Neurocomputing 460 (2021) 84–94

Contents lists available at ScienceDirect

Neurocomputing

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

BERT-JAM: Maximizing the utilization of BERT for neural machine translation

Zhebin Zhang [*], Sai Wu, Dawei Jiang, Gang Chen

College of Computer Science and Technology, Zhejiang University, Hangzhou 310007, China

A R T I C L E I N F O

Article history:
Received 6 November 2020
Revised 27 May 2021
Accepted 4 July 2021
Available online 06 July 2021
Communicated by Zidong Wang

Keywords:
Neural network
Machine translation
Deep learning

A B S T R A C T

Pre-training based approaches have been demonstrated effective for a wide range of natural language processing tasks. Leveraging BERT for neural machine translation (NMT), which we refer to as BERT-enhanced NMT, has received increasing interest in recent years. However, there still exists a research gap in studying how to maximize the utilization of BERT for NMT tasks. Firstly, previous studies mostly focus on utilizing BERT's last-layer representation, neglecting the linguistic features encoded by the intermediate layers. Secondly, it requires further architectural exploration to integrate the BERT representation with the NMT encoder/decoder layers efficiently. And thirdly, existing methods keep the BERT parameters fixed during training to avoid the catastrophic forgetting problem, wasting the chances of boosting the performance via fine-tuning. In this paper, we propose BERT-JAM to fill the research gap from three aspects: 1) we equip BERT-JAM with fusion modules for composing BERT's multi-layer representations into a fused representation that can be leveraged by the NMT model, 2) BERT-JAM utilizes joint-attention modules to allow the BERT representation to be dynamically integrated with the encoder/decoder representations, and 3) we train BERT-JAM with a three-phase optimization strategy that progressively unfreezes different components to overcome catastrophic forgetting during fine-tuning. Experimental results show that BERT-JAM achieves state-of-the-art BLEU scores on multiple translation tasks.

1. Introduction

Pre-training has been demonstrated as a highly effective method for boosting the performance of many natural language processing (NLP) tasks, such as question answering, text classification, and so on. By training on massive unlabeled text data, pre-trained language models are able to learn the contextual representations of input tokens, which are extremely helpful for accomplishing downstream tasks. BERT [1], as one of the most widely used pre-trained language models, is trained using two unsupervised tasks, namely, mask language modeling and next sentence prediction. By adding a few layers on top, BERT can be easily adapted into a task-specific model, which is then fine-tuned on the labeled data to achieve optimal performance. Such a practice has been exercised in various NLP scenarios and has achieved many state-of-the-art (SOTA) results.

The study of BERT-enhanced NMT, i.e., leveraging pre-trained BERT for neural machine translation, has received much research interest. However, exploiting BERT for NMT cannot be easily accomplished by adding a few layers atop BERT and then fine-tuning, as in monolingual NLP tasks. The challenge is brought by the complexity of NMT models which typically consist of an encoder that transforms the source language tokens into a hidden representation and a decoder that predicts the target language tokens based on the hidden representation.

Although various methods for integrating NMT with BERT have been proposed, it remains an open question how to maximize the utilization of BERT for NMT. We identify three aspects of the research gap. Firstly, most previous studies on BERT-enhanced NMT only utilize BERT's last-layer representation without making full use of the intermediate layers. It has been shown that BERT's different layers encode different linguistic features [2], which can be fully used to boost the performance of NMT models. The dynamic fusion method proposed by Weng et al. [3], to our knowledge, is the only work on the utilization of BERT's multi-layer representations for NMT. However, their method conditions the weights for combining the multi-layer representations on the instance-specific representations of the encoder layers. Consequently, it cannot be applied to the decoder layers because the ground-truth decoder representations are not available at the

* Corresponding author.
E-mail addresses: zhebinzhang@zju.edu.cn (Z. Zhang), wusai@zju.edu.cn (S. Wu), jiangdw@zju.edu.cn (D. Jiang), cg@zju.edu.cn (G. Chen).

https://doi.org/10.1016/j.neucom.2021.07.002
0925-2312/© 2021 Elsevier B.V. All rights reserved.

inference stage. Hence, how to make the most of BERT's intermediate layers for both the NMT encoder and decoder requires further exploration.

Secondly, there is much more to learn about how to allow the NMT encoder and decoder to utilize the BERT representation in an efficient manner. Various methods have been proposed, e.g., transforming the encoder representation by adding the BERT representation to it [3], using distillation to transfer the knowledge from BERT to the NMT model [4], etc. The recently proposed BERT-fused model [5] employs attention mechanisms to fuse BERT representations with the encoder/decoder layers and achieves state-of-the-art results. Specifically, they introduce a BERT-encoder cross-attention module into each encoder layer and average its output with the self-attention module, as illustrated in Fig. 1(a). However, we find the approach of averaging the outputs unable to fully utilize the BERT representation when more attention should be paid to the BERT representation.

Thirdly, previous works propose to keep the BERT parameters fixed during training to avoid catastrophic forgetting [4,6]. Catastrophic forgetting refers to the problem that the knowledge pretrained by BERT is gradually lost during fine-tuning. Although catastrophic forgetting is a common problem that comes with fine-tuning, it is exacerbated in the field of BERT-enhanced NMT, which can be attributed to the fact that NMT models take much more iterations to converge. However, giving up fine-tuning BERT because of the forgetting problem prevents us from fully optimizing the model.

To fill the research gap, we propose a BERT-enhanced NMT model named BERT-JAM, which stands for **BERT**-fused **J**oint-**A**ttention **M**odel. We seek to maximize the utilization of BERT for NMT in three ways. Firstly, we incorporate a fusion module into each encoder/decoder layer used for composing BERT's multi-layer representations into a fused representation. In contrast to the dynamic fusion method [3], we let the weights for combining the multi-layer representations be shared among different instances. Not only can our fusion method be applied to both the encoder and the decoder, but also it allows each encoder/decoder layer to have a consistent preference for certain BERT layers. Secondly, we propose a joint-attention module used for integrating the fused BERT representation with encoder and decoder layers, as shown in Fig. 1(b). Different from the BERT-fused model which uses separate attention modules (i.e., cross-attention module and self-attention module) whose outputs are averaged, the joint-attention module simultaneously performs the functions of different attention modules and dynamically allocate attention between the BERT representation and the encoder/decoder representation. Thirdly, we train BERT-JAM with a three-phase optimization strategy that progressively unfreezes different parts of the model to overcome the problem of catastrophic forgetting. The strategy allows the model to benefit from the performance boost brought by fine-tuning BERT.

We summarize the contributions of this paper as follows:

- We propose a novel BERT-enhanced NMT model named BERT-JAM which is equipped with fusion modules and joint-attention modules. The fusion module composes BERT's multi-layer representations into a fused representation and allows BERT-JAM to benefit from the linguistic features encoded by BERT's intermediate layers. And the joint-attention module integrates the functions of different attention modules and dynamically allocates attention between different representations.
- We propose a three-phase optimization strategy to train BERT-JAM that allows us to overcome the catastrophic forgetting problem while fine-tuning.
- This is the first work that studies how the performance of BERT-enhanced NMT models varies with the size of BERT. We find out that, with constrained memory usage and latency bound, deeper BERT models (more layers) are better at boosting the translation performance compared with wider ones (larger dimension).
- We evaluate the proposed BERT-JAM model on several widely used translation tasks. Experimental results show that BERT-JAM achieves SOTA scores on multiple benchmarks, demonstrating the effectiveness of our method.

The rest of this paper is organized as follows. In Section 2, we detail our approach in terms of module description, model architecture, and optimization strategy. Section 3 introduces the experimental setups. In Section 4, we conduct comprehensive experiments with BERT-JAM and discuss the results. We give a review of related works in Section 5 and the conclusions are drawn in Section 6.

