Cross-Thought for Sentence Encoder Pre-training

Shuohang Wang[1]   **Yuwei Fang**[1]   **Siqi Sun**[1]   **Zhe Gan**[1]   **Yu Cheng**[1]
Jing Jiang[2]   **Jingjing Liu**[1]

[1]Microsoft Dynamics 365 AI Research, [2]Singapore Management University
{shuowa, yuwfan, siqi.sun, zhe.gan, yu.cheng, jingjl}@microsoft.com
{jingjiang}@smu.edu.com

Abstract

In this paper, we propose Cross-Thought, a novel approach to pre-training sequence encoder, which is instrumental in building reusable sequence embeddings for large-scale NLP tasks such as question answering. Instead of using the original signals of full sentences, we train a Transformer-based sequence encoder over a large set of short sequences, which allows the model to automatically select the most useful information for predicting masked words. Experiments on question answering and textual entailment tasks demonstrate that our pre-trained encoder can outperform state-of-the-art encoders trained with continuous sentence signals as well as traditional masked language modeling baselines. Our proposed approach also achieves new state of the art on HotpotQA (full-wiki setting) by improving intermediate information retrieval performance.[1]

1 Introduction

Encoding sentences into embeddings (Kiros et al., 2015; Subramanian et al., 2018; Reimers and Gurevych, 2019) is a critical step in many Natural Language Processing (NLP) tasks. The benefit of using sentence embeddings is that the representations of all the encoded sentences can be reused on a chunk level (compared to word-level embeddings), which can significantly accelerate inference speed. For example, when used in question answering (QA), it can significantly shorten inference time with all the embeddings of candidate paragraphs pre-cached into memory and only matched with the question embedding during inference.

There have been several models specifically designed to pre-train sentence encoders with large-scale unlabeled corpus. For example, Skip-thought (Kiros et al., 2015) uses encoded sentence

---

Figure 1: Example of short sequences that can leverage each other for pre-training sentence encoder.

embeddings to generate the next sentence (Figure 2(a)). Inverse Cloze Task (Lee et al., 2019) defines some pseudo labels to pre-train a sentence encoder (Figure 2(b)). However, pseudo labels may bear low accuracy, and rich linguistic information that can be well learned in generic language modeling is often lost in these unsupervised methods. In this paper, we propose a novel unsupervised approach that fully exploits the strength of language modeling for sentence encoder pre-training.

Popular pre-training tasks such as language modeling (Peters et al., 2018; Radford et al., 2018), masked language modeling (Devlin et al., 2019; Liu et al., 2019) and sequence generation (Dong et al., 2019; Lewis et al., 2019) are not directly applicable to sentence encoder training, because only the hidden state of the first token (a special token) (Reimers and Gurevych, 2019; Devlin et al., 2019) can be used as the sentence embedding, but no loss or gradient is specifically designed for the first special token, which renders sentence embeddings learned in such settings contain limited useful information.

Another limitation in existing masked language modeling methods (Devlin et al., 2019; Liu et al., 2019) is that they focus on long sequences (512 words), where masked tokens can be recovered by
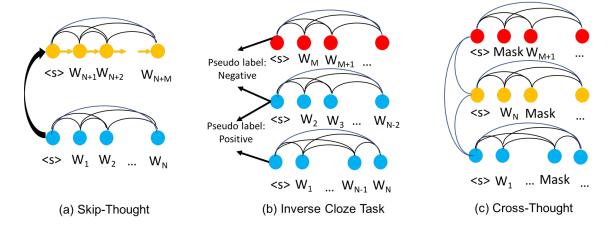
Figure 2: Structures of pre-training models for sentence encoder. (a) is a Seq2Seq model that generates the next sentence based on the embedding of the previous sentence. (b) is the classification/ranking model based on pre-defined pseudo labels. (c) is the structure of our model by using the sentence embedding from other sequences to generate the masked word.

considering the context within the same sequence. This is useful for word dependency learning within a sequence, but less effective for sentence embedding learning.

In this paper, we propose Cross-Thought, which segments input text into shorter sequences, where masked words in one sequence are less likely to be recovered based on the current sequence itself, but more relying on the embeddings of other surrounding sequences. For example, in Figure 1, the masked words "George Washington" and "United States" in the third sequence can only be correctly predicted by considering the context from the first sequence. Thus, instead of performing self-attention over all the words in all sentences, our proposed pre-training method enforces the model to learn from mutually-relevant sequences and automatically select the most relevant neighbors for masked words recovery.

