Block-Skim: Efficient Question Answering for Transformer

Yue Guan[1,2], **Zhengyi Li**[1,2], **Zhouhan Lin**[1], **Yuhao Zhu**[3], **Jingwen Leng**[1,2], **Minyi Guo**[1,2],

[1] Shanghai Jiao Tong University
[2] Shanghai Qi Zhi Institute
[3] University of Rochester
{bonboru,hobbit,leng-jw}@sjtu.edu.cn, guo-my@cs.sjtu.edu.cn, lin.zhouhan@gmail.com, yzhu@rochester.edu

Abstract

Transformer models have achieved promising results on natural language processing (NLP) tasks including extractive question answering (QA). Common Transformer encoders used in NLP tasks process the hidden states of all input tokens in the context paragraph throughout all layers. However, different from other tasks such as sequence classification, answering the raised question does not necessarily need all the tokens in the context paragraph. Following this motivation, we propose Block-Skim, which learns to skim unnecessary context in higher hidden layers to improve and accelerate the Transformer performance. The key idea of Block-Skim is to identify the context that must be further processed and those that could be safely discarded early on during inference. Critically, we find that such information could be sufficiently derived from the self-attention weights inside the Transformer model. We further prune the hidden states corresponding to the unnecessary positions early in lower layers, achieving significant inference-time speedup. To our surprise, we observe that models pruned in this way outperform their full-size counterparts. Block-Skim improves QA models' accuracy on different datasets and achieves 3× speedup on $\text{BERT}_{\text{base}}$ model.

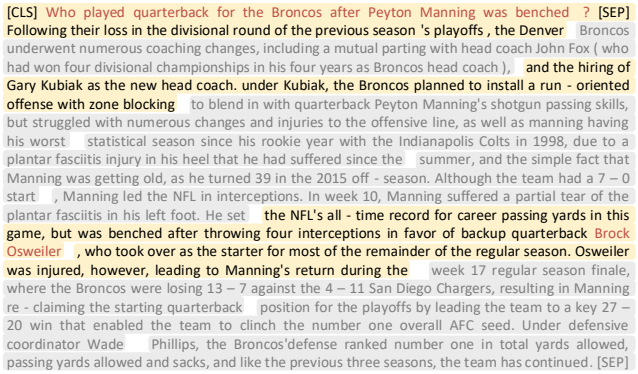

Figure 1: Example of Block-Skim method on a query from the SQuAD dataset. The question and answer tokens are annotated in red. Only question and few evidence blocks are fully processed (annotated by yellow). And other blocks are skimmed for acceleration with the knowledge from attention weights (annotated by grey). Here the block size is 32 tokens.

Introduction

The Transformer model (Vaswani et al. 2017) has pushed model performance on various NLP applications to a new stage by introducing multi-head attention (MHA) mechanism (Lin et al. 2017). Further, the Transformer-based BERT (Devlin et al. 2018) model advances its performances by introducing self-supervised pre-training and has reached state-of-the-art accuracy on many NLP tasks. This has made it at the core of many state-of-the-art models, especially in recent question answering (QA) models (Huang et al. 2020).

Our key insight for QA is that when human beings are answering a question with a passage as a context, they do *not* spend the same level of comprehension for each of the sentences equally across the paragraph. Most of the contents are quickly skimmed over with little attention on it, which means that for a specific question most of the contents are **semantically redundant**. However, in the Transformer architecture, all tokens go through the same amount of computation, which suggests that we can take advantage of that by discarding many of the tokens early in the lower layers of the Transformer. This *semantic level redundancy* sheds light on effectively reducing the sequence lengths at higher layers. Since the execution overhead of self-attention increases quadratically w.r.t. sequence length, this semantic level pruning could significantly reduce the computation time for long contexts.

To excavate the efficiency from this insight, we propose to first chop up the context into blocks, and then learn a classifier to terminate those less relevant ones early in lower layers by looking at the attention weights as shown in Fig. 1. Moreover, with the supervision of ground truth answer positions, a model that jointly learns to discard context blocks as well as answering questions exhibits significantly better performance over its full-size counterpart. Unfortunately, this also makes the proposed Block-Skim method dedicated for extractive QA downstream task. However, QA task is significant in real work production scenarios. Moreover, our method lies in the trade-off space between generality, usability, and efficiency. While sacrificing generality on appli-

cable tasks, our proposed method is easy for adoption as it works as a plug-in for existing models. Similarly, leveraging the QA-specific attention weight patterns makes Block-Skim achieves better speedup results than other methods.

In this paper, we provide the first empirical study on attention feature maps to show that an attention map could carry enough information to locate the answer scope. We then propose Block-Skim, a plug-and-play module to the transformer-based models, to accelerate transformer-based models on QA tasks. By handling the attention weight matrices as feature maps, the CNN-based Block-Skim module extracts information from the attention mechanism to make a skim decision. With the predicted block mask, Block-Skim skips irrelevant context blocks, which do not enter subsequent layers' computation. Besides, we devise a new training paradigm that jointly trains the Block-Skim objective with the native QA objective, where extra optimization signals regarding the question position are given to the attention mechanism directly.

In our evaluation, we show Block-Skim improves the QA accuracy and F1 score on all the datasets and models we evaluated. Specifically, $\text{BERT}_{\text{base}}$ is accelerated for $3\times$ without any accuracy loss.

This paper contributes to the following 3 aspects.

- We for the first time show that an attention map is effective for locating the answer position in the input.
- We propose Block-Skim, which leverages the attention mechanism to improve and accelerate Transformer models on QA tasks. The key is to extract information from the attention mechanism during processing and intelligently predict what blocks to skim.
- We evaluate Block-Skim on several Transformer-based model architectures and QA datasets and demonstrate its efficiency and generality.

Related Work

Recurrent Models with Skimming. The idea to skip or skim irrelevant sections or tokens of input sequence has been studied in NLP models, especially recurrent neural networks (RNN) (Rumelhart, Hinton, and Williams 1986) and long short-term memory network (LSTM) (Hochreiter and Schmidhuber 1997). LSTM-Jump (Yu, Lee, and Le 2017) uses the policy-gradient reinforcement learning method to train an LSTM model that decides how many time steps to jump at each state. They also use hyper-parameters to control the tokens before a jump, maximum tokens to jump, and the maximum number of jumping. Skim-RNN (Seo et al. 2018) dynamically decides the dimensionality and RNN model size to be used at the next time step. In specific, they adopt two "big" and "small" RNN models and select the "small" one for skimming. Structural-Jump-LSTM (Hansen et al. 2018) uses two agents to decide whether to jump by a small step to the next token or structurally to the next punctuation. Skip-RNN (Campos et al. 2017) learns to skip state updates and thus results in the reduced computation graph size. The difference of Block-Skim to these works is twofold. First, the previous works make the skimming decision based on the hidden states or embeddings during processing. However, we are the first to analyze and utilize the attention mechanism for skimming. Secondly, our work is based on the Transformer model (Vaswani et al. 2017), which has outperformed the recurrent type models on most NLP tasks.

