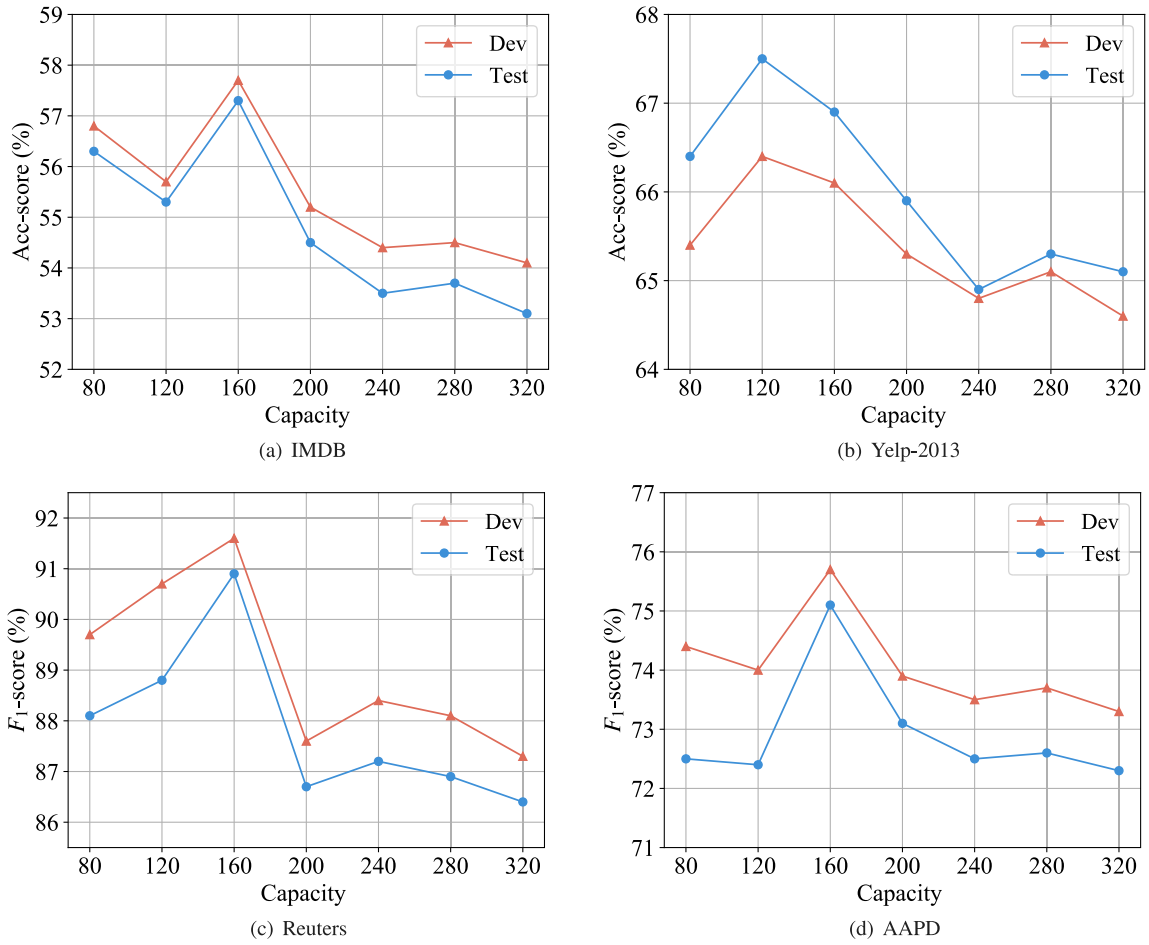

Fig. 3. Effect of the capacity of the local encoder on different datasets.

Table 2
Experimental results comparison of various methods on the document-level classification task using IMDB, Yelp-2013, AAPD and Reuters datasets (%). The best results are shown in bold.

