# Transcormer: Transformer for Sentence Scoring with Sliding Language Modeling

**Anonymous Author(s)**
Affiliation
Address
email

## Abstract

Sentence scoring aims at measuring the likelihood score of a sentence and is widely used in natural language processing scenarios, like reranking, which is to select the best sentence from multiple candidates. Previous works on sentence scoring mainly adopted either causal language modeling (CLM) like GPT or masked language modeling (MLM) like BERT, which have some limitations: 1) CLM only utilizes unidirectional information for the probability estimation of a sentence without considering bidirectional context, which affects the scoring quality; 2) MLM can only estimate the probability of partial tokens at a time and thus requires multiple forward passes to estimate the probability of the whole sentence, which incurs large computation and time cost. In this paper, we propose *Transcormer* – a Transformer model with a novel *sliding language modeling* (SLM) for sentence scoring. Specifically, our SLM adopts a triple-stream self-attention mechanism to estimate the probability of all tokens in a sentence with bidirectional context and only requires a single forward pass. SLM can avoid the limitations of CLM (only unidirectional context) and MLM (multiple forward passes) and inherit their advantages, and thus achieve high effectiveness and efficiency in scoring. Experimental results on multiple tasks demonstrate that our method achieves better performance than other language models.

## 1 Introduction

Sentence scoring is to measure the log-likelihood score of a sentence via language model, so that it can be used to represent the goodness of a sentence. Specifically, a good sentence should have a relatively lower log-likelihood score while a bad sentence should have a higher one. Due to such nature, sentence scoring has been widely used in many natural language processing (NLP) scenarios. For instance, it can be used to rerank candidates in neural machine translation (NMT) or automatic speech recognition (ASR) tasks [1, 2] or evaluate sentences in linguistic acceptability [3]. Therefore, how to design effective language models to calculate sentence scores efficiently is very important.

Recently, neural network based language models (LM) [4, 5, 6, 7] have been considered as the most widely used technique for sentence scoring, since they can produce density estimation of the whole sentence by computing the probability of each token and summing up their values as the sentence score. Specifically, causal language modeling (CLM) [8] and masked language modeling (MLM) [9] are the most representative LMs. For a given sentence $\mathbf{x} = \{x_1, \cdots, x_{|\mathbf{x}|}\}$, where $x_i$ is the $i$-th token of $\mathbf{x}$. CLM can predict next token conditioned on unidirectional context and its objective is to optimize $\sum_{i=1}^{|\mathbf{x}|} \log P(x_i|x_{<i})$. Hence, CLM is usually used for solving natural language generation

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

(NLG) tasks [8, 10, 11]. While for MLM, it replaces a subset of tokens $\mathbf{x}_{\mathcal{S}}$ in $\mathbf{x}$ as special symbol $[\mathrm{MASK}]$ and then predicts the masked tokens based on the corrupted sequence $\mathbf{x}_{\setminus\mathcal{S}}$. The objective of MLM is to optimize $\sum_{i=1}^{|\mathcal{S}|} \log P(y_{\mathcal{S}_i}|\mathbf{y}_{\setminus\mathcal{S}})$, so it is able to learn bidirectional context and can be used for solving natural language understanding (NLU) tasks [9, 12, 13, 14]. Since CLM and MLM can be learned in an unsupervised fashion, many works [9, 12, 13, 8, 10, 11, 15, 16, 17, 14, 18, 19] have pre-trained these LMs on large-scale corpus to extract powerful linguistic representations, and these models can be directly used out of the box to predict the probabilities of tokens. Inspired by the successes of LMs, some works [20, 21, 22, 23, 24] have tried to use pre-trained CLM or MLM to generate sentence scores on NMT or ASR reranking tasks and achieved some promising results.

However, we notice that both CLM and MLM still suffer from some deficiencies in calculating sentence scores. For instance, some works [24] utilized GPT-style model for ASR reranking within single inference, yet GPT model can only extract unidirectional information due to the limitation of CLM, without considering the whole sentence semantics, and thus affect the sentence score. To utilize bidirectional context, some works [21, 22, 23] applied BERT model for rescoring. However, the nature of MLM is to mask some tokens in the sentence for prediction, which means it requires the BERT model to forward multiple times and each forward pass only masks one token for prediction. As a result, it is time-consuming to directly adopt MLM for scoring sentence. In summary, we find that for calculating sentence scores, CLM needs one-pass inference but only uses unidirectional information and MLM is costly in computing sentence score although it can use bidirectional context. Therefore, a natural question arises: is it possible to design a language model to use bidirectional context for sentence scoring with only one inference pass.