Transformer with Input Reduction. Unlike the sequential processing of the recurrent models, the Transformer model calculates all the input sequence tokens in parallel. As such, skimming can be regarded as a reduction in sequence dimension. Power-BERT (Goyal et al. 2020) extracts input sequence at a token level while processing. During the fine-tuning process for downstream tasks, Goyal et al. proposes a soft-extraction layer to train the model jointly. Length-Adaptive Transformer (Kim and Cho 2020) further extends Power-BERT by forwarding the rejected tokens to the final linear layer. Funnel-Transformer (Dai et al. 2020) proposes a novel pyramid architecture with input sequence length dimension reduced gradually regardless of semantic clues. For tasks requiring full sequence length output, such as masked language modeling and extractive question answering, Funnel-Transformer up-samples at the input dimension to recover. DeFormer (Cao et al. 2020) propose to pre-process and cache the paragraphs at shallow layers and only concatenate with the question parts at deep layers. Universal Transformer (Dehghani et al. 2018) proposes a dynamic halting mechanism that determines the refinement steps for each token. Different from these works, Block-Skim utilizes attention information between question and token pairs and skims the input sequence at the block granularity accordingly. Moreover, Block-Skim does not modify the vanilla Transformer model, making it more applicable.

Efficient Transformer. There are also many efforts for designing efficient Transformers (Zhou et al. 2020; Wu et al. 2019; Tay et al. 2020). For example, researchers have applied well-studied compression methods to Transformers, such as pruning (Guo et al. 2020), quantization (Wang and Zhang 2020; Guo et al. 2022), distillation (Sanh et al. 2019), and weight sharing. Other efforts focus on dedicated efficient attention mechanism considering its quadratic complexity of sequence length (Kitaev, Kaiser, and Levskaya 2019; Beltagy, Peters, and Cohan 2020; Zaheer et al. 2020). Block-Skim is orthogonal to these techniques on the input dimension reduction. We demonstrate that Block-Skim is compatible with efficient Transformers with experimental results.

Attention-based Block Relevance Prediction

Token-Level Relevance Analysis

Transformer. The Transformer model adopts the multi-head self-attention mechanism and calculates hidden states for each position as an attention-based weighted sum of input hidden states. The weight vector is calculated by parameterized linear projection query Q and key K as Equation 1. Given a sequence of input embeddings, the output contextual embedding is composed by the input sequence with different attention at each position,

$$Attention(Q, K) = Softmax(QK^T/\sqrt{d_k}), \qquad (1)$$

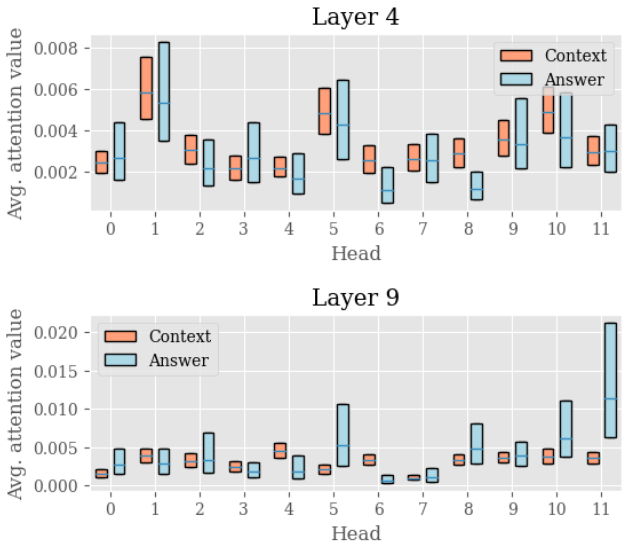

Figure 2: Attention weight value distribution comparison on the answer and irrelevant tokens. The attention heatmaps are profiled on the development set of SQuAD dataset with a $\text{BERT}_{\text{base}}$ model with 12 layers and 12 attention heads per layer. The full results are shown in appendix.

where Q, K are query and key matrix of input embeddings, d_k is the length of a query or key vector. As such, the attention weight feature map is often visualized as a heatmap demonstrating the information gathering relationship along tokens (Kovaleva et al. 2019). The model exploits multiple parallel groups of such attention weights, a.k.a. attention heads, for attending to information at different positions.

Extractive QA is one of the ultimate downstream tasks in the NLP. Given a text document and a question about the context, the answer is a contiguous span of the text. To predict the start and end position of the input context given a question, the embedding of each certain token is processed for all the layers in the Transformer encoder model. In many end-to-end open-domain QA systems, information retrieval is the preceding step at the coarse-grained passage or paragraph level for filtering out irrelevant passages. With the characteristic of the extractive QA problem where answer spans are part of the passage, our question is that whether we can apply a similar filtering technique at fine-grained granularity during the Transformer model inference.

In this work, we propose to augment the attention mechanism with the ability to predict the relevance of contextual tokens without modifying the original Transformer model. Prior work (Goyal et al. 2020) shows that attention strength is a good indicator for answer tokens. However, we analyze the attention weight distribution of a trained $\text{BERT}_{\text{base}}$ model trained with SQuAD (Rajpurkar et al. 2016) dataset and find that the attention weights of multi-head attention only have noticeably patterns at the late layers.

Fig. 2 compares the attention weights at Layer 4 and 9 in the trained $\text{BERT}_{\text{base}}$ model. The tokens are classified to answer tokens or irrelevant tokens with the labels from the dataset. At late layers like Layer 9, the attention weights of answer tokens are significantly larger than those of irrelevant tokens. However, at early layers like Layer 4, the attention weight strength is indistinguishable for answer tokens and irrelevant tokens. For a better latency reduction, it is desirable to find irrelevant tokens as early as possible. However, using the attention weight value as the relevance criterion could be problematic at early layers.