The proposed Cross-Thought architecture is illustrated in Figure 2(c). Specifically, we pre-append each sequence with multiple special tokens, the final hidden states of which are used as the final sentence embedding. Then, we train multiple cross-sequence Transformers over the hidden states of different special tokens independently, to retrieve relevant sequence embeddings for masked words prediction. After pre-training, the attention weights in the cross-sequence Transformers can be directly applied to downstream tasks (e.g., in QA tasks, similarity scores between question and candidate answers can be ranked by their respective sentence embeddings).

Our contributions are summarized as follows. (i) We propose the Cross-Thought model to pre-train a sentence encoder with a novel pre-training task: recovering a masked short sequence by taking into consideration the embeddings of surrounding sequences. (ii) Our model can be easily finetuned on diverse downstream tasks. The attention weights of the pre-trained cross-sequence Transformers can also be directly used for ranking tasks. (iii) Our model achieves the best performance on multiple sequence-pair classification and answer-selection tasks, compared to state-of-the-art baselines. In addition, it further boosts the recall of information retrieval (IR) models in open-domain QA task, and achieves new state of the art on the HotpotQA benchmark (full-wiki setting).

2   Related Work

Sequence Encoder   Many studies have explored different ways to improve sequence embeddings. Huang et al. (2013) proposes deep structured semantic encoders for web search. Tan et al. (2015) uses LSTM as the encoder for non-factoid answer selection, and Tai et al. (2015) proposes tree-LSTM to compute semantic relatedness between sentences. Mou et al. (2016) also uses tree-based CNN as the encoder for textual entailment tasks. Cheng et al. (2016) proposes Long Short-Term Memory-Networks (LSTMN) for inferring the relation between sentences, and Lin et al. (2017) combines LSTM and self-attention mechanism to improve sentence embeddings. Multi-task learning (Subramanian et al., 2018; Cer et al., 2018) has

also been applied for training better sentence embeddings. Recently, in additional to supervised learning, models pre-trained with unsupervised methods begin to dominate the field.

Pre-training Several methods have been proposed to directly pre-train sentence embedding, such as Skip-thought (Kiros et al., 2015), FastSent (Hill et al., 2016), and Inverse Cloze Task (Lee et al., 2019). Although these methods can obtain better sentence embeddings in an unsupervised way, they cannot achieve state-of-the-art performance in downstream tasks even with further fine-tuning. More recently, Peters et al. (2018) proposes to pre-train LSTM with language modeling (LM) task, and Radford et al. (2018) pre-trains Transformer also with LM. Instead of sequentially generating words in a single direction, Devlin et al. (2019) proposes the masked language modeling task to pre-train bidirectional Transformer. Most recently, Guu et al. (2020); Lewis et al. (2020) propose to jointly train sentence-embedding-based information retriever and Transformer to re-construct documents. However, their methods are usually difficult to train with reinforcement learning methods involved, and need to periodically re-index the whole corpus such as Wikipedia. In this paper, to pre-train sentence encoder, we propose a new model Cross-Thought to recover the masked information across sequences. We make use of the heuristics that nearby sequences in the document contain the most important information to recover the masked words. Therefore, the challenging retrieval part can be replaced by soft-attention mechanism, making our model much easier to train.

3 Cross-Thought

In this section, we introduce our proposed pre-training model Cross-Thought, and describe how to finetune the pre-trained model on downstream tasks. Specifically, most parameters in downstream tasks can be initialized by the pre-trained Cross-Thought, and for certain tasks (e.g., ranking) the attention weights across sequences can be directly used without additional parameters (Figure 3).

3.1 Pre-training Data Construction

Our pre-training task is inspired by Masked Language Modeling (Devlin et al., 2019; Liu et al., 2019), and the key difference is the way to construct sequences for pre-training. As our goal is sentence embedding learning, the pre-training task is designed to encourage the model to recover masked words based on sentence-level global context from other sequences, instead of word-level local context within the same sequence (Figure 1). Therefore, unlike previous work that segments raw text into long sequences and shuffles the sequences for pre-training, we propose to create shorter text sequences instead, without shuffling. In this way, a shorter sequence may not contain all the necessary information for recovering the masked words, hence requiring the probing into surrounding sequences to capture the missing information.