In this paper, we introduce *Transcormer*, a Transformer model designed for sentence scoring. More specifically, our Transcormer leverages a novel language modeling scheme, named as *sliding language modeling* (SLM), that produces the probability of all tokens within single inference pass and simultaneously utilizes bidirectional context. To fulfill this target, we innovatively design a triple-stream self-attention mechanism, which consists of two content streams (a forward stream and a backward stream) and one query stream. By employing specifically-designed mask strategies on the attention matrix, our method allows each token in the query stream to leverage all token information except itself (i.e., the tokens before and after it) for estimating its probability to avoid any information leakage. To the best of our knowledge, SLM is the first language modeling tailored for sentence scoring. We pre-train our SLM on large-scale corpus, and then evaluate it on multiple datasets. Experimental results demonstrate that Transcormer outperforms the baselines by up to + 0.8/0.6 BLEU score on small/large-scale NMT tasks, $\sim$20% relative improvements on ASR tasks.

The main contributions of this work are summarized as follows:

- We analyze the pros and cons of CLM and MLM when using them for scoring sentences, and propose Transcormer with a new sliding language modeling, which uses bidirectional context for probability estimation within a single pass.
- We introduce a novel triple-stream self-attention mechanism in SLM, which has two content streams to collect forward/backward semantics, and a query stream to estimate the probability of each token in a sentence.
- Experimental results on multiple datasets demonstrate the effectiveness and efficiency of our SLM for sentence scoring.

## 2 Background

### 2.1 Sentence Scoring

Sentence scoring has a long history in NLP applications, especially in reranking tasks (*e.g.*, reranking for machine translation [25, 26, 27, 28] or speech recognition [29]). Generally, given $n$-best candidates generated by text generation models, reranking aims at scoring each candidate to select the best answer. Early works [25, 26, 27, 28, 30, 31] mainly used statistical LM or combined it with

RNN-based LM to calculate the sentence scores. Recently, end-to-end neural network based LM has became the de facto approach for scoring and has been applied in many NLP tasks [5, 4, 32]. Specifically, causal language modeling (CLM) and masked language modeling (MLM) are the most representative language modeling methods, among which GPT [8, 10, 11] and BERT [9] are the most famous examples, respectively. As aforementioned, CLM conditions on the previous states to predict next token so that it can obtain the probability of each token of the sentence in a single pass, but can only capture unidirectional information. For MLM, it is able to use bidirectional context for prediction but the masked-prediction mechanism limits MLM to producing the probability of all tokens within one forward pass (since MLM can only provide the probability of masked tokens). To calculate the sentence score, a kind of solutions [21, 22, 23] is to forward multiple times and only mask one token each time. Therefore, the MLM for sentence scoring is formulated as: $\sum_{i=1}^{|\mathbf{x}|} \log P(x_i|\mathbf{x}_{\setminus x_i})$. We can find that the cost of MLM for scoring needs $|\mathbf{x}|$ inference passes, which is too time-consuming.

Some recent works [21, 20, 33, 34] have been proposed to alleviate these issues in MLM. Wang et al. [21] and Salazar et al. [20] attempted to use stochastic estimation or distillation to avoid N-passes problem to approximately estimate the probability of each toke produced by MLM, with a sacrifice of performance. Clark et al. [33] adopted a two-cloze tower [35] with noise-contrastive estimation to provide sentence probability, and Shin et al. [34] only considered word embedding as the inputs of key and value in transformer without any interaction. Besides, some works [36, 37] adopted discriminative language modeling to approximately estimate sentence scores based on the paired data. More discussions about related works can be found in Appendix A.2. Therefore, how to calculate sentence scores efficiently with bidirectional context is the main challenge in MLM.

Overall, CLM only needs a single forward pass to estimate the probability of all tokens but cannot extract bidirectional context, while MLM leverages bidirectional information but needs multiple inference passes. Consequently, we raise a natural question: is it possible to design a pre-trained language model to support all token prediction in a single pass and simultaneously leverage bidirectional information? This is exactly the motivation of our method.