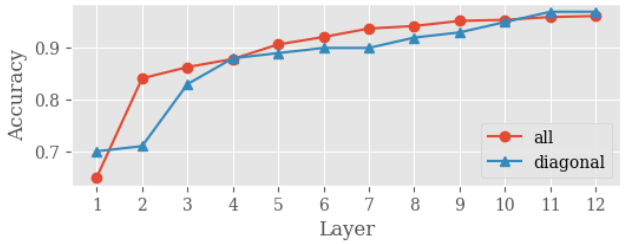

Figure 3: Accuracy of CNN model predicting whether a block contains answer with attention weight as input. The CNN is feed with either an all attention weight heatmap or only the diagonal block region.

CNN Based Block Relevance Prediction

Given the complex pattern of attention weights, we propose to use a CNN-based feature extractor to process the attention heatmaps as input image channels and predict the relevance of each token. To amortize the processing overhead, we split the input sequence $X = (x_0, x_1, ..., x_i)$ into i/k exclusive blocks $block_j = (x_{j\times k}, x_{j\times k+1}..., x_{j\times k+(k-1)})$, where k is the block size, i.e. tokens included in the continuous input span. The relevance of a block is defined as whether it contains the exact final answer. As such, our goal is to figure out the blocks' relevance and skim the irrelevant ones during Transformer inference.

Fig. 4 shows the details of how we extract the attention information from the Transformer and feed them into the CNN model. In the CNN module, we use two 3×3 convolution and one 1×1 convolution, all of which use the `ReLU` operation (Hahnloser and Seung 2001) as the activation function. We insert a 2×2 average pooling layer for the first two 3×3 convolutional layers to reduce the feature map size. In addition, we also use two batch normalization layers (Ioffe and Szegedy 2015) to improve the prediction accuracy. To locate the answer context blocks, we use a linear classification layer to calculate the score for each block. The module outputs a block-level prediction mask that corresponds to the relevance of a block of input tokens to the question.

This model is trained with all attention heatmap profiled from the same set of heatmap data as described before. The prediction accuracy is shown as Fig. 3. In general, the model achieves decent accuracy demonstrating that a CNN model is capable to extract the attending behavior information and locate the answer. Intuitively, the CNN models with higher layer attention heatmaps have better performance. It suggests that the backbone model becomes more convinced on question answering when it gets deeper.

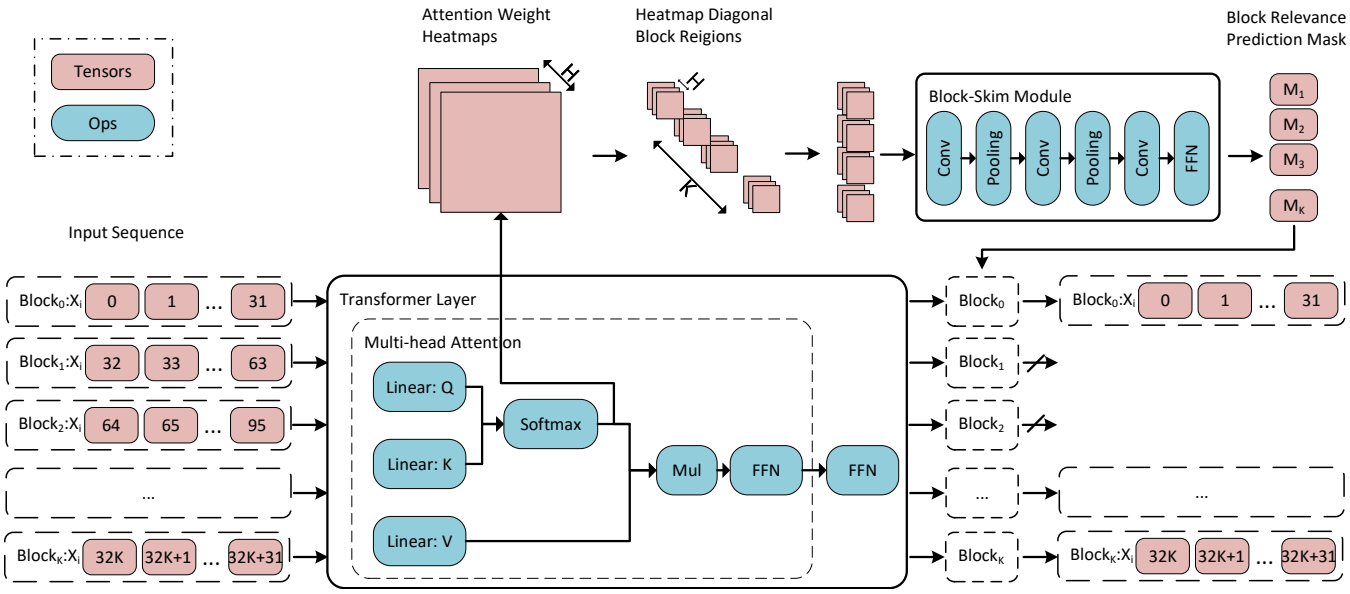

Figure 4: The overall schematic of Block-Skim and the architecture of the CNN model. Here we take block size 32 as example. The total number of blocks and attention heads are K and H. We only show the main operations for simplicity.

Simplifying CNN Predictor with Diagonal Attention

The above method of feeding the whole attention feature map to the CNN predictor has a major problem, which is the predictor needs to deal with the variable size of the attention feature map. As such, we simplify the input to the CNN model with only attention from its diagonal region. In specific, we only feed the diagonal heat-map region as the input representation for each input sequence block, as expressed in Fig. 4.

Our **hypothesis** is that the diagonal region of the attention heat map contains sufficient information to identify the block relevance. Because previous works (Clark et al. 2019; Guan et al. 2020) show that the attention mechanism has several fixed patterns, that is, diagonal, stride, block, or dense types. And all of these patterns can be easily recognized with only the diagonal region.

Similarly, we optimize CNN models with reduced heatmap and the result is shown as Fig. 3. As we can see, the models achieve similar prediction accuracy compared with using a whole attention weight heatmap. The result justifies our hypothesis that it is possible to use the diagonal information from attention heatmaps to predict the answer relevance. By doing so, the computation complexity is also reduced dramatically as the input size is much smaller.

The above finding confirms our hypothesis that the diagonal attention weight indeed carries information for figuring out answer positions. This motivates us to utilize such attention information to narrow the possible answer position along with the processing of the input sequence. In the next section, we introduce our design that uses a plug-and-play end-to-end learning module to extract useful information from the attention weights for skimming decisions.