3.2 Cross-Thought Pre-training

The pre-training model is illustrated in Figure 3(a). As aforementioned, the input of pre-training data consists of M continuous sequences $[X_0, X_1, ..., X_{M-1}]$. Similar to BERT (Devlin et al., 2019), we use the hidden state of the special token as the final sentence embedding. To encode the embeddings with richer semantic information, we propose to pre-append N special tokens S instead of a single one to each sequence X.

We first use Transformer to encode the segmented short sequences as follows:

$$\mathbf{H}_m = \text{Transformer}([\mathbf{S}; \mathbf{X}_m]), \quad (1)$$
$$\mathbf{E}_m = \mathbf{H}_m[0{:}N-1], \quad (2)$$

where $\mathbf{S} \in \mathbb{R}^{N \times d}$ are the embeddings of N special tokens, and $\mathbf{X}_m \in \mathbb{R}^{l_m \times d}$ are the contextualized word embeddings of the m-th sequence. l_m is the sequence length and d is the dimension of the embedding. $[\cdot; \cdot]$ is the concatenation of matrices. $\mathbf{H}_m \in \mathbb{R}^{(N+l_m) \times d}$ are all the hidden states of the Transformer, and $\mathbf{E}_m \in \mathbb{R}^{N \times d}$ are the hidden states on the special tokens, used as the final sequence embedding.

Next, we build cross-sequence Transformer on top of these sequence embeddings, so that each sequence can distill information from other sequences. As the embeddings of different special tokens encode different information, we run Transformer on the embedding of each special token separately:

$$\mathbf{F}_n = [\mathbf{E}_0[n]; \mathbf{E}_1[n]; ...; \mathbf{E}_{M-1}[n]], \quad (3)$$
$$\mathbf{C}_n = \text{Cross-Transformer}(\mathbf{F}_n), \quad (4)$$

where $\mathbf{E}_m[n] \in \mathbb{R}^d$ is the n-th row of $\mathbf{E}_m$. $\mathbf{F}_n \in \mathbb{R}^{M \times d}$ is the concatenation of all the embeddings of the n-th special token in each sequence. $\mathbf{C}_n \in$
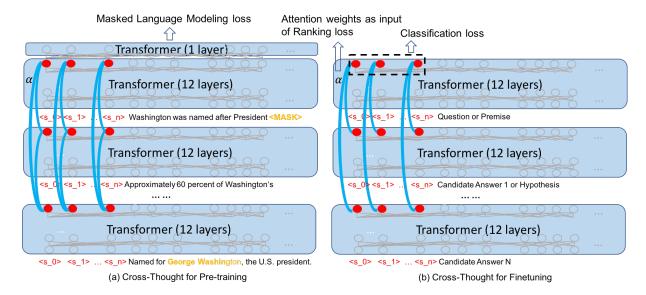
Figure 3: Illustration of Cross-Thought for pre-training and finetuning procedures. Circles in red are sentence embeddings. Lines in blue are the cross-sequence Transformers, the attention weights of which are α in Eqn. (7). Words in red are special tokens, the hidden states of which are the sentence embeddings. Multiple special tokens are used to enrich the sentence embeddings. In (a), the sentence embedding of the third sequence provides context that can help generate the masked word in the first sequence. In (b), the model can be initialized with pre-trained Cross-Thought for Answer Selection or Textual Entailment tasks. The attention weights α and hidden states of cross-sequence Transformers can be used directly for Ranking and Classification tasks.

$\mathbb{R}^{M \times d}$ is the output of cross-sequence Transformer, where all the information across sequences are fused together. As the weights of multi-head attention in cross-sequence Transformer will be used for downstream tasks, we decompose the attention weights of one head in the cross-sequence Transformer on the n-th special tokens as follows:

$$\mathbf{Q} = \mathbf{W}^Q \mathbf{F}_n, \quad (5)$$
$$\mathbf{K} = \mathbf{W}^K \mathbf{F}_n, \quad (6)$$
$$\alpha = \text{Softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d}}\right), \quad (7)$$

where $\mathbf{W}^Q \in \mathbb{R}^{d \times d}, \mathbf{W}^K \in \mathbb{R}^{d \times d}$ are the parameters to learn. $\alpha \in \mathbb{R}^{M \times M}$ are the attention weights that can be directly used in downstream tasks (e.g., for measuring the similarity between question and candidate answers in QA tasks).