Models		IMDB		Yelp-2013		AAPD		Reuters	
		Dev.Acc	Test.Acc	Dev.Acc	Test.Acc	Dev.F1	Test.F1	Dev.F1	Test.F1
Conventional Neural Networks	CNN	42.9	42.7	56.3	57.7	54.5	51.4	83.5	80.8
	LSTM	48.4	48.0	53.1	53.9	69.8	68.8	84.8	82.7
	BiLSTM	49.3	48.9	57.6	58.4	70.3	69.1	86.9	84.7
	ATT-LSTM	49.7	49.3	60.8	62.4	71.0	69.5	87.4	85.1
	SGM	–	–	–	–	–	71.0	82.5	78.8
Hierarchical Neural Networks	NSC	50.9	50.6	61.4	62.7	69.5	67.3	86.7	84.8
	HAN	51.8	51.2	62.7	63.1	70.2	68.0	87.6	85.2
Pretrained Neural Networks	DocBERT	54.4	54.2	65.1	66.3	75.3	73.4	90.5	89.0
	RoBERTa	54.9	54.5	66.2	67.2	75.1	73.2	91.3	90.7
	ALBERT	52.8	52.6	65.3	66.5	74.7	72.8	90.8	90.4
	HAdaBERT	**57.7**	**57.3**	**66.4**	**67.5**	**75.8**	**75.1**	**91.6**	**90.9**

HAdaBERT vs. DocBERT differ significantly ($p < 0.05$).

4.4. Hyperparameter fine-tuning

The different capacities (c) of the local encoders affect the final performance of the HAdaBERT model on the IMDB, Yelp-2013, AAPD and Reuters datasets. Fig. 3 investigates the optimal parameters of the capacity of the local encoder according to the final performance from 80 to 320. As indicated, the best performance is achieved when the local encoder capacity is 160 for IMDB, Reuters and AAPD and 120 for Yelp-2013. Once the capacity exceeds the optimal settings, either the F_1-score or the accuracy declines. It is suggested that the appropriate capacity of the local encoder can ensure that the model obtains enough local information and improve the classification performance through the integration of the global encoder.

Once the optimal parameter settings were obtained, they were used for classification on the test sets of different corpora. Table 2 also shows the performance of the proposed HAdaBERT on both the development set and the test set of IMDB, Yelp-2013, AAPD

Table 3
Results of ablation study of the proposed HAdaBERT model(%). w/o means to remove some parts from the HAdaBERT model.

Models	IMDB		Yelp-2013		AAPD		Reuters	
	Dev.Acc	Test. Acc	Dev.Acc	Test.Acc	Dev.F1	Test.F1	Dev.F1	Test.F1
DocBERT	54.4	54.2	65.1	66.3	75.3	73.4	90.5	89.0
HAdaBERT	**57.7**	**57.3**	**66.4**	**67.5**	**75.8**	**75.1**	**91.6**	**90.9**
HAdaBERT w/o global encoder	56.6	56.1	65.5	66.4	75.2	74.1	90.8	89.3
HAdaBERT w/o adaptive fine-tuning	57.2	56.4	66.1	66.8	75.5	74.4	91.4	90.4
HAdaBERT w/o local encoder	48.7	47.0	64.2	65.8	75.0	73.3	90.8	89.5

Table 4
Results of adaptive fine-tuning strategy and baseline fine-tuning strategy with hierarchical BERT on IMDB, Yelp-2013, AAPD and Reuters datasets (%).

Strategy	IMDB		Yelp-2013		AAPD		Reuters	
	Dev.Acc	Test. Acc	Dev.Acc	Test.Acc	Dev.F1	Test.F1	Dev.F1	Test.F1
Standard fine-tuning	57.2	56.4	66.1	66.8	75.5	74.4	91.4	90.4
Fine-tuning last 4 layer	56.3	55.9	65.2	66.4	74.6	73.7	88.6	87.9
Fine-tuning skip	56.5	56.2	65.9	66.7	75.0	73.8	88.9	88.2
Fine-tuning first-L layer	55.8	55.6	66.0	67.0	75.1	74.0	89.8	88.5
Fine-tuning random-L layer	55.4	54.7	65.3	66.6	74.2	73.3	90.2	89.5
Adaptive fine-tuning	**57.7**	**57.3**	**66.4**	**67.5**	**75.8**	**75.1**	**91.6**	**90.9**

and Reuters, showing that the performance on the test set was very close to that of the development set for all datasets.