Transformer with Block-Skim

The previous section shows the feasibility of using the attention weights to predict the relevance of token blocks. However, naively using the predictor can lead to significant degradation of the QA task accuracy. Because the block relevance predictor is only trained with the answer labels, it could fail in the multi-hop QA task, which requires information beyond the answer labels. To solve this problem, we propose an end-to-end multi-objective joint training paradigm. Then during inference time, the prediction of the Block-Skim model is augmented to filter the input sequence for acceleration. This causes a mismatch between training and inference models. However, skimming blocks during training makes joint training unstable. And our experimental results demonstrate that this mismatch is negligible. We give a detailed demonstration of the proposed joint training paradigm and inference process as follows.

Single-Task Multi-Objective Joint Training

Following the previous experiments, we append the aforementioned CNN models to each layer to predict the blocks' relevance and optimize them together with the backbone Transformer model. As such, there are two types of classifiers in the model augmented with Block-Skim module. The first is the original QA classifier at the last layer and the second is the block-level relevance classifier at each layer. These two classifiers optimize the same downstream task of predicting the answer position with an identical target label. However, they are fed with a different type of loss objectives, that is, the QA objective with Transformer output embeddings and the Block-Skim objective with attention weights. We jointly train these classifiers so that the training objective is to minimize the sum of all classifiers' losses.

The loss function of each block-level classifier is calculated as the cross-entropy loss against the ground truth label whether a block contains answer tokens or not. Equation 2 gives the formal definition. The total loss of the block-level classifier $\mathcal{L}_{BlockSkim}$ is the sum of all blocks that only contain passage tokens. The reason is that we only want to throw away blocks with irrelevant passage tokens instead of questions. Blocks that have question tokens are not used in the training process.

$$\mathcal{L}_{BlockSkim} = \sum_{m_i \in \{\texttt{passage blocks}\}} CELoss(m_i, y_i)$$
$$y_i = \begin{cases} 1 & \text{, block i has answer tokens} \\ 0 & \text{, block i has no answer tokens} \end{cases} \tag{2}$$

For the calculation of the final total loss $\mathcal{L}_{total}$, we introduce two hyper-parameters in Equation 3. We first use a harmony coefficient α so that different models and settings could adjust the ratio between the QA loss and block-level relevance classifier loss. It is decided by grid search on the development set. We then use the balance factor β to adjust the loss from positive and negative relevance blocks because there are typically many more blocks that contain no answer tokens (i.e., negative bocks) than the blocks that do contain answer tokens (i.e., positive bocks). This hyper-parameter selection will be explained in detail in experiments setup.

$$\mathcal{L}_{total} = \mathcal{L}_{QA} + \alpha \sum_{i_{th} \text{ layer}} (\beta \mathcal{L}^{i,y=1}_{BlockSkim} + \mathcal{L}^{i,y=0}_{BlockSkim}) \tag{3}$$

Our Block-Skim is a convenient plugin module owing to the following two reasons. First, it does not affect the backbone model calculation, because it only regularizes the attention value distribution with extra parameters to the backbone model. In other words, a model trained with Block-Skim can be used with it removed. Second, the introduced Block-Skim objective neither needs an extra training signal nor reduces the QA accuracy. In fact, we will show that the extra gradient signal feeding to the attention improves the original QA accuracy.

Multi-hop QA. Our joint training approach can also address the challenge in the multi-hop QA tasks (Yang et al. 2018), where deriving answers requires multiple pieces of evidence and reasoning. Although the block relevance prediction only uses the answer label signal, the original QA task ensures that evidence needs to be kept. In other words, the evidence reasoning information is encoded implicitly in the contextual embeddings. To illustrate such a point, we perform an ablation study that incorporates the evidence label in the Block-Skim predictor training. The predictor accuracy does not improve with the additional evidence label, which confirms the effectiveness of our single-task multi-objective joint training.

Inference with Block-Skim

We now describe how to use the Block-Skim to accelerate the QA task inference. Although we add the block-level relevance classification loss in the joint training process, we do not actually throw away any blocks because it can skip answer blocks and the QA task training becomes unstable. However, we only augment block reduction with the Block-Skim module during the inference for saving computation and avoiding heavy changes to the underlying Transformer. During inference computation, we split the input sequence by the block granularity, which is a hyper-parameter in our model. The model skips a set of blocks according to the skimming module results for the following layers. With those design features, Block-Skim works as an add-on component to the original Transformer model and is compatible with many Transformer variant models as well as model compression methods.

We provide an analytical model to demonstrate the latency speedup potential of Block-Skim. Suppose that we insert the Block-Skim module to a vanilla model with the total L layers, and a portion of m_i blocks remains for the following layers after layer i. The ideal processing complexity of one token for one Transformer layer is noted as T_{layer}. Here we make an approximation that the computation complexity is linear to the sequence length N. This is a conservative approximation because the attention mechanism is $O(N^2)$. The performance speedup is formulated by Equation 4 if we ignore the computation overhead of Block-Skim. In fact, the computation of a single Block-Skim module is smaller than Transformer layers for 100 times. For example, when $\sum_{m_k \in \{\texttt{passage blocks}\}} m_k = 0.9$, it means 10% of tokens are skimmed each layer. This skimming decision will result in a total speedup ratio of $1.86\times$.

$$\begin{aligned} Speedup &= \frac{T_{Vanilla}}{T_{Block-Skim}} \\ &= \frac{L \cdot N \cdot T_{layer}}{\sum_{i=0}^{L} (\prod_{j=0}^{i} \sum_{m_{j,k} \in \{layer_j\}} m_{j,k} \cdot N) \cdot T_{layer}} \\ &= \frac{L}{\sum_{i=0}^{L} \prod_{j=0}^{i} \sum_{m_{j,k} \in \{layer_j\}} m_{j,k}} \end{aligned} \tag{4}$$

Evaluation

Experimental Setup

Dataset. We evaluate our method on 6 extractive QA datasets, including SQuAD 1.1 (Rajpurkar et al. 2016), Natural Questions (Kwiatkowski et al. 2019), TriviaQA (Joshi et al. 2017), NewsQA (Trischler et al. 2016), SearchQA(Dunn et al. 2017) and HotpotQA (Yang et al. 2018). HotpotQA provides questions that require multi-hop reasoning to answer with supporting facts. The diversity of these datasets such as various passage lengths and different document sources lets us evaluate the general applicability of the proposed method.