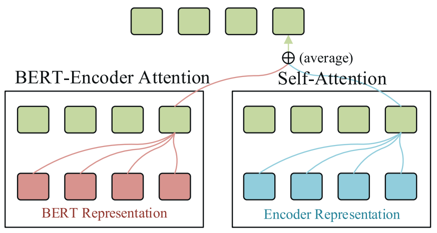
2. Approach

This section presents our proposed approach to boosting machine translation performance with BERT. We begin by introducing the backgrounds of BERT-enhanced NMT. Then, we introduce two building blocks of our model, i.e., the joint-attention module and the fusion module. Next, we elaborate on the architecture of BERT-JAM. Finally, we describe the three-phase optimization strategy used to train the model.

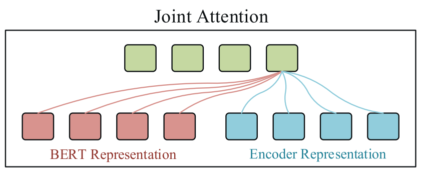
2.1. Backgrounds

2.1.1. Neural machine translation

NMT is modeled as a sequence-to-sequence task that learns the mapping from a sequence of source language tokens $X = (x_1, x_2, \dots, x_n)$ to a sequence of target language tokens $Y = (y_1, y_2, \dots, y_m)$. Existing NMT models are mostly based on the widely adopted encoder-decoder architecture.



(a) The BERT-fused model: separate attention modules.



(b) BERT-JAM: joint-attention module.

Fig. 1. Comparison between using separate attention modules and using the joint-attention module.

On the source side, the encoder transforms the source tokens into a hidden representation as

$$\boldsymbol{E} = (e_1, e_2, \ldots, e_n) = \texttt{Enc}(X). \tag{1}$$

While on the target side, the decoder computes the probabilities of the target tokens based on $\boldsymbol{E}$. Specifically, for the target token y_i, the decoder first takes as input $\boldsymbol{E}$ and the previous tokens $y_{<i} = (y_1, y_2, \ldots, y_{i-1})$ and outputs a hidden representation for y_i given by

$$d_i = \texttt{Dec}(\boldsymbol{E}, y_{<i}). \tag{2}$$

Then linear projection and softmax function are applied on d_i to derive the probability of the token y_i, formally expressed as

$$p_\theta(y_i|\boldsymbol{E}, y_{<i}) = \texttt{softmax}(\texttt{linear}(d_i)), \tag{3}$$

where θ represents the parameters of the model. The training objective is to optimize θ by maximizing the log-likelihood of the entire target sequence, which is defined as

$$\mathcal{L} = \frac{1}{m}\sum_{i=1}^{m} \log p_\theta(y_i|\boldsymbol{E}, y_{<i}). \tag{4}$$

2.1.2. BERT-enhanced NMT

To further boost the performance of machine translation, researchers seek to feed NMT models with extra pre-trained knowledge obtained from massive monolingual corpora beyond the limited bilingual data. To this end, BERT-enhanced NMT models have been proposed to make use of the rich contextual information provided by BERT. A common practice to achieve this goal is to use BERT to encode the source language sentence into the representation $\boldsymbol{B} = \texttt{BERT}(X)$, which can be fed into the NMT model in various ways. The straightforward method is to initialize the encoder with BERT, so that the encoder representation $\boldsymbol{E}$ in Eqs (2)-(4), is replaced by $\boldsymbol{B}$. Another method uses BERT as the embedding layer of the encoder so that the encoder representation becomes $\boldsymbol{E} = \texttt{Enc}(\boldsymbol{B})$. Recent studies approach this problem differently. Yang et al. [4] used a distillation method to transfer the knowledge from BERT to the encoder. And Zhu et al. [5] proposed a BERT-fused model that uses attention mechanisms to fuse Transformer's encoder and decoder layers with BERT representations.

2.2. Joint attention

The attention mechanism proposed by Vaswani et al. [7] operates on a set of query vectors $\{q_1, q_2, \ldots, q_n\}$, a set of key vectors $\{k_1, k_2, \ldots, k_m\}$, and a set of value vectors $\{v_1, v_2, \ldots, v_m\}$, which are all embedded in the same dimension d_{model}. The query, key and value vectors can be packed into matrices denoted by Q, K, and V respectively, where $Q \in \mathbb{R}^{n* \, d_{\text{model}}}$ and $K, V \in \mathbb{R}^{m* \, d_{\text{model}}}$. The attention operation maps the three matrices to an output matrix as

$$\texttt{Attn}(Q, K, V) = \texttt{softmax}(\frac{QK^T}{\sqrt{d_{\text{model}}}})V. \tag{5}$$

Two kinds of modules can be built upon the attention operation, i.e., self-attention module and cross-attention module. For the self-attention module, the matrices Q, K, and V are all projected from the same input matrix, denoted by R, using three different parameter matrices W^Q, W^K, and W^V. So the self-attention module is formulated as

$$\texttt{Attn}_{\texttt{self}}(R) = \texttt{Attn}(W^Q R, W^K R, W^V R). \tag{6}$$

For the cross-attention module, Q is projected from the input matrix R while K and V are projected from another input matrix S. And the output is given by

$$\texttt{Attn}_{\texttt{cross}}(R, S) = \texttt{Attn}(W^Q R, W^K S, W^V S). \tag{7}$$

In BERT-JAM, we propose to combine the self-attention module and cross-attention module using a single joint-attention module. Intuitively, the joint-attention module allows the input matrix R to simultaneously attend to itself and another input matrix S. We formulate the joint-attention module as

$$\texttt{Attn}_{\texttt{joint}}(R, S) = \texttt{Attn}(W^Q R, W_1^K R \| W_2^K S, W_1^V R \| W_2^V S), \tag{8}$$

where W^Q, W_1^K, W_2^K, W_1^V, and W_2^V are all parameter matrices and $\|$ is the concatenation operator.

BERT-JAM utilizes joint attention in different ways for the encoder and the decoder layers. In each encoder layer, we employ a joint-attention module, named BERT-encoder joint attention, to simultaneously attend to the BERT representation and the encoder representation. And each decoder layer incorporates two joint-attention modules. The first one, called BERT-decoder joint attention, attends to the concatenation of the BERT representation and the decoder representation. The second one, termed encoder-decoder joint attention, attends to both the encoder representation and the decoder representation. The formulation of these modules will be given in Section 2.4.

2.3. Fusion of BERT layers

Although most BERT-based works focus on utilizing the last-layer output of BERT for downstream tasks, previous studies have shown that BERT's different layers encode different linguistic features [2]. Few existing studies have explored leveraging BERT's intermediate layers for NMT tasks except the dynamic fusion mechanism proposed by Weng et al. [3]. Their method takes a weighted sum of the BERT's multi-layer representations and the weight assigned to each BERT layer is determined by the product of the BERT representation and encoder's representation. As a result, the dynamic fusion mechanism cannot be used for the decoder because the ground-truth decoder representation is not available at the inference stage.

We approach this problem differently. Instead of conditioning the weights assigned to the BERT layers on the representation of the encoder/decoder, we propose that the weights should be independently trained and be shared across different instances. This approach endows BERT-JAM with two advantages. Firstly, decoupling the calculation of the weights from the representation of encoder/decoder allows the decoder to benefit from utilizing BERT's multi-layer representations, as the encoder does. Secondly, sharing the weights across different instances allows each encoder/decoder layer to have a consistent preference for certain BERT layers that carry different linguistic features.