### 2.2 Multiple-Stream Self-Attention

The pioneer of multiple-stream self-attention is XLNet [15], which introduces two-streams self-attention to incorporate autoregressive pre-training for language understanding, which pre-trains Transformer [38] via using a content stream and a query stream. In details, for the $t$-th step, the content stream is able to capture the dependency from the tokens before the $t$-th step and itself (*i.e.*, $x_{\leq t}$), while the query stream is only allowed to view the tokens before the $t$-th step (*i.e.*, $x_{<t}$) to avoid information leakage. Besides, there are some other variants of two-stream self-attention [16, 39, 40], which are designed for solving NLU tasks. We observe that these works mainly used a single content stream and then used one or many query streams to predict more information. Different from these works, we introduce a triple-stream self-attention mechanism, which enables query stream to leverage two content streams for prediction, and thus enjoys the benefits of additional bidirectional context for estimating token probability.

## 3 Transcormer

To inherent the advantages of CLM and MLM for sentence scoring and avoid their limitations, we propose Transcormer – a Transformer model with a novel sliding language modeling for sentence scoring. First, we summarize that an ideal language modeling for sentence scoring should satisfy two requirements: 1) model should be able to use bidirectional context for effective probability estimation of each token; 2) model should produce the probability of all tokens in a sentence within a single inference pass for efficiency. Therefore, to fulfill these two requirements, we formulate a new language modeling, named as sliding language modeling (SLM), and describe it in Section 3.1. In SLM, we propose a Triple-Stream Self-Attention mechanism based on Transformer (please see Section 3.2 for details) to use bidirectional context for each token prediction and avoid information

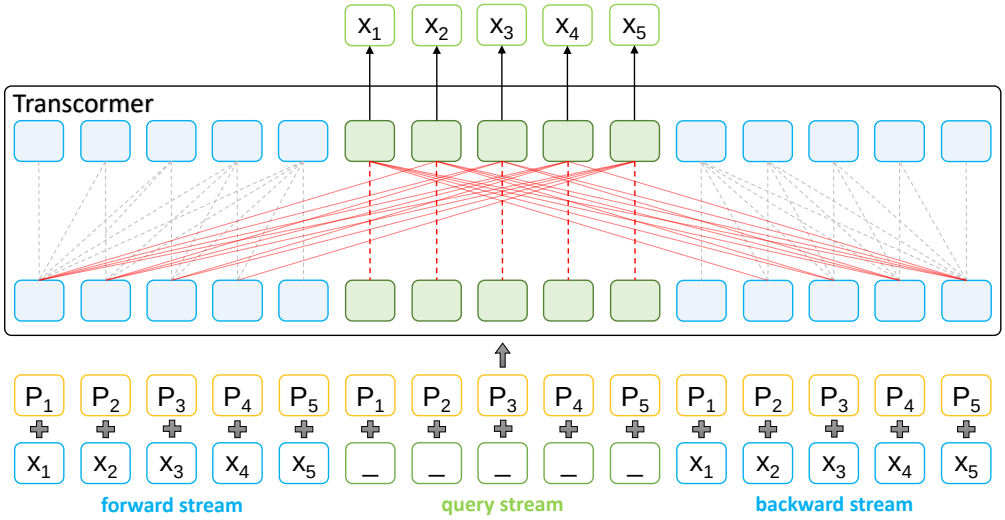



Figure 1: Transcormer with sliding language modeling. The left and the right (in blue) are forward and backward streams, respectively, and the middle (in green) is query stream. For query stream, the inputs are only the positional information. We use gray and red line to represent the allowed attended positions in the content and query streams.

leakage. We also discuss the differences between SLM and other LMs in Section 3.3. Figure 1 presents the pipeline of our Transcormer for sentence scoring with SLM.