Model. We follow the setting of the BERT model to use the structure of the Transformer encoder and a linear classification layer for all the datasets. As previously explained, Block-Skim works as an add-on module to the vanilla Transformer, and therefore is generally applicable to all Transformer-based models, as well as model compression methods. To illustrate this point, we apply the Block-Skim method to two BERT models with different size settings. We evaluate the base setting with 12 heads and 12 layers, as well

Datasets	SQuAD		HotpotQA		NewsQA		NaturalQuestions		TriviaQA		SearchQA		Avg.	
	F1	Speedup	F1	Speedup	F1	Speedup	F1	Speedup	F1	Speedup	F1	Speedup	F1	Speedup
Balance Factor	20		20		30		30		100		150		-	
Vanilla BERT	88.32	1×	74.39	1×	66.57	1×	78.85	1×	72.61	1×	79.93	1×	76.78	1×
Block-Skim Training	88.92	1×	74.88	1×	67.76	1×	78.98	1×	73.29	1×	80.32	1×	77.36	1×
Block-Skim Inference	88.52	3.01×	74.47	2.28×	65.14	2.53×	78.48	2.56×	72.80	1.81×	79.84	3.17×	76.54	2.56×
Deformer	87.2	3.1×	-	-	-	-	-	-	-	-	-	-	-	-
Length-Adaptive Transformer	88.7	2.22×	-	-	-	-	-	-	-	-	-	-	-	-

Table 1: Validation F1 score and FLOPs speedup of $\text{BERT}_{\text{base}}$ model evaluated on different QA datasets. The balance factor is determined by calculating the block number distribution on training set.

as the large setting with 24 layers and 16 heads as described in prior work (Devlin et al. 2018).

Model Compression Methods. We conduct the following model compression methods on $\text{BERT}_{\text{base}}$ models to demonstrate the compatibility of our Block-Skim.

- **Distillation.** Knowledge distillation uses a teacher model to transfer the knowledge to a smaller student model. Here we adopt DistilBERT (Sanh et al. 2019) setting to distill a 6-layer model from the $\text{BERT}_{\text{base}}$ model.
- **Weight Sharing.** By sharing weight parameters among layers, the amount of weight parameters reduces. Note that weight sharing does not impact the computation FLOPs (floating-point operations). We evaluate Block-Skim on ALBERT (Lan et al. 2019) that shares weight parameters among all layers.
- **Pruning.** Instead of conventional weight pruning techniques, we evaluate head pruning (Michel, Levy, and Neubig 2019) that is specific to the attention mechanism in Transformer models. The pruning of attention heads reduces the input feature size to the Block-Skim module. We prune 50% of attention heads based on the attention head importance criterion introduced in prior work (Michel, Levy, and Neubig 2019).

Input Dimension Reduction Baselines. We also compare with input dimension reduction methods Deformer and Length-Adaptive Transformer. Deformer(Cao et al. 2020) pre-process and caches the context paragraphs at early layers to reduce the actual inference sequence length. Length-Adaptive Transformer (Kim and Cho 2020) is a successive version of Power-BERT which forwards the tokens rejected to the final layer by attention strength.

Training Setting. We implement the proposed method based on open-sourced library from Wolf et al. (2019)[1]. For each baseline model, we use the released pre-trained checkpoints [2]. We follow the training setting used by Devlin et al. (2018) and Liu et al. (2019) to perform the fine-tuning on the above extractive QA datasets. We initialize the learning rate to $3e-5$ for BERT models and $5e-5$ for ALBERT with a linear learning rate scheduler. For SQuAD dataset, we apply batch size 16 and maximum sequence length 384.

[1]The source code is available at https://github.com/ChandlerGuan/blockskim.

[2]We use pre-trained language model checkpoints released from https://huggingface.co/models.

Seed		0	1	2	3	4	Avg.	Std.
Vanilla	EM	80.95	81.08	80.98	81.06	80.96	81.00	0.06
	F1	88.32	88.44	88.59	88.44	88.42	88.44	0.10
Block-Skim	EM	81.52	81.25	81.24	81.51	81.84	81.47	0.25
	F1	88.92	88.48	88.66	88.76	88.99	88.76	0.20

Table 2: Results of multiple runs under same training and hyper-parameter setting with different random seeds.

And for the other datasets, we apply batch size 32 and maximum sequence length 512. We perform all the experiments reported with random seed 42. We train a baseline model and Block-Skim model with the same setting for two epochs and report accuracies from MRQA task benchmark for comparison. We use four V100 GPUs with 32 GB memory for the training experiments.

The balance factor β is determined by block sample numbers and reported in Tbl. 1. The harmony factor α is 0.01 for ALBERT and 0.1 for all the other models we used. It is determined by hyper-parameter grid search from $1e-3$ to 10 with a step of $\times 10$.

We use the inference FLOPs as a general measurement of the model computational complexity on all platforms. We use TorchProfile(Liu 2020) to calculate the FLOPs for each model and normalize the results as a ratio to $\text{BERT}_{\text{base}}$.

Joint Training Results

We first evaluate Block-Skim joint training effect to the QA task by comparing $\text{BERT}_{\text{base}}$ models and their variants with Block-Skim augmented. In their Block-Skim versions, the Block-Skim modules only participate in the training process and are removed in the inference task. Tbl. 1 shows the result on multiple QA datasets. Block-Skim outperforms the baseline training objective on all datasets evaluated and exceeds with 0.58% F1 score on average. This suggests that the Block-Skim objective is consistent with the QA objective and even improves its accuracy. The results show the wide applicability of our method to different datasets with varying difficulty and complexity.

We further show the robustness when using the Block-Skim joint training as an add-on module. Tbl. 2 shows the result of multiple runs using the identical optimization setting with different random seeds. By introducing the Block-Skim loss in Training, the QA accuracy of the backbone model is improved for 0.4 on exact match and 0.32 on F1 score.