4.5. Comparative results

Table 2 shows the comparative results of the proposed HAdaBERT model against several baselines. For the conventional neural network, BiLSTM achieved a better performance than either the CNN or LSTM model. When the attention mechanism was introduced to the LSTM, the ATT-LSTM model emphasized important information in the document, thus improving the classification performance.

By introducing the hierarchical structure, both the NSC and HAN models outperform the conventional neural networks since they captured the complex syntactic relationships of the documents. HAN provided better results than NSC because of the introduction of an attention mechanism in the hierarchical networks, which assigns relevant importance to different components. Notably, the performance of the hierarchical neural network is slightly lower than that of the traditional neural network for the AAPD dataset. The possible reason is that the hierarchical network is imbalanced because of a large sentence margin in the AAPD dataset. Since both a very long and very short sentence are assigned to a single local encoder, the key component features in the long sentence are more difficult to extract.

The pretrained language models (DocBERT, RoBERTa, and ALBERT) are far better than the previous baseline models due to the high quality of the contextual word representation contributed by the pretrained language models. Further, the proposed HAdaBERT outperformed DocBERT with a statistically significant performance difference ($p < 0.05$). For instance, the proposed HAdaBERT outperformed DocBERT by 3.1%, 1.2%, 1.7%, and 1.9% on the test sets of IMDB, Yelp-2013, AAPD and Reuters, respectively. Since the previous PLMs apply a truncation strategy, some information may be lost when the important keywords are in the truncated parts. The proposed HAdaBERT model achieved the best performance for all datasets since it captures high-quality local and global information and is adaptively fine-tuned to improve performance. Another observation is that the proposed HAdaBERT model shows the greatest improvement for the IMDB (longest document sequence length) dataset, which clearly demonstrates the effectiveness of the proposed model to learn document representation of an appreciable length.

4.6. Ablation study

Table 3 shows the results of the ablation experiments performed with the proposed model to further assess the effectiveness of each component (i.e., the local encoder, global encoder, and adaptive fine-tuning strategy). We successively removed a component to investigate whether its absence reduced the performance of the proposed HAdaBERT model. As indicated, the various ablation models produced varying degrees of performance decline, indicating that each component plays an indispensable role in performance improvements.

HAdaBERT w/o global encoder replaces the AGM layer with an average pooling layer as the global encoder. It only achieves performances 56.1%, 66.4%, 74.1% and 89.3% of the original model with the IMDB, Yelp-2013, AAPD, and Reuters datasets, respectively. However, it outperformed DocBERT, especially with the IMDB dataset, with a 1.9% improvement in the accuracy score. These results demonstrate the effectiveness of the proposed global encoder.

The removal of the adaptive fine-tuning (i.e., HAdaBERT w/o adaptive fine-tuning) leads to a decline in performance for all datasets compared to HAdaBERT. The possible reason is that the adaptive fine-tuning strategy customizes a specific fine-tuning strategy for each input sample to avoid the risk of overfitting. By dynamically selecting the layer that should be fine-tuned for each sample, it can improve the performance of the classification. HAdaBERT w/o local encoder applied a traditional fixed-length overlap as local encoder instead of the proposed containers, which led to a lower performance than HAdaBERT, indicating the effectiveness of the container as a local encoder.

4.7. The effect of adaptive fine-tuning strategy

To investigate the effect of the adaptive fine-tuning strategy, we compared the adaptive fine-tuning strategy against several fine-tuning methods, and the results are shown in Table 4. As indicated, by only fine-tuning the first-L layer, the last 4 layers always achieved similar performances for all datasets. Here, L represents the average fine-tuning layers of the adaptive fine-tuning strategy for the different datasets in Fig. 5. The skip strategy means that only odd layers are used, while even layers are fixed. These three strategies achieved lower performances than the standard strategy in which all layers are fine-tuned, indicating