### 3.1 Sliding Language Modeling

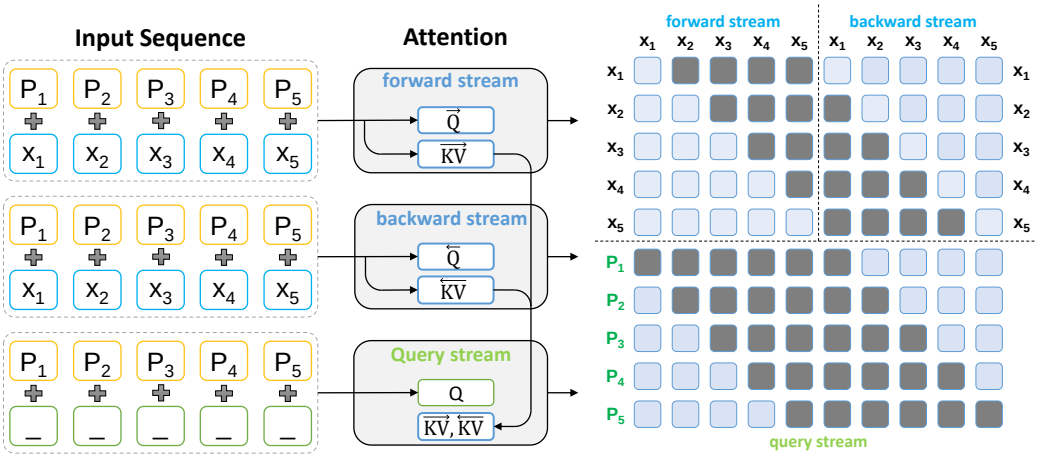
Considering the pros and cons of CLM and MLM for scoring, we notice: 1) CLM can produce probability of all tokens within one forward pass, and thus obtain the unidirectional information of the whole sentence; 2) MLM for sentence scoring needs multiple inference passes, so that many context has been repeatedly calculated and cause a waste of computation. So, is it possible to reuse token information to build bidirectional context for token prediction?

Therefore, we propose sliding language modeling (SLM) to address the inherent flaws in previous LMs (*i.e.*, CLM and MLM) for sentence scoring. Specifically, we first maintain two individual streams to collect forward (left-to-right) context and backward (right-to-left) context. And for each token prediction, we decompose the sentence information as the past tokens (the tokens before it) and future tokens (the tokens after it) respectively. As a result, our SLM enforces each token to only capture the dependency from its past tokens and its future tokens concurrently, so that each token can utilize the whole sentence information (except itself) to estimate token probability. The objective function of SLM is formulated as:

$$\mathcal{L} = \sum_{i=1}^{|\mathbf{x}|} \log P(x_i | \mathbf{x}_{<i}, \mathbf{x}_{>i}; \theta), \tag{1}$$

where $\mathbf{x}_{<i}$ and $\mathbf{x}_{>i}$ respectively correspond to the tokens before the $i$-th token and after the $i$-th token, and $\theta$ represents the parameters of SLM. Thanks to such design, our SLM can utilize bidirectional context to produce the probability of each token within one forward pass, and thus satisfy the above requirements for sentence scoring. However, previous experiences [15, 41] pointed out that the states with bidirectional information will cause information leakage when propagating to the next layer [1]. Therefore, how to implement SLM to avoid information leakage and maintain different states together is still a troublesome problem.

[1]For example, assume the sequence has 3 tokens, and the hidden states of the $i$-th token at the first layer as $h_i^1$. So each position $h_1^1$, $h_2^1$ and $h_3^1$ should collect the information from positions $\{2, 3\}$, $\{1, 3\}$, $\{1, 2\}$. However, when $h_1^1$ and $h_3^1$ are delivered to $h_2^2$ at the second layer, it will cause a cyclic leakage as $h_2^2$ should not obtain information from position 2.




(a) (b)

Figure 2: (a) The structure of our triple-stream self-attention used in our sliding language modeling. The query stream reuses the hidden states from both forward and backward (content) stream as the key and value in attention. (b) The attention mask matrix used in our triple-stream self-attention. The above row is the attention matrix for the forward and backward stream and the below row is the attention matrix for the query stream. The cell in gray color means this position cannot be attended.

### 3.2 Triple-Stream Self-Attention

Based on the Eqn 1 of our proposed SLM, model needs to maintain two states to collect forward and backward contexts for prediction, and we call these two states as the forward stream and backward stream respectively. To avoid information leakage, we additionally maintain an individual state for prediction and control it to only capture the dependency from the forward and backward streams. We name this state as the query stream. Therefore, we propose a novel Triple-Stream Self-Attention to update each stream, and the detailed design is described as following.