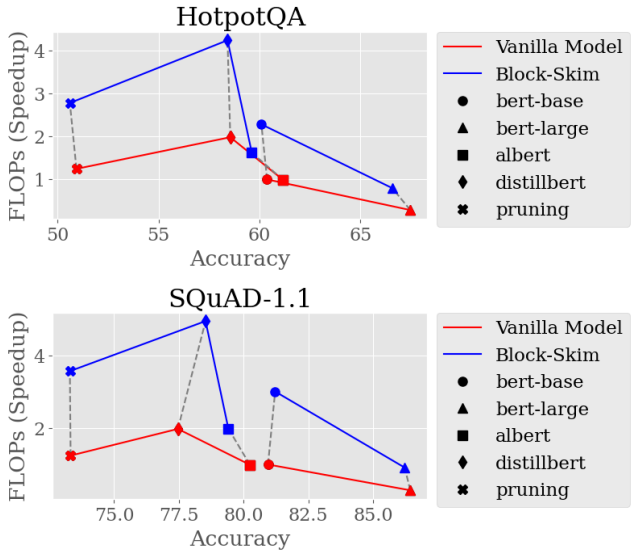

Figure 5: FLOPs speedup of different models and model compression methods with Block-Skim on SQuAD and HotpotQA datasets. The FLOPs are normalized to $\text{BERT}_{\text{base}}$ result of $48.32G$ FLOPs result. The results of vanilla models with different size, model compression algorithms and Block-Skim augmented methods are grouped together.

And triggering Block-Skim always surpasses the backbone model with the same setting. This is because the extra training objective provides direct gradient signals to the attention mechanism and regularizes its value distribution.

Inference Speedup Results

Results on Various Datasets. The FLOPs speedup result normalized to $\text{BERT}_{\text{base}}$ model is demonstrated in Tbl. 1. Block-Skim achieves $2.59\times$ speedup on average with a minor accuracy degradation of 0.23 on different datasets evaluated. On the multi-hop QA dataset HotpotQA, our method also achieves 2.28 times speedup. The results show that the proposed Block-Skim method is capable to identify the semantic redundancy with attention information.

Comparison to Vanilla BERT Baseline. Block-Skim improves the $\text{BERT}_{\text{base}}$ model inference latency by $3.1\times$ and $2.4\times$ respectively on SQuAD and HotpotQA datasets. When treating the model size settings of vanilla BERT model as a trade-off between accuracy and complexity, Block-Skim improves this trade-off by a margin. As shown in Fig. 5 our method accelerate $\text{BERT}_{\text{large}}$ as fast as the vanilla $\text{BERT}_{\text{base}}$ model but with a much higher accuracy. In specific, the latency of vanilla $\text{BERT}_{\text{large}}$ model is $3.47\times$ of $\text{BERT}_{\text{base}}$, and our method reduces the gap to $1.09\times$ on SQuAD dataset, which translates to the $3.18\times$ speedup.

Compatibility to Model Compression Methods. We compare the Block-Skim's compatibility to other model compression methods with Fig. 5. These model compression methods trade accuracy for computation complexity to different extents. For example, distilling 12-layer $\text{BERT}_{\text{base}}$ model to 6 layers results in a 2% accuracy decrease and 2 times speedup. With Block-Skim method appended to this model, the methods can be further accelerated with no or minor accuracy loss. Specifically, using Block-Skim with DistilBERT achieves $5\times$ speedup compared to the vanilla BERT model. And even with head-pruning reducing the attention information, Block-Skim is also compatible and achieves over $2\times$ speedup. Even though still compatible, Block-Skim gets less acceleration on ALBERT models. We suggest that sharing parameters of attention mechanism makes it harder to optimize with extra Block-Skim objective. As the proposed Block-Skim method aims to reduce the input sequence dimension semantic redundancy, it is compatible to these model compression methods focusing on model redundancy theoretically. By designing Block-Skim not to modify the backbone model, our method is generally applicable to these algorithms as well as other model pruning methods (Guo et al. 2020; Qiu et al. 2019; Gan et al. 2020).

Block-Skim achieves close speedup with less accuracy degradation compared to Deformer and more speedup with similar accuracy degradation compared to Length-Adaptive Transformer on SQuAD-1.1 dataset. This suggests Block-Skim captures the runtime semantic redundancy better. Although this also makes it only applicable to QA tasks.

ID	Description	Update Transformer	Skim Training	Block-Skim Module	Block Size	QA EM	QA F1
	SQuAD						
1	Baseline	✓	-	-	-	80.92	88.32
2	Block-Skim	✓		✓	32	81.52	88.92
3	Freeze Transformer			✓	32	80.92	88.32
4	Skim Traning	✓	✓	✓	32	79.27	86.83
5	Block Size 1	✓		✓	1	81.22	88.60
6	Block Size 8	✓		✓	8	81.25	88.63
7	Block Size 16	✓		✓	16	81.35	88.75
8	Block Size 64	✓		✓	64	81.39	88.65
9	Block Size 128	✓		✓	128	80.90	88.33
	HotpotQA						
10	Baseline	✓	-	-	-	60.37	74.39
11	Block-Skim	✓		✓	32	60.54	74.88
12	Evidence Loss	✓		✓	32	60.78	74.85

Table 3: Ablation studies of the Block-Skim components with $\text{BERT}_{\text{base}}$ backbone model on SQuAD and HotpotQA datasets.

Ablation Study

We design a series ablation experiment of components in Block-Skim to study their individual effect. The experiments are performed based on the same setting. We report the detailed results in Tbl. 3, and summarize the key findings as follows.

ID-3. Instead of joint training, we perform a two-step training. We first perform the fine-tuning for the QA task. We then perform the Block-Skim training with the baseline QA model frozen. In other words, we only use the Block-Skim objective and only update the weights in the Block-Skim modules. Therefore, the QA accuracy remains the same as the baseline model, which is lower than the joint training (ID-3). Meanwhile, the Block-Skim classifier also has a lower accuracy than the joint training especially at layer 6.

ID-4 We skim blocks during the joint Block-Skim QA training process. Because the mis-skimmed blocks may confuse

the QA optimization, it leads to a considerable accuracy loss.

ID-5-ID-9. We study the impact of different block sizes. Specifically, when the block size is 1, it is equivalent to skim at the token granularity. Our experimental result shows that the accuracy of Block-Skim classifier is better when the block size is larger. On the other hand, a larger block size also leads to less number of blocks and therefore the performance speedup becomes limited. To this end, we choose the block size of 32 as a design sweet spot.

ID-11-ID-12. We evaluate the applicability of Block-Skim to multi-hop QA task with HotpotQA dataset. As introduced in , we add supporting facts (i.e., evidence) for each question to the Block-Skim objective in the ID-12 experiment by labelling evidence blocks to 1 in Eq.2 for the skim predictor modules. This leads to a higher QA accuracy. But the average accuracy of skim predictors at all layers is worse, which is 86.08% compared to 92.67%. This ablation experiment shows that our single-task multi-objective joint training is already able to capture the evidence information, rendering explicitly adding it to the training unnecessary.