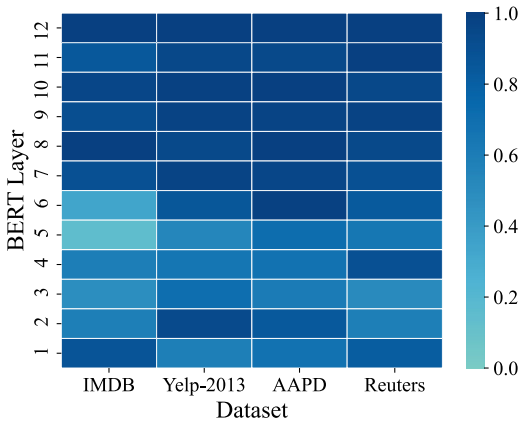

Fig. 4. Visualization of policies on IMDB, Yelp-2013, AAPD and Reuters.

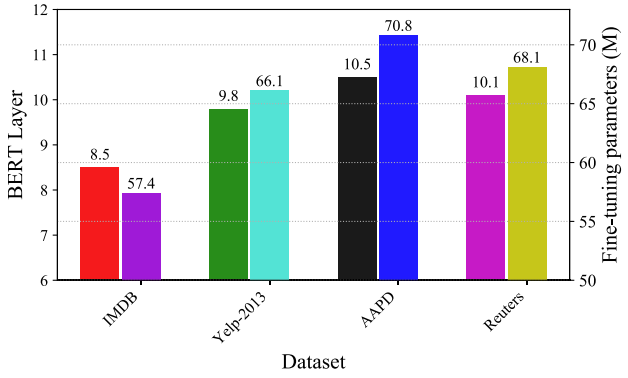

Fig. 5. Average number of layers and parameters that were fine-tuned by using the adaptive fine-tuning strategy on IMDB, Yelp-2013, AAPD and Reuters.

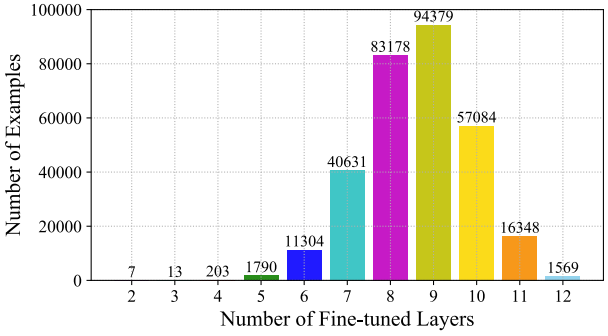

Fig. 6. Statistics of the number of layers that are fine-tuned on IMDB dataset.

that a fixed fine-tuning strategy does not lead to a good result. However, randomly selecting layers for fine-tuning causes the model to fail to converge, thus also causing lower performance. In addition, the proposed adaptive fine-tuning strategy always obtains the best results compared to the other fine-tuning strategies.

Fig. 4 shows the possibility that a layer is fine-tuned in the different datasets by using the proposed HAdaBERT model. A deeper color means that the layer is more likely to be selected for fine-tuning in the training phase. As indicated, the upper 5 layers were always selected for fine-tuning for all datasets, which also confirms the similar phenomenon reported in [21]. The high-level layer can encode rich semantic and syntactic information and is more suitable for downstream tasks. Similarly, for different input samples, it is necessary to selectively fine-tune the low-level layers to obtain features at the phrase or word level. Notably, it is not necessary to fine-tune all layers.

To better understand the improvement of adaptive fine-tuning, Fig. 5 shows the average number of layers and parameters that are fine-tuned for all datasets. The commonly used 12-layer BERT-based model contains 108M parameters. By using an adaptive fine-tuning strategy, only 53.5%, 61.2%, 65.6% and 63.1% of the total parameters and approximately 9.5 layers need to be trained with IMDB, Yelp-2013, AAPD and Reuters datasets, respectively. Even though fewer layers and parameters should be fine-tuned, the model achieved good performance. This confirms the effectiveness of the proposed adaptive fine-tuning strategy.