Finally, to encourage the embedding from other sequences retrieved by cross-sequence Transformer to help generate the masked words in the current sequence, we use another Transformer layer on top of the merged sequence embeddings as follows:

$$\mathbf{G}_m = [\mathbf{C}_0[m]; \mathbf{C}_1[m]; ...; \mathbf{C}_{N-1}[m]; \mathbf{H}_m[N:]],$$
$$\mathbf{O}_m = \text{Transformer}(\mathbf{G}_m), \quad (8)$$

where $\mathbf{C}_n[m] \in \mathbb{R}^d$ is the hidden state of cross-sequence Transformer on the n-th special token

of the m-th sequence. $\mathbf{H}_m[N:] \in \mathbb{R}^{l_m \times d}$ from Eqn.(1) is the hidden state for non-special words X_m. $\mathbf{O}_m \in \mathbb{R}^{(N+l_m) \times d}$ will be used to generate the masked word:

$$\mathcal{L}_{\text{mask}} = \sum_{m,i} -\log(P(a_{m,i}|\mathbf{O}_m)), \quad (9)$$

where $P(a_{m,i}|\mathbf{O}_m)$ is the probability of generating the i-th masked word in the m-th sequence.

3.3 Cross-Thought Finetuning

To demonstrate how pre-trained Cross-Thought can initialize models for downstream tasks, we take two tasks as examples: answer selection and sequence-pair classification (under the setting of using sentence embeddings only, without word-level cross-sequence attention (Devlin et al., 2019)). The procedure is illustrated in Figure 3 (b).

Answer Selection The goal is to select one answer from a candidate pool, $\{X_1, X_2, ..., X_{M-1}\}$ based on question X_0. We consider the representations of candidate answers that are cached and can be matched to question embeddings when a new question comes in. Based on the pre-trained model, the attention weights in Eqn.(7) from different heads of the cross-sequence Transformers can be directly applied to rank the answer candi-

dates. For further finetuning, the loss relying on the attention weights is defined as follows:

$$\mathcal{L}_{\text{answer}} = -\log(\alpha[0][m]), \quad (10)$$

where $\alpha[0]$ are the attention weights between question X_0 and all the answer candidates, and m is the index of the correct answer in $\{X_1, X_2, ..., X_{M-1}\}$. Note that we have multiple cross-sequence Transformers on different special tokens, and each Transformer has multiple heads. Thus, we use the mean value of all the attention matrices as the final weights.

Sentence Pair Classification   The goal is to identify the relation between two sequences, X_0 and X_1. Reusable sequence embeddings are very useful in some tasks, such as finding the most similar pair of sentences from a large candidate pool, which requires large-scale repetitive encoding and matching without pre-computed sentence embeddings. As the pre-training of cross-sequence Transformer is designed to fuse the embeddings of different sequences, the merged representations in Eqn.(4) can be used for downstream classification as follows:

$$\begin{aligned} \bar{\mathbf{C}} &= [\mathbf{C}_0; \mathbf{C}_1; ...; \mathbf{C}_{N-1}], \\ \bar{\mathbf{c}} &= \text{Flatten}(\bar{\mathbf{C}}), \\ \mathcal{L}_{\text{cls}} &= -\log\left(\text{Softmax}\left(\mathbf{W}^c \bar{\mathbf{c}}^T\right)[y]\right), (11) \end{aligned}$$

where $\bar{\mathbf{C}} \in \mathbb{R}^{2N \times d}$ is the concatenation of the hidden states of all the cross-sequence Transformers on N different special tokens. Note that there are only two sequences here for classification. $\bar{\mathbf{c}} \in \mathbb{R}^{2Nd}$ is the reshaped matrix for final classification, and cross-entropy loss is used for optimization.

4   Experiments

In this section, we conduct experiments based on our pre-trained models, and provide additional detailed analysis.

4.1   Datasets

We conduct experiments on five datasets, the statistics of which is shown in Table 1.

MNLI (Williams et al., 2018)[2]: Multi-Genre Natural Language Inference matched (MNLI-m) and mismatched (MNLI-mm) are textual entailment tasks. The goal is to classify the relation between premise and hypothesis sentences into three

Dataset	#train	#test	#seq	Goal
MNLI-m	373K	10K	2	classification
MNLI-mm	373K	10K	2	classification
SNLI	549K	10K	2	classification
QQP	346K	391K	2	classification
Quasar-T	29K	3K	100	ranking
HotpotQA	86K	7K	5M	ranking

Table 1: Statistics of the datasets. #train and #test are the number of samples for training and testing. #seq is the number of sequences needed to use for each sample. 5M is for 5 million.

classes: entailment, contradiction and neutral. The train and test sets come from the same *source* and same *genre* in MNLI-m, and different in MNLI-mm.