Specifically, we incorporate a fusion module into each encoder/decoder layer to compute a fused representation of BERT. Denoting by $\vec{\boldsymbol{B}} = (\boldsymbol{B}_1, \boldsymbol{B}_2, \ldots, \boldsymbol{B}_{L_B})$ the multi-layer representations of BERT where L_B is the layer number, the fusion module works in a similar way to the gated linear unit [8]. The fusion module first performs a linear combination of BERT representations and then uses a computed gate g to control the element-wise information flowing from BERT to the encoder/decoder. With a slight abuse of notation, we denote by $\hat{\boldsymbol{B}}$ the fused representation, formally given by

$$\begin{aligned} \hat{\boldsymbol{B}} &= \texttt{Fuse}(\vec{\boldsymbol{B}}) = g \otimes \left(\sum_{i=1}^{L_B} \alpha_i \boldsymbol{B}_i\right), \\ g &= \sigma\left(\sum_{i=1}^{L_B} \beta_i \boldsymbol{B}_i\right), \end{aligned} \tag{9}$$

where $\{\alpha_i\}$ and $\{\beta_i\}$ are trainable weights, σ is the sigmoid function, and $\otimes$ is the element-wise product. Note that the weights $\{\alpha_i\}$ and

$\{\beta_i\}$ are not shared across the encoder/decoder layers so that each layer can independently determine which BERT layers to concentrate on.

2.4. BERT-JAM architecture

The architecture of our proposed BERT-JAM is shown in Fig. 2. BERT-JAM consists of three main parts: an L_B-layer BERT model, an L-layer encoder, and an L-layer decoder. Given a sequence of source tokens $X = (x_1, x_2, \ldots, x_n)$, BERT-JAM generates the target tokens $Y = (y_1, y_2, \ldots, y_m)$ by performing the following steps:

Step 1: The source sequence X is first embedded and encoded by an L_B-layer BERT model. We extract the multi-layer representations from BERT's intermediate layers, represented as $\vec{\boldsymbol{B}} = \mathtt{BERT}(X) = (\boldsymbol{B}_1, \boldsymbol{B}_2, \ldots, \boldsymbol{B}_{L_B})$.

Step 2: Each encoder/decoder layer is equipped with an independent fusion module used to compose BERT's multi-layer representations, produced in the first step, into a fused representation $\hat{\boldsymbol{B}} = \mathtt{Fuse}(\vec{\boldsymbol{B}})$, as described in Section 2.3. The fused BERT representation produced by the i-th encoder/decoder layer is denoted as $\hat{\boldsymbol{B}}^i_{Enc}/\hat{\boldsymbol{B}}^i_{Dec}$.

Step 3: The encoder contains L identical layers that progressively transform the embedded representation of X into a hidden representation. We denote by $\boldsymbol{E}^i = (e^i_1, e^i_2, \ldots, e^i_n)$ the representation of the i-th encoder layer, where $i \in [0, L]$ and $\boldsymbol{E}^0 = \mathtt{Embed}(X)$ is the embeddings of X. The i-th encoder layer takes as input $\boldsymbol{E}^{i-1}$, which is produced by the previous layer, and transforms it into $\boldsymbol{E}^i$, given by

$$\begin{aligned} \boldsymbol{E}' &= \mathtt{Attn}_{\mathtt{joint}}(\boldsymbol{E}^{i-1}, \hat{\boldsymbol{B}}^i_{Enc}), \\ \boldsymbol{E}^i &= \mathtt{FFN}(\boldsymbol{E}'), \end{aligned} \tag{10}$$

where $\mathtt{Attn}_{\mathtt{joint}}()$ is the joint-attention module described in Section 2.2 and $\mathtt{FFN}()$ is the position-wise feed-forward network introduced by Transformer [7]. The joint-attention module allows the encoder layer to simultaneously perform self-attention and BERT-encoder cross attention. We also follow the Transformer architecture to adopt residual connections [9] and layer normalization [10], which we omit in the formula for simplicity. The output of the last encoder layer is denoted by $\boldsymbol{E}^L$.

Step 4: The decoder workers differently in the training stage and the inference stage. In the training stage, the decoder takes as input the ground-truth target sequences and outputs the probability distribution of each token conditioned on the previous token. We denote by $\boldsymbol{D}^i = (d^i_1, d^i_2, \ldots, d^i_m)$ the representation of the i-th decoder layer, where $i \in [0, L]$ and $\boldsymbol{D}^0 = \mathtt{Embed}(Y)$ is the embedding of Y. At the i-th decoder layer, $\boldsymbol{D}^{i-1}$ is transformed into $\boldsymbol{D}^i$ following

$$\begin{aligned} \boldsymbol{D}' &= \tfrac{1}{2}\left(\mathtt{Attn}_{\mathtt{joint}}(\boldsymbol{D}^{i-1}, \hat{\boldsymbol{B}}^i_{Dec}) + \mathtt{Attn}_{\mathtt{joint}}(\boldsymbol{D}^{i-1}, \boldsymbol{E}^L)\right), \\ \boldsymbol{D}^i &= \mathtt{FFN}(\boldsymbol{D}'). \end{aligned} \tag{11}$$

Each decoder layer employs two joint-attention modules for the transformation. The first one, named BERT-decoder joint attention, performs self-attention and BERT-decoder cross attention at the same time. And the second one, called encoder-decoder joint attention, allows the decoder representation to simultaneously attend to itself and the encoder representation. The output of the last decoder layer, $\boldsymbol{D}^L$, is linearly projected to the target vocabulary space and normalized by a softmax function to obtain the probability distribution. In the inference stage, the decoder works in an autoregressive manner. Since the working principles are similar, we omit further descriptions of this stage.

2.5. Three-phase optimization

For a wide variety of NLP tasks, fine-tuning BERT, i.e., adding a few layers atop BERT and training on the task-specific dataset for a few iterations, has been shown an economical yet highly effective method to obtain well-performing models. However, previous studies [4] claim that fine-tuning BERT in BERT-enhanced NMT models offers no gain due to the catastrophic forgetting problem [6]. As a result, it has been common to keep BERT frozen when training BERT-enhanced NMT models.

However, since BERT accounts for a large portion of the parameters, giving up fine-tuning BERT harms the chances of maximizing the utilization of BERT. As we have observed, the crux of the matter

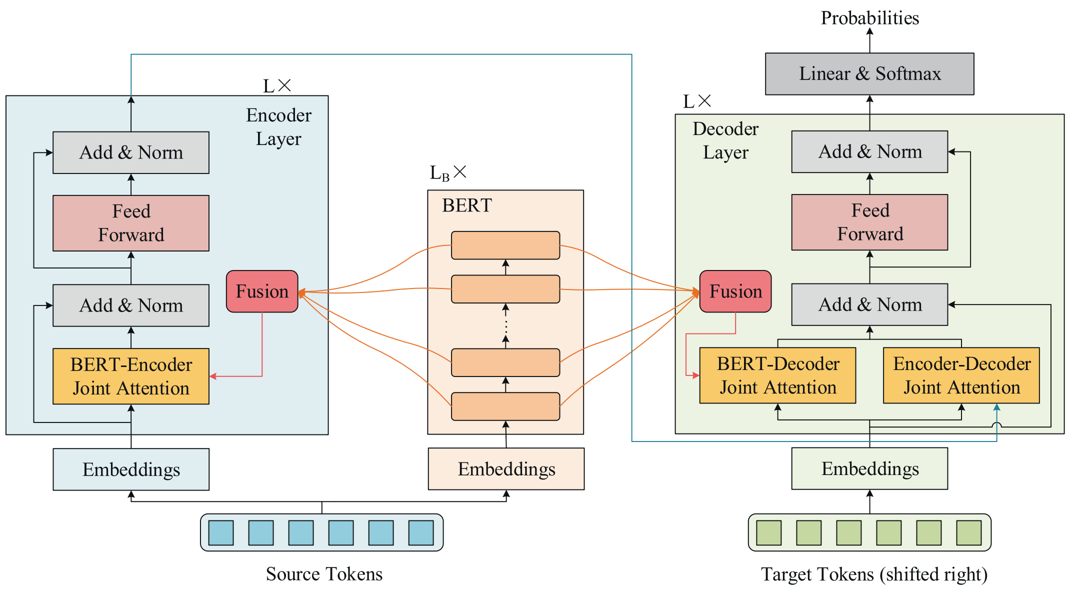


Fig. 2. The architecture of BERT-JAM. The leftmost and rightmost components correspond to the encoder and decoder respectively, both of which consist of L layers. In the middle is an L_B-layer BERT model which provides multi-layer representations of the source tokens.

is that BERT-enhanced NMT models take much more iterations to converge compared with other NLP models such as classification models. For one thing, instead of adding a few untrained layers atop BERT, NMT models built upon BERT additionally introduce an entire encoder-decoder architecture, making the size of untrained parameters comparable to that of pre-trained BERT. For another thing, a large amount of bilingual data is typically required to fully train the NMT model. Consequently, it takes so many iterations to optimize the untrained parameters that the pre-trained knowledge carried by BERT is gradually lost during learning of the new knowledge.