In addition to the visualization of the fine-tuned samples for each layer, we also determined how many layers were specifically fine-tuned for samples in the IMDB dataset, and the results are shown in Fig. 6. Most samples tend to fine-tune 8–9 layers instead of all 12 layers. More surprisingly, some samples needed only to fine-tune 6 or fewer layers. This proves that a small sample only requires that a few layers be fine-tuned. On the other hand, adaptively selecting a layer for fine-tuning and fixing others help these simple samples share pretrained parameters to reduce the risk of overfitting, thereby improving the performance of the model.

4.8. Case study

To illustrate the effectiveness of HAdaBERT in encoding document representations, we further analyzed several specific cases in the IMDB dataset. Table 5 shows the polarity predictions made by the different models, including HAdaBERT, DocBERT and ATT-LSTM. The bold words in each example play important roles in the polarity classification of the document. The previous part of example 1 showed that words in italics related negative opinions, *cheapish plot* and *didn't impress* towards the device, but the overall feelings about the movie were stated as *loved,moved* and *impressed*. DocBERT has made an incorrect classification because the truncation strategy resulted in the loss of important information. Similarly, ATT-LSTM also made an incorrect prediction. The rational reason is that when ATT-LSTM models the semantics of documents, the long-distance dilutes the importance of various constituent words. Therefore, it is difficult to apply a self-attention mechanism to learn both syntactic relationships within the sentences and can lead to misclassification. Even though *cheapish* carries negative connotations about a device, it may be less important in the global context. In contrast, the HAdaBERT model emphasized *loved* and *impressed* with respect to the film. It shows that the global encoder captures the syntactic dependence between different local encoders and learns the contribution of each container to the document classification.

The beginning of example 2 described only the background and the main content of the movie, showing no subjective polarity. While the DocBERT truncated *recommended* and *higher rating* which carry positive connotations, and therefore, the model made an incorrect classification. Both examples prove that the proposed HAdaBERT model has a great ability to capture local information and syntactic relationships between different local encoders; thus, it classified both examples correctly.

5. Conclusion

In this paper, a hierarchical BERT with an adaptive fine-tuning strategy was proposed for document classification. It consists of two parts, including both the local encoder and global encoder, which can effectively capture both the local and global information of the document. To address the limitations of existing fine-tuning strategies, an adaptive fine-tuning strategy was proposed

Table 5
Analysis of typical sample cases.

Example	Text	Method	Prediction	Label	Correct
1	...you're familiar with other X-men movies...*cheapish plot* devices... Jennifer Lawrence (whose role is pivotal) *didn't impress* me in First Class...I **loved** this movie and was genuinely **impressed** and **moved**...	HAdaBERT	9	9	✓
		DocBERT	2	9	×
		ATT-LSTM	4	9	×
2	...This is her husband, with whom she has been through a lot, who is over taken by troubles,not from his own doing…**Recommended** highly.This one is should have a **higher rating** by the movie critics.	HAdaBERT	10	10	✓
		DocBERT	4	10	×
		ATT-LSTM	6	10	×

to customize a specific fine-tuning strategy for each input sample and dynamically select the layer of fine-tuning required by the target sample. The experimental results showed that the proposed method achieved better performance than previous methods, including conventional, hierarchical and pretrained neural networks for document classification.

Future work will explore a more efficient local encoder and introduce reinforcement learning to investigate a better fine-tuning strategy.

CRediT authorship contribution statement

Jun Kong: Investigation, Methodology, Software, Formal analysis, Validation, Writing – original draft. **Jin Wang:** Conceptualization, Software, Formal analysis, Resource and Writing – review & editing, Resources and Funding acquisition. **Xuejie Zhang:** Project administration, Resources, Supervision and Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (NSFC) under Grants Nos. 61702443, 61966038 and 61762091. The authors would like to thank the anonymous reviewers for their constructive comments.

References

[1] C. Apté, F. Damerau, S.M. Weiss, Automated learning of decision rules for text categorization, ACM Trans. Inf. Syst. (TOIS) 12 (3) (1994) 233–251, http://dx.doi.org/10.1145/183422.183423.

[2] F. Liu, J. Zheng, L. Zheng, C. Chen, Combining attention-based bidirectional gated recurrent neural network and two-dimensional convolutional neural network for document-level sentiment classification, Neurocomputing 371 (2020) 39–50.