Conclusion

In this work, we propose a plug-and-play Block-Skim module to Transformer and its variants for efficient QA processing. We empirically demonstrate that the attention mechanism can provide instructive information for locating the answer span. Leveraging this insight, we propose to learn the attention in a supervised manner, which terminates irrelevant blocks at early layers, significantly reducing the computations. Besides, the proposed Block-Skim training objective provides attention mechanism with extra learning signal and improves QA accuracy on all datasets and models we evaluated. With the use of Block-Skim module, such distinction is strengthened in a supervised fashion. This idea may be also applicable to other tasks and architectures.

Appendix Attention Distribution

We show the full results of attention weight value distribution discussed in Fig.2. Fig. 6 shows that deeper layers have more distinguishable patterns.

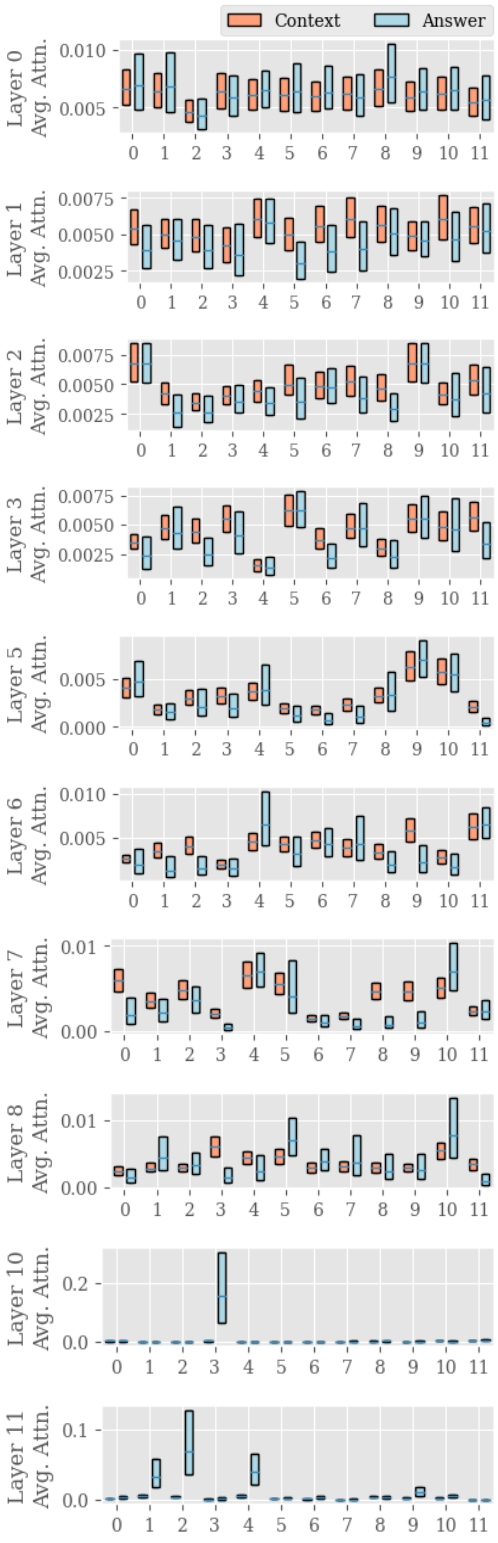

Figure 6: Attention weight value distribution comparison on the answer and irrelevant tokens. The attention heatmaps are profiled on the development set of SQuAD dataset with a $\text{BERT}_{\text{base}}$ model with 12 layers and 12 attention heads per layer.

Acknowledgements

This work was supported by the National Key R&D Program of China under Grant 2019YFF0302600, and the National Natural Science Foundation of China (NSFC) grant (62072297, 62106143, and 61832006). We would like to thank the anonymous reviewers for their thoughtful comments and constructive suggestions. Zhouhan Lin is also supported by Shanghai Pujiang Program. We also thank Yuxian Qiu and Kexin Li with whom we have inspiring discussions on the evaluation experiment design. Finally, we thank Zhihui Zhang for helping the presentation and visualization of experimental results. Jingwen Leng and Zhouhan Lin are the corresponding authors of this paper.

References

Beltagy, I.; Peters, M. E.; and Cohan, A. 2020. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*.

Campos, V.; Jou, B.; Giró-i-Nieto, X.; Torres, J.; and Chang, S. 2017. Skip RNN: Learning to Skip State Updates in Recurrent Neural Networks. *CoRR*, abs/1708.06834.

Cao, Q.; Trivedi, H.; Balasubramanian, A.; and Balasubramanian, N. 2020. Deformer: Decomposing pre-trained transformers for faster question answering. *arXiv preprint arXiv:2005.00697*.

Clark, K.; Khandelwal, U.; Levy, O.; and Manning, C. D. 2019. What Does BERT Look at? An Analysis of BERT's Attention. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, 276–286.

Dai, Z.; Lai, G.; Yang, Y.; and Le, Q. V. 2020. Funnel-Transformer: Filtering out Sequential Redundancy for Efficient Language Processing. *arXiv preprint arXiv:2006.03236*.

Dehghani, M.; Gouws, S.; Vinyals, O.; Uszkoreit, J.; and Kaiser, L. 2018. Universal Transformers. In *International Conference on Learning Representations*.

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

Dunn, M.; Sagun, L.; Higgins, M.; Guney, V. U.; Cirik, V.; and Cho, K. 2017. Searchqa: A new q&a dataset augmented with context from a search engine. *arXiv preprint arXiv:1704.05179*.

Gan, Y.; Qiu, Y.; Leng, J.; Guo, M.; and Zhu, Y. 2020. Ptolemy: Architecture Support for Robust Deep Learning. In *2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*.

Goyal, S.; Choudhary, A. R.; Chakaravarthy, V.; ManishRaje, S.; Sabharwal, Y.; and Verma, A. 2020. PoWER-BERT: Accelerating BERT inference for Classification Tasks. *arXiv preprint arXiv:2001.08950*.

Guan, Y.; Leng, J.; Li, C.; Chen, Q.; and Guo, M. 2020. How Far Does BERT Look At: Distance-based Clustering and Analysis of BERT ′ s Attention. *arXiv preprint arXiv:2011.00943*.

Guo, C.; Hsueh, B. Y.; Leng, J.; Qiu, Y.; Guan, Y.; Wang, Z.; Jia, X.; Li, X.; Guo, M.; and Zhu, Y. 2020. Accelerating Sparse DNN Models without Hardware-Support via Tile-Wise Sparsity. *arXiv preprint arXiv:2008.13006*.