Based on the above observation, we find that the key to this problem is to fully optimize the untrained parameters before fine-tuning BERT. Inspired by the gradual unfreezing method [11] used for fine-tuning pre-trained models on text classification tasks, we propose a three-phase optimization strategy to train BERT-JAM. Specifically, we divide all the parameters of BERT-JAM into three partitions that belong to different components. The three partitions of parameters correspond to the encoder-decoder architecture, the fusion modules, and the pre-trained BERT, respectively. The training proceeds by progressively unfreezing the parameters in different partitions, as described below.

Phase 1: Warmup Training. At the first phase, we optimize the untrained parameters in the encoder and decoder while keeping the parameters of BERT and the fusion modules frozen. Since the fusion modules are not trained at this phase, it's crucial that the parameters are properly initialized to provide a meaningful fusion of the BERT representations. Hence, we initialize each fusion module so that only the last-layer representation of BERT is leveraged, formally given by

$$\begin{aligned}\alpha_i &= \begin{cases}1, & i = L_B\\ 0, & i \in [1, L_B - 1]\end{cases}\\ \beta_i &= 0, \quad i \in [1, L_B].\end{aligned} \tag{12}$$

Replacing α_i and β_i in Eq. (9) with the above initialization yields $g = \frac{1}{2}$ and $\hat{\boldsymbol{B}} = \frac{\boldsymbol{B}_{L_B}}{2}$, which means that only the last layer of BERT is fed into the encoder and the decoder. To compensate for the halved value, we multiply the output of the fusion module by 2 only at this phase. The reason for this initialization is that the way of training BERT determines that its last-layer representation contains the most useful information for downstream tasks. And starting from the most salient representation allows the model to converge faster.

Phase 2: Adjust Fusion Weights. After the parameters in the encoder and the decoder are fully trained, we additionally unfreeze the fusion modules to allow BERT's intermediate layers to contribute to the model. As the training progresses, the fusion module gradually adjusts the weights used to compose BERT's multi-layer representations into a fused representation. The fusion modules contained in the encoder/decoder layers are independent of each other so that they can individually develop preferences for certain BERT layers.

Phase 3: Fine-tune BERT. At the final phase, we unfreeze the whole model and fine-tune the BERT parameters to further boost the performance. The unfreezing of the BERT layers introduces a large number of trainable parameters, hence it's crucial to keep the training under control so that the model won't overfit on the training data. With this in mind, we keep a careful eye on the validation loss at the end of each epoch and stop training when we observe degraded performance.

3. Experimental setup

This section details the experimental setup in terms of data preparation, model configuration, training settings, and evaluation metrics. We implement our model upon the Fairseq repository.[1] The source code for our work has been made publicly available.[2]

Table 1
Construction of the datasets.

Corpus	Training	Validation	Test
IWSLT'14 En-De	160k	(7k *from training set*)	{*dev2010, dev2012, tst2010, tst2011, tst2012*}
IWSLT'14 En-Es	183k	{*dev2010, tst2010, tst2011, tst2012*}	{*tst2013, tst2014*}
IWSLT'17 En-Fr	236k	{*tst2011, tst2012, tst2013, tst2014, tst2015*}	{*tst2016, tst2017*}
IWSLT'17 En-Zh	235k	{*tst2011, tst2012, tst2013, tst2014, tst2015*}	{*tst2016, tst2017*}
WMT'14 En-De	4.5M	{*newstest2012, newstest2013*}	{*newstest2014*}

3.1. Data preparation

We conduct extensive experiments on multiple translation tasks in both low-resource and high-resource scenarios. For the low-resource scenario, we work on multiple IWSLT corpora [12] containing a few hundred thousand training examples for each language pair. For the high-resource scenario, we experiment on the WMT'14 English-German corpus which contains millions of sentence pairs. We construct training, validation, and test datasets from these corpora as shown in Table 1, and describe the preprocessing of the data as follows.

3.1.1. IWSLT

For the low-resource scenario, we evaluate BERT-JAM on five IWSLT translation tasks, namely, IWSLT'14 English-German (En-De), IWSLT'14 German-English (De-En), IWSLT'14 English–Spanish (En-Es), IWSLT'17 English-French (En-Fr), and IWSLT'17 English-Chinese (En-Zh). We prepare the data following the setup in [5]. The numbers of training examples for the tasks are presented in Table 1. Note that for the IWSLT'14 En-De and De-En tasks, the validation set is constructed by drawing 7k examples from the training set, and the test set is constructed by concatenating multiple TED Talk files. For other IWSLT tasks, the TED Talk files of the corresponding years are used as validation/testing sets. As for the data preprocessing, letters are lowercased for the En-De and De-En tasks only. And words in all datasets are tokenized using Moses toolkit [13] and split into sub-words using byte pair encoding (BPE) [14] with 10k symbols. A joined vocabulary is built by merging source and target sentences for each language pair.

3.1.2. WMT

For the high-resource scenario, we evaluate BERT-JAM on the WMT'14 En-De dataset in both translation directions which contains 4.5M training sentence pairs. We tokenize the data as in the IWSLT tasks without lowercasing the letters. Also, words are split into sub-words using BPE with 32k symbols and a joined vocabulary is built. Following the setup in [5], we develop on the concatenation of *newstest2012* and *newstest2013* and test on *newstest2014.*

3.2. Model configuration

For all the translation tasks, both the encoder and the decoder in BERT-JAM consist of $L = 6$ layers. We denote by d_{model}, d_{ff} and N_{head} the embedding dimension, the feed-forward network dimension and the number of attention heads respectively. For the IWSLT

[1] https://github.com/pytorch/fairseq
[2] https://github.com/HollowFire/bert-jam

tasks, we set $d_{model} = 512, d_{ff} = 1024$ and $N_{head} = 4$. For the WMT tasks, we employ a larger model with $d_{model} = 1024, d_{ff} = 4096$ and $N_{head} = 16$. The dropout ratio is set to 0.3.

Depending on the translation tasks, we use BERT models in different sizes that are pre-trained for different languages, as listed in Table 2. For IWSLT tasks we choose $\text{BERT}_{\text{BASE}}$ with $L_B = 12$ layers and embedding dimension 768. Note that the `bert-base-german-uncased`[3] and `bert-base-uncased`[4] models are pre-trained on German and English corpora respectively. We have intended to use the $\text{BERT}_{\text{LARGE}}$ model with $L_B = 24$ layers and embedding dimension 1024 for the WMT tasks. But since there's no available $\text{BERT}_{\text{LARGE}}$ pre-trained for German, we resort to the smaller `bert-base-german-uncased` for the WMT De-En task. As for the WMT En-De task, we use `bert-large-uncased`[5] which is the enlarged version of `bert-base-uncased`.