[3] J. Wang, L.C. Yu, K.R. Lai, X. Zhang, Dimensional sentiment analysis using a regional CNN-LSTM model, in: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL-2016, 2016, pp. 225–230, http://dx.doi.org/10.18653/v1/p16-2037.

[4] P. Yang, X. Sun, W. Li, S. Ma, W. Wu, H. Wang, SGM: Sequence generation model for multi-label classification, in: Proceedings of the 27th International Conference on Computational Linguistics, 2018, pp. 3915–3926, URL: http://arxiv.org/abs/1806.04822.

[5] K.M. Hermann, T. Kočiský, E. Grefenstette, L. Espeholt, L. Espeholt, W. Kay, M. Suleyman, P. Blunsom, Teaching machines to read and comprehend, in: Proceedings of the 28th International Conference on Neural Information Processing Systems, NIPS-2015, 2015, pp. 1693–1701.

[6] Y. Kim, Convolutional neural networks for sentence classification, in: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP, 2014, pp. 1746–1751, http://dx.doi.org/10.1109/CLEI.2017.8226381.

[7] X. Zhang, Y. LeCun, Character-level convolutional networks for text classification, in: Proceedings of Advances in Neural Information Processing Systems, NIPS-2015, 2015, pp. 649–657, URL: http://arxiv.org/abs/1502.01710.

[8] O. Irsoy, C. Cardie, Opinion mining with deep recurrent neural networks, in: Proceedings of the Conference on Empirical Methods in Natural Language Processing, EMNLP-2014, 2014, pp. 720–728, http://dx.doi.org/10.3115/v1/d14-1080.

[9] D. Yogatama, C. Dyer, W. Ling, P. Blunsom, Generative and discriminative text classification with recurrent neural networks, 2017, arXiv preprint arXiv:1703.01898, URL: http://arxiv.org/abs/1703.01898.

[10] J. Chung, C. Gulcehre, K. Cho, Y. Bengio, Empirical evaluation of gated recurrent neural networks on sequence modeling, 2014, arXiv preprint arXiv:1412.3555, URL: http://arxiv.org/abs/1412.3555.

[11] K.S. Tai, R. Socher, C.D. Manning, Improved semantic representations from tree-structured long short-term memory networks, in: Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing, ACL-IJCNLP 2015, 2015, pp. 1556–1566, http://dx.doi.org/10.3115/v1/p15-1150.

[12] D. Bahdanau, K.H. Cho, Y. Bengio, Neural machine translation by jointly learning to align and translate, 2014, arXiv preprint arXiv:1409.0473.

[13] S. Sabour, N. Frosst, G.E. Hinton, Dynamic routing between capsules, in: Proceedings of Advances in Neural Information Processing Systems, NIPS-2017, 2017, pp. 3857–3867.

[14] J. Gong, X. Qiu, S. Wang, X. Huang, Information aggregation via dynamic routing for sequence encoding, in: Proceedings of the 27th International Conference on Computational Linguistics, 2018, pp. 2742–2752.

[15] Z. Yang, D. Yang, C. Dyer, X. He, A. Smola, E. Hovy, Hierarchical attention networks for document classification, in: Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2016, 2016, pp. 1480–1489.

[16] J. Devlin, M.W. Chang, K. Lee, K. Toutanova, BERT: Pre-training of deep bidirectional transformers for language understanding, in: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, 2019, pp. 4171–4186.

[17] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, R. Soricut, Albert: A lite bert for self-supervised learning of language representations, 2019, arXiv preprint arXiv:1909.11942.

[18] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, V. Stoyanov, RoBERTa: A robustly optimized BERT pretraining approach, 2019, arXiv preprint arXiv:1907.11692.

[19] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, Ł. Kaiser, I. Polosukhin, Attention is all you need, in: Proceedings of Advances in Neural Information Processing Systems, NIPS-2017, 2017, pp. 5998–6008, URL: http://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf.

[20] A. Adhikari, A. Ram, R. Tang, J. Lin, DocBERT: BERT for document classification, 2019, arXiv preprint arXiv:1904.08398.