SNLI (Bowman et al., 2015)[3]: The dataset of Stanford Natual Language Inference is another textual entailment task.

QQP (Wang et al., 2018): Quora Question Pairs is to identify whether two questions are duplicated or not.

Quasar-T (Dhingra et al., 2017)[4]: This is a dataset for question answering by searching the related passages and then reading it to extract the answer. In this dataset, we evaluate the models by whether it can correctly select the sentence containing the gold answer from the candidate pool.

HotpotQA (Yang et al., 2018)[5]: A dataset of diverse and explainable multi-hop question answering. We focus on the full-wiki setting, where the model needs to extract the answer from all the abstracts in Wikipedia and related sentences.

Note that for the datasets of MNLI, QQP and HotpotQA, the test sets are hidden and the number of submissions is limited. For a fair comparison between our models and baselines, we split 5% of the training data as validation set and use the original validation set as test set.

4.2   Implementation Details

All the models are pre-trained on Wikipedia and finetuned on downstream tasks. We also evaluate the pre-trained models on whether they can perform unsupervised paragraph selection, and whether the improvement over paragraph ranking can lead to better answer prediction on HotpotQA task. As our experiments are to evaluate the ability of sentence

---

[2]https://gluebenchmark.com/tasks

[3]https://nlp.stanford.edu/projects/snli/
[4]https://github.com/bdhingra/quasar
[5]https://hotpotqa.github.io/

	MNLI-m Acc	MNLI-mm Acc	SNLI Acc	QQP Acc	Quasar-T Recall@1	HotpotQA Recall@20	HotpotQA(u) Recall@20
ICT	65.2	65.3	81.8	85.0	40.5	81.5	44.9
Skip-Thought	65.3	65.7	81.5	84.6	40.2	82.0	37.6
Language Model	73.9	74.2	84.1	86.8	41.9	81.0	33.4
Masked LM-1-512	74.3	74.1	84.9	87.3	43.2	81.5	10.0
Masked LM-1-64	73.8	74.0	84.3	87.0	42.5	81.0	10.0
Masked LM (160G)	75.5	75.7	86.3	89.3	43.5	87.5	10.0
Cross-Thought-1-512	74.5	74.1	85.0	87.5	43.5	81.7	10.0
Cross-Thought-1-64	76.2	76.4	86.3	90.0	47.2	88.0	51.9
Cross-Thought-3-64	76.5	**76.6**	86.5	**90.3**	48.2	88.4	55.4
Cross-Thought-5-64	**76.8**	**76.6**	86.8	90.3	**48.5**	**88.9**	**56.5**

Table 2: Results on only using sentence embedding for classification and ranking. Cross-Thought-3-64 is to train Cross-Thought by pre-appending 3 special tokens to the sequences that are segmented into 64 tokens. For HotpotQA, we only evaluate on how well the model can retrieve gold paragraphs. Results for HotpotQA(u) are without finetuning. Acc: Accuracy. Recall@20: recall for the top 20 ranked paragraphs.

Models	HotpotQA (full-wiki)		
	Pas EM	Ans EM/F1	Sup EM/F1
Cognitive Graph (Ding et al., 2019)	57.8	37.6/49.4	23.1/58.5
Semantic Retrieval (Nie et al., 2019)	63.9	46.5/58.8	39.9/71.5
Recurrent Retriever (Asai et al., 2020)	72.7	60.5/73.3	49.3/76.1
Masked LM-1-64 + reranker + reader	77.2	60.9/73.5	52.9/77.3
Cross-Thought-1-64 + reranker + reader	**80.0**	**62.3/75.1**	**54.3/78.6**

Table 3: Results on HoptpotQA (full-wiki setting). We use sentence embeddings from the finetuned model of Cross-Thought or Masked LM as information retriever (IR) to collect candidate paragraphs. Pas EM: exact match of gold paragraphs; Ans EM/F1: exact match/F1 on short answer; Sup EM/F1: exact match/F1 on supporting facts.

encoder, we only build a light layer on sentence embeddings for classification task, and use only dot product (Cross-Thought, ICT) or cosine similarity (Skip-thought, LM, MLM) between sentence embeddings for ranking task. Note that for fair comparison, all the encoders in our experiments have the same structure as RoBERTa-base (12 layers, 12 self-attention heads, hidden size 768). For all experiments, we use Adam (Kingma and Ba, 2015) as the optimizer and use the tokenizer of GPT-2 (Radford et al., 2018).