Guo, C.; Qiu, Y.; Leng, J.; Gao, X.; Zhang, C.; Liu, Y.; Yang, F.; Zhu, Y.; and Guo, M. 2022. SQuant: On-the-Fly Data-Free Quantization via Diagonal Hessian Approximation. In *International Conference on Learning Representations*.

Hahnloser, R. H.; and Seung, H. S. 2001. Permitted and forbidden sets in symmetric threshold-linear networks. In *Advances in neural information processing systems*, 217–223.

Hansen, C.; Hansen, C.; Alstrup, S.; Simonsen, J. G.; and Lioma, C. 2018. Neural Speed Reading with Structural-Jump-LSTM. In *International Conference on Learning Representations*.

Hochreiter, S.; and Schmidhuber, J. 1997. Long short-term memory. *Neural computation*, 9(8): 1735–1780.

Huang, Z.; Xu, S.; Hu, M.; Wang, X.; Qiu, J.; Fu, Y.; Zhao, Y.; Peng, Y.; and Wang, C. 2020. Recent Trends in Deep Learning Based Open-Domain Textual Question Answering Systems. *IEEE Access*, 8: 94341–94356.

Ioffe, S.; and Szegedy, C. 2015. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *International Conference on Machine Learning*, 448–456.

Joshi, M.; Choi, E.; Weld, D. S.; and Zettlemoyer, L. 2017. TriviaQA: A Large Scale Distantly Supervised Challenge Dataset for Reading Comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 1601–1611.

Kim, G.; and Cho, K. 2020. Length-Adaptive Transformer: Train Once with Length Drop, Use Anytime with Search. *arXiv preprint arXiv:2010.07003*.

Kitaev, N.; Kaiser, L.; and Levskaya, A. 2019. Reformer: The Efficient Transformer. In *International Conference on Learning Representations*.

Kovaleva, O.; Romanov, A.; Rogers, A.; and Rumshisky, A. 2019. Revealing the dark secrets of BERT. *arXiv preprint arXiv:1908.08593*.

Kwiatkowski, T.; Palomaki, J.; Redfield, O.; Collins, M.; Parikh, A.; Alberti, C.; Epstein, D.; Polosukhin, I.; Devlin, J.; Lee, K.; et al. 2019. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7: 453–466.

Lan, Z.; Chen, M.; Goodman, S.; Gimpel, K.; Sharma, P.; and Soricut, R. 2019. ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. In *International Conference on Learning Representations*.

Lin, Z.; Feng, M.; Santos, C. N. d.; Yu, M.; Xiang, B.; Zhou, B.; and Bengio, Y. 2017. A structured self-attentive sentence embedding. *arXiv preprint arXiv:1703.03130*.

Liu, Y.; Ott, M.; Goyal, N.; Du, J.; Joshi, M.; Chen, D.; Levy, O.; Lewis, M.; Zettlemoyer, L.; and Stoyanov, V. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach.

Liu, Z. 2020. Torchprofile. https://github.com/zhijian-liu/torchprofile/.

Michel, P.; Levy, O.; and Neubig, G. 2019. Are Sixteen Heads Really Better than One? *Advances in Neural Information Processing Systems*, 32: 14014–14024.

Qiu, Y.; Leng, J.; Guo, C.; Chen, Q.; Li, C.; Guo, M.; and Zhu, Y. 2019. Adversarial Defense Through Network Profiling Based Path Extraction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Rajpurkar, P.; Zhang, J.; Lopyrev, K.; and Liang, P. 2016. SQuAD: 100, 000+ Questions for Machine Comprehension of Text. In *EMNLP*.

Rumelhart, D. E.; Hinton, G. E.; and Williams, R. J. 1986. Learning representations by back-propagating errors. *nature*, 323(6088): 533–536.

Sanh, V.; Debut, L.; Chaumond, J.; and Wolf, T. 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

Seo, M.; Min, S.; Farhadi, A.; and Hajishirzi, H. 2018. Neural Speed Reading via Skim-RNN. In *International Conference on Learning Representations*.

Tay, Y.; Dehghani, M.; Bahri, D.; and Metzler, D. 2020. Efficient Transformers: A Survey. *arXiv e-prints*, arXiv–2009.

Trischler, A.; Wang, T.; Yuan, X.; Harris, J.; Sordoni, A.; Bachman, P.; and Suleman, K. 2016. NEWSQA: A MACHINE COMPREHENSION DATASET.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. In *Advances in neural information processing systems*, 5998–6008.

Wang, C.; and Zhang, X. 2020. Q-BERT: A BERT-based Framework for Computing SPARQL Similarity in Natural Language. In *Companion Proceedings of the Web Conference 2020*, 65–66.

Wolf, T.; Debut, L.; Sanh, V.; Chaumond, J.; Delangue, C.; Moi, A.; Cistac, P.; Rault, T.; Louf, R.; Funtowicz, M.; Davison, J.; Shleifer, S.; von Platen, P.; Ma, C.; Jernite, Y.; Plu, J.; Xu, C.; Scao, T. L.; Gugger, S.; Drame, M.; Lhoest, Q.; and Rush, A. M. 2019. HuggingFace's Transformers: State-of-the-art Natural Language Processing. *ArXiv*, abs/1910.03771.

Wu, Z.; Liu, Z.; Lin, J.; Lin, Y.; and Han, S. 2019. Lite Transformer with Long-Short Range Attention. In *International Conference on Learning Representations*.

Yang, Z.; Qi, P.; Zhang, S.; Bengio, Y.; Cohen, W. W.; Salakhutdinov, R.; and Manning, C. D. 2018. HotpotQA: A Dataset for Diverse, Explainable Multi-hop Question Answering. In *EMNLP*.

Yu, A. W.; Lee, H.; and Le, Q. 2017. Learning to Skim Text. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Zaheer, M.; Guruganesh, G.; Dubey, A.; Ainslie, J.; Alberti, C.; Ontanon, S.; Pham, P.; Ravula, A.; Wang, Q.; Yang, L.; et al. 2020. Big bird: Transformers for longer sequences. *arXiv preprint arXiv:2007.14062*.

Zhou, W.; Xu, C.; Ge, T.; McAuley, J.; Xu, K.; and Wei, F. 2020. BERT Loses Patience: Fast and Robust Inference with Early Exit. *arXiv*, arXiv–2006.