[21] G. Jawahar, B. Sagot, D. Seddah, What does BERT learn about the structure of language? in: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, ACL-2019, 2019, pp. 3651–3657, http://dx.doi.org/10.18653/v1/p19-1356.

[22] C. Sun, X. Qiu, Y. Xu, X. Huang, How to fine-tune bert for text classification? in: China National Conference on Chinese Computational Linguistics, 2019, pp. 194–206.

[23] R. Johnson, Deep pyramid convolutional neural networks for text categorization, in: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL-2017, 2017, pp. 562–570.

[24] A. Conneau, H. Schwenk, Y.L. Cun, L. Barrault, Very deep convolutional networks for text classification, in: Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL-2017, 2017, pp. 1107–1116, http://dx.doi.org/10.18653/v1/e17-1104.

[25] N. Kalchbrenner, E. Grefenstette, P. Blunsom, A convolutional neural network for modelling sentences, in: Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL-2014, 2014, pp. 655–665, http://dx.doi.org/10.3115/v1/p14-1062.

[26] J.L. Elman, Finding structure in time, Cogn. Sci. 14 (1990) 179–211, http://dx.doi.org/10.1016/0364-0213(90)90002-E.

[27] J. Liu, W.C. Chang, Y. Wu, Y. Yang, Deep learning for extreme multi-label text classification, in: Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR-2017, 2017, pp. 115–124, http://dx.doi.org/10.1145/3077136.3080834.
[28] A. Joulin, E. Grave, P. Bojanowski, T. Mikolov, Bag of tricks for efficient text classification, in: Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL-2017, 2017, pp. 427–431, http://dx.doi.org/10.18653/v1/e17-2068.
[29] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Comput. 9 (8) (1997) 1735–1780, http://dx.doi.org/10.1162/neco.1997.9.8.1735.
[30] P. Zhou, Z. Qi, S. Zheng, J. Xu, H. Bao, B. Xu, Text classification improved by integrating bidirectional LSTM with two-dimensional max pooling, in: Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics, COLING-2016, 2016, pp. 3485–3495.
[31] S. Sukhbaatar, A. Szlam, J. Weston, R. Fergus, End-to-end memory networks, in: Proceedings of Advances in Neural Information Processing Systems, NIPS-2015, 2015, pp. 2440–2448.
[32] K. Tran, A. Bisazza, C. Monz, Recurrent memory networks for language modeling, in: Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics, NAACL-HLT 2016, 2016, pp. 321–331, http://dx.doi.org/10.18653/v1/n16-1036.
[33] Z.y. Dou, Capturing user and product information for document level sentiment analysis with deep memory network, in: Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP-2017, 2017, pp. 521–526.
[34] K. Xu, J.L. Ba, R. Kiros, A. Courville, Show, attend and tell: Neural image caption generation with visual attention, in: Proceedings of the 32th International Conference on Machine Learning, 2015, pp. 2048–2057.
[35] M.t. Luong, C.D. Manning, Effective approaches to attention-based neural machine translation, in: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP-2015, 2015, pp. 1412–1421.
[36] S. Gao, A. Ramanathan, G. Tourassi, Hierarchical convolutional attention networks for text classification, in: Proceedings of the 3rd Workshop on Representation Learning for NLP, 2018, pp. 11–23.
[37] J. Zheng, F. Cai, T. Shao, H. Chen, Self-interaction attention mechanism-based text representation for document classification, Appl. Sci. 8 (2018) 613, http://dx.doi.org/10.3390/app8040613.
[38] D. Tang, B. Qin, T. Liu, Document modeling with gated recurrent neural network for sentiment classification, in: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP-2015, 2015, pp. 1422–1432, http://dx.doi.org/10.18653/v1/d15-1167.
[39] J. Xu, D. Chen, X. Qiu, X. Huang, Cached long short-term memory neural networks for document-level sentiment classification, in: Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP-2016, 2016, pp. 1660–1669, http://dx.doi.org/10.18653/v1/d16-1172.