For model pre-training, all models including the baselines we re-implement are trained with Wikipedia pages.[6] We use 16 NVIDIA V100 GPUs for model training. Our code is mainly based on the RoBERTa codebase,[7] and we use similar hyperparameters as RoBERTa-base training. Each train-ing sample contains 500 short sequences with 64 tokens, and we randomly mask 15% of the tokens in the sequences. During training, we fix the position embeddings for the pre-appended special tokens, and randomly select 64 continuous positions from 0 to 564 for the other words. Thus, the model can be used to encode longer sequences in downstream tasks. The batch size is set to 128 (4 million tokens). We use warm-up steps 10,000, maximal update steps 125,000, learning rate 0.0005, dropout 0.1 for pre-training. Each model is pre-trained for around 4 days.

For model finetuning, in experiments for MNLI, SNLI and QQP, we use batch size 32, warmup steps 7,432, maximal update steps 123,873, and learning rate 0.00001. For Quasar-T and HotpotQA, we set batch size 80, warmup steps 2,220, maximal update steps 20,360, and learning rate 0.00005. Dropout is the only hyper-parameter we tuned, and 0.1 is the best from [0.1, 0.2, 0.3]. As HotpotQA in full-wiki

---

[6]The Wikipedia dump we use is enwiki-20191001-pages-articles-multistream.

[7]https://github.com/pytorch/fairseq

setting does not provide answer candidates, we randomly sample 100 negative paragraphs from the top 1000 paragraphs ranked by BM25 scores during training. During inference, we use sentence embeddings to further rank the top 1000 paragraphs. For the unsupervised experiment on HotpotQA, we only rank top 200 paragraphs.

4.3 Baselines

Existing baseline methods are mostly trained with different encoders or different datasets. For fair comparison, we re-implement all these baselines by using a 12-layer Transformer as the sentence encoder and Wikipedia as the source for pre-training. There are three groups of baselines considered for evaluation:

Pre-trained Sentence Embedding

- *ICT* (Lee et al., 2019): Inverse cloze task treats a sentence and its context as a positive pair, otherwise negative. Sentences are masked from the context 10% of the time. This model is trained by ranking loss based on the dot product between sequence embeddings.

- *Skip-Thought* (Kiros et al., 2015): The task is to encode sentences into embeddings that are used to re-construct the next and the previous sentences. This model is based on encoder-decoder structure without considering attention across sequences (Cho et al., 2014). We use 6-layer Transformer as the decoder for re-construction.

Language Modeling In addition, we also re-implement benchmark baselines on the classic Language Modeling (LM) and Mask Language Modeling tasks, as most existing models are pre-trained with different unlabeled datasets:

- *Language Model (LM)* (Radford et al., 2018): The task is to predict the probability of the next word based on given context. As the words are sequentially encoded, to evaluate the performance of this model on HotpotQA in the unsupervised setting, we use the last hidden state as the sentence embedding (instead of the first one by ICT, MLM, Skip-Thought).

- *Masked Language Model (MLM)* (Devlin et al., 2019): The task is to generate randomly masked words from sequences. We explore different settings of training data. "Masked LM-1-512" trains a Transformer on sequences with 512 tokens and pre-appends 1 special token to each

sequence. "Masked LM-1-64" is trained on sequences with 64 tokens. Both models are trained with Wikipedia text only. "Masked LM (160G)" is the RoBERTa model pre-trained on a much larger corpus.

Multi-hop Question Answering To further evaluate our model on multi-hop question answering task in open-domain setting, we compare our framework with several strong baselines on HotpotQA:

- *Cognitive Graph* (Ding et al., 2019): It uses an iterative process of answer extraction and further reasoning over graphs built upon extracted answers.

- *Semantic Retrieval* (Nie et al., 2019): It uses a semantic retriever on both paragraph- and sentence-level to retrieve question-related information.

- *Recurrent Retriever* (Asai et al., 2020): It uses a recurrent retriever to collect useful information from Wikipedia graphs for question answering.

4.4 Experimental Results

Results on the classification and ranking tasks are summarized in Table 2. Results of our pipeline on HotpotQA (full-wiki) are shown in Table 3.