As part of our experiments to be presented in Section 4.2, we will evaluate BERT-JAM on IWSLT'14 En-De with pre-trained BERT models of varying sizes ranging from the most compact BERT model with 2 encoder layers and embedding dimension 128 to the largest one which is the same as $\text{BERT}_{\text{BASE}}$. These pre-trained models are available at this repository.[6]

3.3. Training

We experiment on a single machine equipped with 4 NVIDIA V100 GPUs. We train the model using the Adam optimizer [15] with $\beta_1 = 0.9$ and $\beta_2 = 0.98$. The learning rate follows the inverse square root schedule which is firstly linearly warmed up from 10^{-7} to 0.0005 for $4k$ steps and then given by

$$lr = 0.0005 \frac{\sqrt{step_{warm\_up}}}{\sqrt{step_{current}}}. \tag{13}$$

The models are trained in batches containing up to $4k$ tokens. For the WMT tasks, we accumulate the gradients for 32 iterations before updating to simulate training on 128 GPUs. For all tasks, we follow the three-phase optimization strategy described in Section 2.5. Following previous studies [7,16], we obtain the model used for inference by averaging the checkpoint weights of the last 10 epochs. We train the model until convergence in the first two phases. In the third phase, since too much training makes the pre-trained knowledge become forgotten, we stop when the model achieves minimal loss on the validation set.

3.4. Evaluation

We use beam search to generate target sentences on the test sets in inference mode. Beam search is a heuristic search algorithm widely used for sequence-to-sequence generation. Instead of selecting the word with the maximum probability at each decoding step as in the greedy search algorithm, beam search tries to find the most likely sequence of words based on the joint probability. It is parameterized by a beam width which is the size of the candidate set. Besides, length penalty is applied to handle sentences of different lengths. Following the setup in [5], for the IWSLT tasks, we set beam width to 5 and length penalty to 1. For the WMT tasks, the beam width is 4 and the length penalty is 0.6.

BLEU score [17] is a commonly used metric for the measurement of translation qualities. It works by counting matching n-grams between the candidate translation and the reference text.

Table 2
BERT models used for different tasks.

Task		BERT model
IWSLT	De-En	`bert-base-german-uncased`
	En-De	`bert-base-uncased`
	En-Es	
	En-Fr	
	En-Zh	
WMT	De-En	`bert-base-german-uncased`
	En-De	`bert-large-uncased`

However, since BLEU is a parameterized metric, different choices of the parameters can lead to the variation of the scores. For a fair comparison with existing studies, we follow the same implementations of BLEU described in their papers. Specifically, for translation tasks in both directions on the IWSLT'14 En-De and the WMT'14 En-De datasets, we use `multi-bleu.perl`.[7] For other tasks, we use detokenized SACREBLEU [18] instead. Note that for the WMT'14 En-De task only, we additionally perform compound splitting following Vaswani et al. [7] before calculating BLEU scores to produce comparable results.

4. Experiments and results

This section introduces the experiments we conduct to evaluate our method. We first report the results on the benchmark translation tasks in both low-resource and high-resource scenarios. Then we work on the IWSLT'14 En-De task where we vary the size of BERT to study how it affects the translation performance. Finally, we conduct ablation studies to justify the design choices in our proposed model.

4.1. Main results

4.1.1. IWSLT

For the low-resource scenario, we experiment on multiple IWSLT datasets. We present in Table 3 the results on the widely used IWSLT'14 De-En dataset. As shown, BERT-JAM outperforms the previous SOTA model by a large margin, setting a new SOTA score of 38.66 on this task.

The results of the IWSLT En-X (X∈{De, Es, Fr, Zh}) tasks are presented in Table 4. We compare BERT-JAM with two baseline models, the Transformer and the BERT-fused model. As shown, BERT-JAM outperforms the BERT-fused model on all translation tasks except En-Zh. The findings could be explained by the fact that our proposed joint-attention module is better at translating between languages that share cognate words. As for distant language pairs such as English and Chinese, attending to their concatenated token representations might hurt the performance.

4.1.2. WMT

For the high-resource scenario, we experiment on the WMT'14 En-De and De-En tasks. The WMT'14 En-De dataset is one of the most used benchmark tasks for evaluating NMT models. As reported in Table 5, we compare BERT-JAM with strong baseline models including those which make use of pre-trained language models (the lower half) and those which don't (the upper half). BERT-JAM outperforms all the previous models and achieves a BLEU score of 31.59. Our result is only inferior to those models that utilize extra training data beyond the corpus [25,26].

For the WMT'14 De-En task, as shown in Table 6, by comparing to the latest results reported in the literature so far, our BERT-JAM

[3] https://s3.amazonaws.com/models.huggingface.co/bert/bert-base-german-dbmdz-uncased-pytorch_model.bin
[4] https://s3.amazonaws.com/models.huggingface.co/bert/bert-base-uncased.tar.gz
[5] https://s3.amazonaws.com/models.huggingface.co/bert/bert-large-uncased.tar.gz
[6] https://github.com/google-research/bert

[7] https://github.com/moses-smt/mosesdecoder/blob/master/scripts/generic/multi-bleu.perl

Table 3
Translation qualities on the IWSLT'14 De-En task.

Model	BLEU
Transformer-Base [7]	34.64
DynamicConv [16]	35.2
BERT-fused [5]	36.11
MAT [19]	36.22
MUSE [20]	36.3
MAT + Knee [21]	36.6
BERT-JAM	**38.66**

Table 4
Translation qualities on the IWSLT En-X tasks.

	Transformer-Base	BERT-fused	BERT-JAM
En-De	28.57	30.34	**31.20**
En-Es	39.0	41.4	**42.3**
En-Fr	35.9	38.7	**39.8**
En-Zh	26.3	**28.2**	27.9

Bold indicates the model with the best BLEU score for each task.

Table 5
Translation qualities on the WMT'14 En-De task.

Model	BLEU
Transformer-Big [7]	29.3
DynamicConv [16]	29.7
Evolved Transformer [22]	29.8
MUSE [20]	29.9
Imamura and Sumita [23]	29.04
APT framework [3]	29.23
CTNMT [4]	30.1
BERT-fused [5]	30.75
BERT-JAM	**31.59**

achieves a BLEU score of 33.85, advancing the SOTA score by 2 points.

4.2. Varying BERT size

Although BERT-enhanced NMT models can benefit from leveraging a deeper and wider BERT model, the performance boost comes at the cost of increased computational complexity, at both training stage and inference stage. To apply our method to scenarios that put restrictions on memory usage and latency bound, we find it necessary to study the variation in performance with the size of the leveraged BERT model. To this end, we work on IWSLT'14 En-De by feeding BERT-JAM with BERT models in different sizes and keeping the rest of the architecture unchanged. The BERT models used in this experiment have layer numbers L_B ranging from 2 to 12 and embedding dimensions H ranging from 128 to 768, leading to 24 different models.

The translation qualities in terms of BLEU score for all models are presented in Table 7 where the numbers in the parentheses represent the parameter sizes of the corresponding BERT models. We present a scatter plot in Fig. 3 to provide better visualization of the results. For the data points in the plot, we use markers in different shapes to represent different embedding dimensions (as annotated in the legend) and use different colors to represent different layer numbers (the deeper the color, the larger the layer number). For convenience, in the following discussions, we use the notation $L_B - m/H - n$ to denote the model with a layer number of m and an embedding dimension of n.

Table 6
Translation qualities on the WMT'14 De-En task.

Model	BLEU
FlowSeq-large [24]	28.29
Transformer-Big [7]	31.44
MAT + Knee [21]	31.9
BERT-JAM	**33.85**

Several interesting observations can be obtained from the results. Firstly, when the embedding dimension is small ($H = 128/256$), increasing the layer number effectively improves the BLEU score without significantly enlarging the model size. Secondly, given an approximately same parameter size, deeper models (larger layer number) achieve higher BLEU scores than the wider ones (larger embedding dimension). For example, as shown in Table 7, $L_B - 8/H - 512$ ($41M$ parameters) outperforms $L_B - 2/H - 768$ ($39M$ parameters) by nearly 1 point of BLEU score. More notably, $L_B - 12/H - 256$ ($17M$ parameters), though with a much smaller parameter size, is able to outperform $L_B - 2/H - 768$ ($39M$ parameters). Thirdly, given an approximately same BLEU score, deeper models have the advantage of retaining a smaller parameter size. For example, $L_B - 12/H - 256$ ($17M$ parameters) ties with $L_B - 6/H - 512$ ($35M$ parameters) with a much smaller parameter size. So is the case with $L_B - 12/H - 512$ ($54M$ parameters) and $L_B - 10/H - 768$ ($95M$ parameters).