[40] J. Wang, L.C. Yu, K.R. Lai, X. Zhang, Investigating dynamic routing in tree-structured LSTM for sentiment analysis, in: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP, 2019, pp. 3432–3437, http://dx.doi.org/10.18653/v1/D19-1343, URL: https://www.aclweb.org/anthology/D19-1343.
[41] K. Lai, L.-C. Yu, X. Zhang, J. Wang, Tree-structured regional CNN-LSTM model for dimensional sentiment analysis, IEEE/ACM Trans. Audio Speech Lang. Process. 28 (2019) 581–591, http://dx.doi.org/10.1109/TASKP.2019.2959251.
[42] Y. Yin, Y. Song, M. Zhang, Document-level multi-aspect sentiment classification as machine comprehension, in: Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP-2017, 2017, pp. 2044–2054, http://dx.doi.org/10.18653/v1/d17-1217.
[43] J.-B. Remy, A.J.P. Tixier, M. Vazirgiannis, Bidirectional context-aware hierarchical attention network for document understanding, 2019, arXiv preprint arXiv:1908.06006, URL: http://arxiv.org/abs/1908.06006.
[44] J. Ive, G. Gkotsis, R. Dutta, R. Stewart, S. Velupillai, Hierarchical neural model with attention mechanisms for the classification of social media text related to mental health, in: Proceedings of the Fifth Workshop on Computational Linguistics and Clinical Psychology: From Keyboard To Clinic, 2018, pp. 69–77, http://dx.doi.org/10.18653/v1/w18-0607.
[45] G. Rao, W. Huang, Z. Feng, Q. Cong, LSTM With sentence representations for document-level sentiment classification, Neurocomputing 308 (2018) 49–57, http://dx.doi.org/10.1016/j.neucom.2018.04.045.
[46] C. Gong, K. Shi, Zhendong Niu, Hierarchical text-label integrated attention network for document classification, in: Proceedings of the 2019 3rd High Performance Computing and Cluster Technologies Conference, HPCCT-2019, 2019, pp. 254–260.
[47] M.E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, L. Zettlemoyer, Deep contextualized word representations, in: Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics, 2018, pp. 2227–2237.
[48] H. Catherine, M.L. Cook, E. Mckone, Categorical reparameterization with gumbel-softmax, 2017, arXiv preprint arXiv:1611.01144.
[49] D. Tang, B. Qin, T. Liu, Learning semantic representations of users and products for document level sentiment classification, in: Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing, ACL-IJCNLP 2015, 2015, pp. 1014–1023, http://dx.doi.org/10.3115/v1/p15-1098.
[50] E.R. Isaac, Test of Hypothesis-Concise Formula Summary.
[51] A. Yeh, More accurate tests for the statistical significance of result differences, in: Proceedings of the 18th International Conference on Computational Linguistics, COLING 2000, 2000, URL: https://aclanthology.org/C00-2137.
[52] M. Schuster, K.K. Paliwal, Bidirectional recurrent neural networks, IEEE Trans. Signal Process. 45 (11) (1997) 2673–2681.
[53] H. Chen, M. Sun, C. Tu, Y. Lin, Z. Liu, Neural sentiment classification with user and product attention, in: Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP-2016, 2016, pp. 1650–1659, http://dx.doi.org/10.18653/v1/d16-1171.

Jun Kong is currently pursuing his Master's Degree in the School of Information Science and Engineering, Yunnan University, China. He received a Bachelor's Degree in Information Engineering from Southwest Forestry University, China. His research interests include natural language processing, text mining, and machine learning.

Jin Wang is an associate professor in the School of Information Science and Engineering, Yunnan University, China. He holds a Ph.D. in Computer Science and Engineering from Yuan Ze University, Taoyuan, Taiwan, and another Ph.D. in Communication and Information Systems from Yunnan University, Kunming, China. His research interests include natural language processing, text mining, and machine learning.

Xuejie Zhang is a professor in the School of Information Science and Engineering, and Director of High-Performance Computing Center, Yunnan University, China. He received his Ph.D. in Computer Science and Engineering from the Chinese University of Hong Kong in 1998. His research interests include high performance computing, cloud computing, and big data analytics.