Effect of Pre-training Tasks Among all the pre-training tasks, our proposed method Cross-Thought achieves the best performance. With finetuning, LM pre-training tasks work better than the Skip-Thought and ICT methods which are specifically designed for learning sentence embedding. Moreover, we provide a fair comparison between "Cross-thought-1-64" and "Masked LM-1-64", both of which segment Wikipedia text into short sequences in 64 tokens for pre-training, and only use the hidden state of the first special token as sentence embedding. Results show that our Cross-Thought model achieves much better performance than Masked LM-1-64, as well as the Transformer pre-trained on 160G data (10 times larger than Wikipedia).

Effect of Training on Short Sequences Results on "Cross-Thought-1-512" and "Cross-Thought-1-64" (using sequences of 512 tokens and 64 tokens, respectively) clearly show that shorter sequences lead to more effective pre-training. Moreover, we also observe that "Cross-Thought-1-512" and "Masked LM-1-512" achieve almost the same performance. It means that our Cross-Thought has

Q	Jens Risom introduced what type of design, characterized by minimalism and functionality?	Washington was named after President <MASK>by an act of the <MASK>Congress during the creation of Washington Territory in 1853.
C1	**Scandinavian design** is a design movement characterized by simplicity, minimalism and functionality that emerged in the 1950s in the five Nordic countries of Finland, Norway, Sweden, Iceland and Denmark... (attention weight: **0.259**)	Washington, officially the State of Washington, is a state in the Pacific Northwest region of the United States. Named for **George Washington**, the first U.S. president. (attention weight: **0.987**)
C2	Risom was one of the first designers to introduce **Scandinavian design** in the United States... (attention weight: **0.194**)	Approximately 60 percent of Washington's residents live in the Seattle metropolitan area, the center of transportation, business, and industry. (attention weight: **0.012**)
C3	Dutch Design can be characterized as minimalist, experimental, innovative, quirky, and humorous... (attention weight: **0.098**)	Manufacturing industries in Washington include aircraft and missiles, shipbuilding, and other ... (attention weight: **0.001**)

Table 4: Case study on unsupervised passage ranking. The attention weights are learned by cross-sequence Transformer from pre-training. The examples on the left come from HotpotQA and are the ranked passages from 200 candidates for answering the question. The examples on the right are in the format of Masked Language Modeling task, where our Cross-Thought needs to recover the masked words by leveraging other sequences. C: the ranked passages by attention weights.

to be trained on short sequences (64 tokens); otherwise, it would learn more on the word dependencies within sequence other than the sequence embeddings. Actually, the effect of short sequences is also proved by Skip-Thought which focuses on generating sequences in sentence level, but our Cross-Thought can achieve better performance.

Effect of Sentence Embedding Size As we keep the number of parameters fixed for the encoders trained with different tasks, increasing the dimension of hidden state will lead to more parameters to train. Instead, for each sequence, we pre-append more special tokens, the hidden states of which are concatenated together as the final sentence embedding. Experiments on "Cross-Thought-1-64", "Cross-Thought-3-64" and "Cross-Thought-5-64" compare pre-appending 1, 3 and 5 different special tokens to sequences for pre-training. We can see that a larger sentence embedding size can significantly improve performance on the ranking tasks while not on the classification tasks. We hypothesize that the main reason is ranking tasks are more challenging, with many different pairs to compare, for which the contextual sentence embeddings can provide additional information.

Effect of Paragraph Ranking without Finetuning We also conduct an analysis on whether pre-trained sentence embeddings can be directly used for downstream tasks without finetuning. Although model performance without finetuning is generally worse than supervised training, experiments in column "HotpotQA(u)" further validate the previously discussed three conclusions. Besides, we observe

that although the model pre-trained by masked language modeling leads to better performance after finetuning, it is not designed to train sentence embeddings, thus cannot be used for passage ranking. While all the other methods achieve much better performance than masked language modeling, our model "Cross-Thought-5-64" with the largest embedding size achieves the best performance.

Effect of Cross-Thought as Information Retriever (IR) on QA Task Our pipeline of solving HotpotQA (full-wiki) consists of three steps: (i) Fast candidate paragraph retrieval; (ii) Multi-hop paragraphs re-ranking by a more complex model; and (iii) Answer and supporting facts extraction. We evaluate our proposed method on how well the finetuned sentence embeddings can be utilized in the first step for IR, with the re-ranker and answer extractor fixed. "Masked LM-1-64" and "Cross-Thought-1-64" in Table 3 show that our pre-trained model achieves better performance than the baseline model pre-trained on single sequences. Moreover, the pipeline integrating our sentence embedding achieves new state of the art on HotpotQA (full-wiki).