With the above observations, we can safely draw the conclusion that, when we have to resort to a more compact architecture due to memory and latency constraints, it's advisable to prioritize layer number over embedding dimension.

4.3. Ablation study

In order to justify the design choices we make for the model architecture and the optimization strategy, we conduct a group of ablation studies on the IWSLT'14 De-En task. We compare the full-featured BERT-JAM with different variations of the model, as described below.

- $\mathcal{M}_0$: We train the full-featured BERT-JAM following the three-phase optimization strategy. To provide insightful analysis of the model and the training process, we report the BLEU scores after finishing each optimization phase.
- $\mathcal{M}_1/\mathcal{M}_2$: To justify the usage of BERT for the encoder-decoder architecture, we train two variations, $\mathcal{M}_1$ and $\mathcal{M}_2$, which correspond to using BERT only for the encoder and the decoder, respectively.
- $\mathcal{M}_3$: To study the design choice we make for the fusion module, we substitute this module with a naive implementation which simply takes a linear combination of BERT's multi-layer representations. The same three-phase optimization strategy is employed for the training.
- $\mathcal{M}_4$: To study if BERT's intermediate layers contribute to the translation performance, we only use BERT's last-layer representation for this variation. Since the fusion module is not needed, we merge the first two optimization phases and keep the total number of iterations unchanged.
- $\mathcal{M}_5$: To prove the effectiveness of our proposed optimization strategy, we train BERT-JAM with a simpler two-phase optimization scheme for this variation. At the first phase, we freeze BERT and train the parameters of the encoder-decoder architecture and the fusion modules. At the second phase, we unfreeze BERT and optimize all the parameters.

Table 7
Translation qualities on the IWSLT'14 En-De task with varying BERT sizes.

BLEU (Params)	$H = 128$	$H = 256$	$H = 512$	$H = 768$
$L_B = 2$	28.63(4 M)	29.18(10 M)	29.63(22 M)	29.67(39 M)
$L_B = 4$	28.77(5 M)	29.36(11 M)	30.12(29 M)	30.46(53 M)
$L_B = 6$	28.90(5 M)	29.58(13 M)	30.30(35 M)	30.73(67 M)
$L_B = 8$	28.99(6 M)	29.92(14 M)	30.63(41 M)	30.75(81 M)
$L_B = 10$	29.13(6 M)	30.04(16 M)	30.73(48 M)	30.92(95 M)
$L_B = 12$	29.25(6 M)	30.29(17 M)	30.84(54 M)	31.20(109 M)

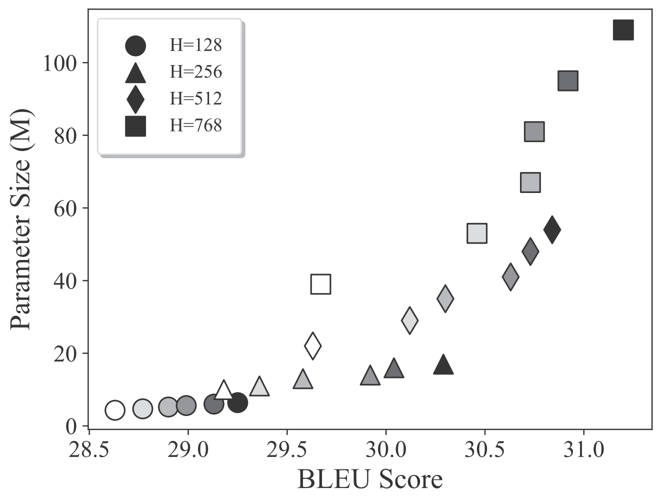


Fig. 3. Scatter plot of BLEU score and parameter size for BERT models with varying L_B and H. Markers in different shapes correspond to different embedding dimensions, and deeper colors indicate larger layer numbers.

Table 8 shows the BLEU scores of different variations at the end of each optimization phase. We discuss the results from different aspects of our proposed approach.

4.3.1. Usage of BERT

By comparing $\mathcal{M}_0$ with $\mathcal{M}_1$ and $\mathcal{M}_2$, we find that using BERT for both the encoder and decoder achieves the best performance. If BERT is used only for the encoder or decoder, the scores drop to 37.95 and 38.06, respectively. This shows that both the encoder and the decoder can benefit from utilizing the BERT representations.

4.3.2. Fusion module

BERT-JAM employs the fusion module to make use of BERT's multi-layer representations. We find that, compared with only utilizing BERT's last-layer representation ($\mathcal{M}_4$), adding the fusion module ($\mathcal{M}_0$) helps improve the BLEU score from 38.35 to 38.66. However, by comparing $\mathcal{M}_3$ with $\mathcal{M}_4$, it's surprising that linearly combining BERT's intermediate layers achieves almost the same performance as simply using the last-layer representation. This suggests that a naive approach to handling BERT's intermediate layers fails to make use of the linguistic features encoded by these layers, and that the performance boost brought by the fusion module is largely attributed to the incorporated gating mechanism.

4.3.3. Fine-tuning BERT

To provide an intuitive understanding of the training process, we plot the training and validation loss of the full-featured BERT-JAM in Fig. 4. Notably, at the fine-tuning phase (epoch 51∼60), validation loss first shows a sudden drop and then rebounds. The drop in validation loss can be explained by the update of BERT's parameters. To be specific, in this experiment, BERT-JAM contains 159*M* parameters in total, with 109*M* of them belonging to BERT.

Table 8
Translation qualities on the IWSLT'14 De-En task in terms of BLEU scores for each variation at the end of each phase.

Model	BLEU score		
	phase 1 (40 epochs)	phase 2 (10 epochs)	phase 3 (10 epochs)
$\mathcal{M}_0$	36.85	37.25	38.66
$\mathcal{M}_1/\mathcal{M}_2$	36.37/36.51	36.84/37.02	37.95/38.06
$\mathcal{M}_3$	36.71	37.07	38.39
$\mathcal{M}_4$	36.97		38.35
$\mathcal{M}_5$	37.12		38.17

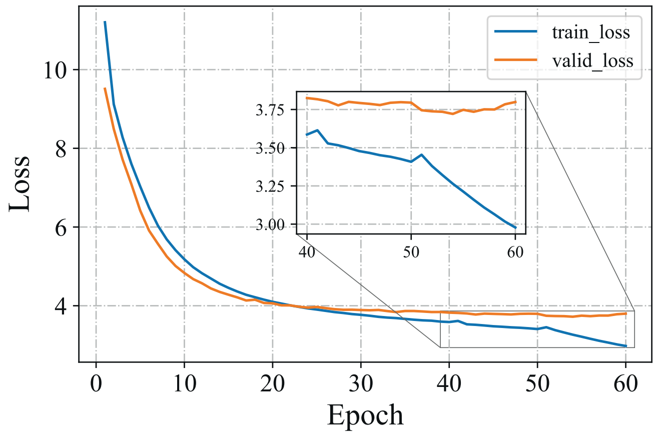


Fig. 4. Training and validation loss on the IWSLT'14 De-En task as a function of training epochs.

Unfreezing of these parameters allows for further optimization of the model. As a downside, the rebound of the validation loss shows that the hazard of catastrophic forgetting appears during training and the model starts to overfit on the training data. The dramatic drop of the training loss confirms the overfitting. Since we average the checkpoints of the last 10 epochs to obtain the final model used for inference, in practice we don't stop training immediately the validation loss starts to rebound. Instead, we wait until the lowest point of the validation loss falls in the middle of the averaging interval. As shown in Table 8, our method allows the third optimization phase to bump up the BLEU score of $\mathcal{M}_0$ from 37.25 to 38.66.