4.5 Case Study

Table 4 provides a case study on the unsupervised passage ranking and masked language modeling tasks. For the case from HotpotQA, we can see that attention weights from the cross-sequence Transformer in Cross-Thought can rank the paragraph with gold answer to the first place among the 200 candidate paragraphs.

For the case from Mased Language Modeling, we also observe that the sentence that can be used to recover the masked words receives much higher attention weight compared to others, validating our motivation on retrieving the useful sentence embeddings from other sequences to enhance masked word recovery in the current sequence.

5 Conclusion

We propose a novel approach, Cross-Thought, to pre-train sentence encoder. Experiments demonstrate that using Cross-Thought trained with short sequences can effectively improve sentence embedding. Our pre-trained sentence encoder with further finetuning can beat several strong baselines on many NLP tasks.

References

Akari Asai, Kazuma Hashimoto, Hannaneh Hajishirzi, Richard Socher, and Caiming Xiong. 2020. Learning to retrieve reasoning paths over wikipedia graph for question answering. In *International Conference on Learning Representations (ICLR)*.

Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. 2015. A large annotated corpus for learning natural language inference. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, et al. 2018. Universal sentence encoder. *arXiv preprint arXiv:1803.11175*.

Jianpeng Cheng, Li Dong, and Mirella Lapata. 2016. Long short-term memory-networks for machine reading. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *International Conference on Learning Representations (ICLR)*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *North American Chapter of the Association for Computational Linguistics (NAACL)*.

Bhuwan Dhingra, Kathryn Mazaitis, and William W Cohen. 2017. Quasar: Datasets for question answering by search and reading. *arXiv preprint arXiv:1707.03904*.

Ming Ding, Chang Zhou, Qibin Chen, Hongxia Yang, and Jie Tang. 2019. Cognitive graph for multi-hop reading comprehension at scale. In *Association for Computational Linguistics (ACL)*.

Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. 2019. Unified language model pre-training for natural language understanding and generation. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. 2020. Realm: Retrieval-augmented language model pre-training. *arXiv preprint arXiv:2002.08909*.

Felix Hill, Kyunghyun Cho, and Anna Korhonen. 2016. Learning distributed representations of sentences from unlabelled data. In *North American Chapter of the Association for Computational Linguistics (NAACL)*.

Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. In *International Conference on Information & Knowledge Management (CIKM)*.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. *International Conference on Learning Representations (ICLR)*.

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *Advances in neural information processing systems (NeurIPS)*.

Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. 2019. Latent retrieval for weakly supervised open domain question answering. In *Association for Computational Linguistics (ACL)*.

Mike Lewis, Marjan Ghazvininejad, Gargi Ghosh, Armen Aghajanyan, Sida Wang, and Luke Zettlemoyer. 2020. Pre-training via paraphrasing. *arXiv preprint arXiv:2006.15020*.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*.

Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding. *International Conference on Learning Representations (ICLR)*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Lili Mou, Rui Men, Ge Li, Yan Xu, Lu Zhang, Rui Yan, and Zhi Jin. 2016. Natural language inference by tree-based convolution and heuristic matching. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Yixin Nie, Songhe Wang, and Mohit Bansal. 2019. Revealing the importance of semantic retrieval for machine reading at scale. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *North American Chapter of the Association for Computational Linguistics (NAACL)*.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. *URL https://s3-us-west-2. amazonaws. com/openai-assets/researchcovers/languageunsupervised/language understanding paper. pdf*.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Sandeep Subramanian, Adam Trischler, Yoshua Bengio, and Christopher J Pal. 2018. Learning general purpose distributed sentence representations via large scale multi-task learning. In *International Conference on Learning Representations (ICLR)*.

Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Association for Computational Linguistics (ACL)*.

Ming Tan, Cicero dos Santos, Bing Xiang, and Bowen Zhou. 2015. Lstm-based deep learning models for non-factoid answer selection. *arXiv preprint arXiv:1511.04108*.

Alex Wang, Amapreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *North American Chapter of the Association for Computational Linguistics (NAACL)*.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Empirical Methods in Natural Language Processing (EMNLP)*.