4.3.4. Phased training

It might seem odd that our optimization strategy trains the parameters of the encoder-decoder architecture and the fusion modules in different phases instead of training them altogether. Intuitively, we let the parameters of the encoder-decoder architecture be fully optimized before introducing BERT's intermediate layers. Since the pre-training of BERT is based on its last-layer representation, freezing the fusion modules in the first optimization phase contributes to stabilized training of BERT-JAM. By comparing $\mathcal{M}_0$ with $\mathcal{M}_5$, we find that optimizing the untrained parameters altogether makes the BLEU score drop from 38.66 to 38.17.

5. Related work

5.1. Neural machine translation

Neural machine translation (NMT), which aims to use artificial neural networks for the translation between human languages, has drawn continuous research attention over the past decade. NMT is proposed in contrast to traditional statistical machine translation (SMT) such as phrase-based SMT. Instead of trying to tune many sub-components which requires heavy engineering, NMT uses a simpler end-to-end model to generate translations word by word. The main difference that distinguishes an NMT model from traditional SMT models is that it uses continuous vector space for the representation of words [27]. And the continuous representations allow the NMT model to efficiently learn the mapping between source and target sentences without suffering from the problem of sparsity. It has been demonstrated that even a naive and straightforward NMT model can outperform the widely studied and mature SMT systems [28].

Most of the NMT models used today are based on the encoder-decoder architecture which was first proposed by Cho et al. [29] formally. In the beginning, most NMT models were based on recurrent neural networks (RNN). Cho et al. [29] first used an RNN encoder to transform the source sentence into a fixed-length context vector and then used an RNN decoder to predict the target sentence based on the vector. This practice has been followed by later researches [30,31].

After realizing that the fixed-length vector is the bottleneck of the NMT model, Bahdanau et al. [32] proposed an attention mechanism that allows the decoder to concentrate on certain parts of the source sentence when translating. Many of the subsequent studies on NMT are under the same framework of an encoder-decoder architecture and an attention-based approach. The key feature that distinguishes these studies from each other is the block-building structures used in their models. Some of them focused on the recurrent structures including the standard RNN [32], LSTM [33,34], and GRU [35]. Others explored replacing the recurrent units with convolutional structures [36,16]. Vaswani et al. [7] took the idea of attention one step further and proposed the famous Transformer model. The transformer layers dispense with any recurrent or convolutional units and depend solely on the attention mechanism to perform the transformation of hidden states. Follow-up researches are devoted to improving Transformer with various techniques [37–39].

Existing studies have explored leveraging joint attention for NMT [40,38], which is similar to the module employed by BERT-JAM. However, their method mixes the query/key/value vectors of self-attention modules between the corresponding encoder and decoder layers to enable layer-wise coordination and parameter sharing, which aims to keep the corresponding layer representations at the same semantic level. In contrast, the implementation of BERT-JAM keeps the query vector untouched and integrates the key/value vectors with BERT representations, which serves to provide richer contextual information.

5.2. BERT-enhanced NMT

BERT [1] is essentially a Transformer encoder that is pre-trained on two tasks, namely, masked language modeling and next sentence prediction. It takes as input a sequence of words and encodes them into hidden representations. A large amount of unlabeled text data are used for the pre-training such that BERT can learn contextualized representations of words. The BERT-encoded representations of the input sequences are associated with rich contextual information which greatly assists NLP tasks. It has been shown that building neural models upon BERT is highly effective in achieving significant performance gains.

BERT-enhanced NMT aims at improving translation performance by utilizing the BERT representations. Researchers take different approaches to exploiting BERT for NMT. Imamura and Sumita [23] proposed to substitute the Transformer encoder with a pre-trained BERT and optimize on the bilingual corpus. In order to cope with the problem of catastrophic forgetting, Yang et al. [4] proposed an asymptotic distillation method to transfer the pre-trained information from BERT to the NMT model. Clinchant et al. [41] performed a systematic study of different approaches to boosting translation performance using BERT. Our method is inspired by the recently proposed BERT-fused model [5] which employs attention mechanisms to fuse the BERT representation with the encoder/decoder layers. They use separate modules to performs different attention functions, whereas we propose to integrate self-attention and cross-attention modules into a single joint-attention module. We notice that Weng et al. [3] also explored utilizing BERT's intermediate layers for NMT. However, our fusion method and theirs exhibit fundamental differences in the mechanism, which has been elaborated on in Section 2.3.

6. Conclusions

In this work, we identify the research gap in maximizing the utilization of BERT for translation tasks and propose a BERT-enhanced NMT model called BERT-JAM to fill this gap in three ways. Firstly, we equip BERT-JAM with fusion modules for fusing BERT's multi-layer representations to allow the model to leverage the linguistic features encoded by BERT's intermediate layers. Secondly, we propose a joint-attention module to allow the encoder/decoder layers to dynamically allocate attention between the fused BERT representation and the encoder/decoder representation. And thirdly, we train BERT-JAM with a novel three-phase optimization strategy to benefit from fine-tuning BERT while avoiding catastrophic forgetting. The comprehensive evaluation shows that BERT-JAM outperforms existing models and achieves new SOTA BLEU scores on multiple translation tasks, demonstrating the effectiveness of our method.

The method proposed in this paper belongs to the feature-based methods which use pre-trained language models to provide contextual representations of the source language to boost the performance of NMT models. With the advent of more powerful models pre-trained with a generative objective (e.g., GPT [42]), it's interesting to explore enriching the target language representations using such pre-trained models. Nevertheless, feature-based methods cannot be straightforwardly employed for this purpose because heavy computation cost is incurred at the inference stage. Instead, previous works proposed to use a distillation method to transfer the knowledge from pre-trained models to the decoder [3]. However, distillation methods cannot make use of the linguistic features encoded by the intermediate layers. Hence, how to effectively integrate different methods might be one of the directions of future studies.

CRediT authorship contribution statement

Zhebin Zhang: Conceptualization, Methodology, Software, Writing - original draft. **Sai Wu:** Data curation, Investigation, Validation. **Dawei Jiang:** Writing - review & editing, Formal analysis, Visualization. **Gang Chen:** Supervision, Project administration, 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.

Acknowledgements

This work was supported by the Key Research and Development Program of Zhejiang Province of China (Grant No. 2020C01024); the Natural Science Foundation of Zhejiang Province of China (Grant No. LY18F020005); and the National Natural Science Foundation of China (Grant No. 61872315).

References

[1] J. Devlin, M. 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, 2019, pp. 4171–4186, doi: 10.18653/v1/n19-1423..
[2] 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, 2019, pp. 3651–3657, doi: 10.18653/v1/P19-1356..
[3] R. Weng, H. Yu, S. Huang, S. Cheng, W. Luo, Acquiring knowledge from pre-trained model to neural machine translation, in: The 34th AAAI Conference on Artificial Intelligence, 2020, pp. 9266–9273.
[4] J. Yang, M. Wang, H. Zhou, C. Zhao, W. Zhang, Y. Yu, L. Li, Towards Making the Most of BERT in Neural Machine Translation, in: The Thirty-Fourth AAAI Conference on Artificial Intelligence, 2020, pp. 9378–9385..
[5] J. Zhu, Y. Xia, L. Wu, D. He, T. Qin, W. Zhou, H. Li, T. Liu, Incorporating BERT into neural machine translation, in: 8th International Conference on Learning Representations, 2020..
[6] I.J. Goodfellow, M. Mirza, D. Xiao, A. Courville, Y. Bengio, An Empirical Investigation of Catastrophic Forgetting in Gradient-Based Neural Networks, 2015..
[7] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, u. Kaiser, I. Polosukhin, Attention is all you need, in: Proceedings of the 31st International Conference on Neural Information Processing Systems, 2017, pp. 6000–6010.
[8] Y.N. Dauphin, A. Fan, M. Auli, D. Grangier, Language Modeling with Gated Convolutional Networks 70 (2017) 933–941.
[9] K. He, X. Zhang, S. Ren, J. Sun, Deep Residual Learning for Image Recognition, in: 2016 IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770–778, doi: 10.1109/CVPR.2016.90..
[10] L.J. Ba, J.R. Kiros, G.E. Hinton, Layer Normalization..
[11] J. Howard, S. Ruder, Universal Language Model Fine-tuning for Text Classification, in: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, 2018, pp. 328–339, doi: 10.18653/v1/P18-1031..
[12] M. Cettolo, C. Girardi, M. Federico, WIT3: Web Inventory of Transcribed and Translated Talks, in: Proceedings of the 16th Annual conference of the European Association for Machine Translation, 2012, pp. 261–268.
[13] P. Koehn, H. Hoang, A. Birch, C. Callison-Burch, M. Federico, N. Bertoldi, B. Cowan, W. Shen, C. Moran, R. Zens, C. Dyer, O. Bojar, A. Constantin, E. Herbst, Moses: Open Source Toolkit for Statistical Machine Translation, in: Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics, 2007..
[14] R. Sennrich, B. Haddow, A. Birch, Neural machine translation of rare words with subword units, in: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, 2016, https://doi.org/10.18653/v1/p16-1162.
[15] D.P. Kingma, J. Ba, Adam: A method for stochastic optimization, in: 3rd International Conference on Learning Representations, 2015.
[16] F. Wu, A. Fan, A. Baevski, Y.N. Dauphin, M. Auli, Pay less attention with lightweight and dynamic convolutions, in: 7th International Conference on Learning Representations, 2019.
[17] K. Papineni, S. Roukos, T. Ward, W.-J. Zhu, Bleu: a method for automatic evaluation of machine translation, in: Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, 2002, pp. 311–318, doi: 10.3115/1073083.1073135..
[18] M. Post, A Call for clarity in reporting BLEU scores, in: Proceedings of the 3rd Conference on Machine Translation: Research Papers, 2018, pp. 186–191, doi: 10.18653/v1/w18-6319..
[19] Y. Fan, S. Xie, Y. Xia, L. Wu, T. Qin, X.-Y. Li, T.-Y. Liu, Multi-branch Attentive Transformer (2020).
[20] G. Zhao, X. Sun, J. Xu, Z. Zhang, L. Luo, MUSE: Parallel Multi-Scale Attention for Sequence to Sequence Learning..
[21] N. Iyer, V. Thejas, N. Kwatra, R. Ramjee, M. Sivathanu, Wide-minima Density Hypothesis and the Explore-Exploit Learning Rate Schedule..
[22] D.R. So, Q.V. Le, C. Liang, The Evolved Transformer, in: Proceedings of the 36th International Conference on Machine Learning, vol. 97, 2019, pp. 5877–5886..
[23] K. Imamura, E. Sumita, Recycling a Pre-trained BERT Encoder for Neural Machine Translation, in: Proceedings of the 3rd Workshop on Neural Generation and Translation, 2019, pp. 23–31, doi: 10.18653/v1/D19-5603..
[24] X. Ma, C. Zhou, X. Li, G. Neubig, E.H. Hovy, FlowSeq: Non-autoregressive conditional sequence generation with generative flow, in: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, 2019, pp. 4281–4291, doi: 10.18653/v1/D19-1437..
[25] S. Edunov, M. Ott, M. Auli, D. Grangier, Understanding back-translation at scale, in: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, 2018, pp. 489–500, doi: 10.18653/v1/D18-1045..
[26] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, P.J. Liu, Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer..
[27] N. Kalchbrenner, P. Blunsom, Recurrent continuous translation models, in: Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, 2013, pp. 1700–1709..
[28] I. Sutskever, O. Vinyals, Q.V. Le, Sequence to sequence learning with neural networks, Advances in Neural Information Processing Systems 27 (2014) 3104–3112.
[29] K. Cho, B. van Merrienboer, Ç. Gülçehre, D. Bahdanau, F. Bougares, H. Schwenk, Y. Bengio, Learning phrase representations using RNN encoder-decoder for statistical machine translation, in: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, 2014, pp. 1724–1734, doi: 10.3115/v1/d14-1179..
[30] T. Luong, I. Sutskever, Q.V. Le, O. Vinyals, W. Zaremba, Addressing the rare word problem in neural machine translation, in: Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing, 2015, 11–19, doi: 10.3115/v1/p15-1002..
[31] K. Cho, B. van Merrienboer, D. Bahdanau, Y. Bengio, On the properties of neural machine translation: encoder-decoder approaches, in: Proceedings of the 8th Workshop on Syntax, Semantics and Structure in Statistical Translation, 2014, 103–111, doi: 10.3115/v1/W14-4012..
[32] D. Bahdanau, K. Cho, Y. Bengio, Neural machine translation by jointly learning to align and translate, in: 3rd International Conference on Learning Representations, 2015.
[33] Y. Wu, M. Schuster, Z. Chen, Q.V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, J. Klingner, A. Shah, M. Johnson, X. Liu, L. Kaiser, S. Gouws, Y. Kato, T. Kudo, H. Kazawa, K. Stevens, G. Kurian, N. Patil, W. Wang, C. Young, J. Smith, J. Riesa, A. Rudnick, O. Vinyals, G. Corrado, M. Hughes, J. Dean, Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation..
[34] T. Luong, H. Pham, C.D. Manning, Effective approaches to attention-based neural machine translation, in: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, 2015, pp. 1412–1421, doi: 10.18653/v1/d15-1166..
[35] S. Jean, K. Cho, R. Memisevic, Y. Bengio, On using very large target vocabulary for neural machine translation, in: Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing, 2015, pp. 1–10.
[36] J. Gehring, M. Auli, D. Grangier, D. Yarats, Y.N. Dauphin, Convolutional sequence to sequence learning, in: Proceedings of the 34th International Conference on Machine Learning, vol. 70, 2017, pp. 1243–1252..
[37] J. Gu, C. Wang, J. Zhao, Levenshtein Transformer, Advances in Neural Information Processing Systems 32 (2019) 11179–11189.
[38] J.A.R. Fonollosa, N. Casas, M.R. Costa-jussà, Joint Source-Target Self Attention with Locality Constraints..
[39] S.R. Indurthi, I. Chung, S. Kim, Look Harder: A neural machine translation model with hard attention, in: Proceedings of the 57th Conference of the Association for Computational Linguistics, 2019, pp. 3037–3043, doi: 10.18653/v1/p19-1290..
[40] T. He, X. Tan, Y. Xia, D. He, T. Qin, Z. Chen, T.-Y. Liu, Layer-wise coordination between encoder and decoder for neural machine translation, Advances in Neural Information Processing Systems 31 (2018) 7944–7954.
[41] S. Clinchant, K.W. Jung, V. Nikoulina, On the use of BERT for neural machine translation, in: Proceedings of the 3rd Workshop on Neural Generation and Translation, 2019, pp. 108–117, doi: 10.18653/v1/D19-5611..
[42] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever, Improving Language Understanding by Generative Pre-Training URL:https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf..

Zhebin Zhang received his B.S. degree from Zhejiang University in 2016. Currently, he is pursuing the Ph.D. degree in computer science and technology at the same university. His research interests include natural language processing and machine learning, mainly focusing on machine translation.

Sai Wu received his Ph.D. degree from National University of Singapore (NUS) in 2011 and is currently an associate professor at College of Computer Science, Zhejiang University. His research interests include distributed databases, cloud systems, indexing techniques and AI-powered databases. He has served as a Program Committee member for VLDB, ICDE, SIGMOD, KDD and CIKM.

Dawei Jiang received his Ph.D. degree from Southeast University in 2008 and is currently a senior researcher at Zhejiang University. His research interests include distributed and parallel database systems, cloud data management, big data management and financial technology. He has published over 20 research articles in international journals and conference proceedings.

Gang Chen received his Ph.D. degree from Zhejiang University in 1998. He is currently a professor at College of Computer Science, Zhejiang University. He has successfully led the investigation in research projects aimed at building China's indigenous database management systems. His research interests range from relational database systems to largescale data management technologies. He is a member of ACM and a senior member of CCF.