To fulfill our target, we choose Transformer [38] as our basic model, due to its flexibility in capturing global dependency. Assume the input sequence as $\{w_1, w_2, \cdots, w_n\}$ and its positions as $\{p_1, p_2, \cdots, p_n\}$, where $w_i$ and $p_i$ represents the embedding of $i$-th token and its position in the sentence, and $n$ is the token number. For the query stream, we only use the positional embeddings $\{p_1, p_2, \cdots, p_n\}$ as the input. For the forward and backward streams, we also maintain two individual states and both of them use the tokens plus its position (*i.e.*, $\{w_1 + p_1, w_2 + p_2, \cdots, w_n + p_n\}$) as the input. For the $l$-th layer calculation, we denote the forward and backward streams of the position $i$ as $\vec{h}_i^l$ and $\overleftarrow{h}_i^l$, and they are updated as:

$$\vec{h}_i^l = \text{Attention}(\mathbf{Q} = \vec{h}_i^{l-1}, \mathbf{KV} = \vec{h}_{\leq i}^{l-1}; \theta), \tag{2}$$

$$\overleftarrow{h}_i^l = \text{Attention}(\mathbf{Q} = \overleftarrow{h}_i^{l-1}, \mathbf{KV} = \overleftarrow{h}_{\geq i}^{l-1}; \theta), \tag{3}$$

where $\text{Attention}(\cdot, \cdot)$ refers to the self-attention [38] in Transformer and $\mathbf{Q}$, $\mathbf{K}$, $\mathbf{V}$ denote the query, key and value in self-attention. Hence, $\vec{h}_i^l$ collects the information before the position $i$ and itself, and $\overleftarrow{h}_i^l$ collects the information after the position $i$ and itself. For the query stream, we denote $q_i^l$ as its hidden states, and concatenate forward stream $\vec{h}_i^l$ and backward stream $\overleftarrow{h}_i^l$ of the current layer to be the key/value of query stream. So $q_i^l$ is updated as:

$$q_i^l = \text{Attention}(\mathbf{Q} = q_i^{l-1}, \mathbf{KV} = \left[\vec{h}_{<i}^{l-1}, \overleftarrow{h}_{>i}^{l-1}\right]; \theta). \tag{4}$$

Here we find that $q_i^l$ is required to only capture the dependency from the forward stream before the position $i$ and the backward stream after the position $i$, rather than itself. Due to such design, the query stream is able to capture bidirectional context for estimating token probability and avoid information leakage, which is more effective than CLM in using context. More importantly, our

| LM | Model | Cost | Context | Scenario |
|---|---|---|---|---|
| CLM | GPT [10] | $\times 1$ | forward | NLG |
| MLM | BERT [9] | $\times$n | bidirectional | NLU |
| Bi-LM | ELMO [42] | $\times 2$ | forward + backward | NLU |
| SLM | Transcormer | $\times 3$ | bidirectional | Scoring |

Table 1: Comparisons between SLM and other LMs. We assume all LMs adopt the same architectures (*e.g.*, Transformer). The "Cost" column means the relative computations compared with CLM when calculating a sentence with $n$ tokens. The "Context" column means the contextual information usage for prediction.

triple-stream self-attention enables model to predict the probability of all tokens in a sentence within a single forward pass, which demonstrates more efficiency than MLM. Figure 2 presents the detailed design of our triple-stream self-attention. We can find that in the query stream, the masked matrix is like a sliding window to control each token to view its previous states in forward stream and its future states in backward stream. And that is why we name our model as sliding language modeling.

### 3.3 Discussion

To better understand our SLM, we analyze the advantages of our SLM over other LMs. The comparisons are listed in Table 1. We select three representative LMs for comparisons, which are CLM (BERT), MLM (GPT) and bidirectional LM (Bi-LM, used in ELMO [42] [2]) respectively. From Table 1, we have the following observations:

1. When compared with CLM, our SLM requires 3 $\times$ computations. However, our SLM can fully use the whole sentence information for prediction while CLM can only use unidirectional information. Even scaling CLM as 3 $\times$ parameters, it still can not use bidirectional context for prediction. This also demonstrates the effectiveness of our SLM in using context.
2. MLM is powerful at extracting bidirectional context but it needs $n\times$ inferences to calculate the whole sentence information limited to its masked prediction. Our SLM just needs a single inference and uses bidirectional information for prediction with only $3\times$ computations. Especially, our SLM shows higher efficiency compared with MLM when $n$ is large.
3. Bi-LM can also extract forward and backward contextual information, but it just simply concatenates the forward and backward features for the final prediction, without any interactions. Instead, our SLM can iteratively fuse the bidirectional information thanks to our triple-stream self-attention mechanism.

Overall, the design of SLM is dedicated for sentence scoring, while CLM prefers NLG tasks and MLM/Bi-LM prefer NLU tasks.

## 4 Experiments

In this section, we describe our experimental setup, and the results on NMT and ASR datasets.

### 4.1 Experimental Setup

We adopt Transformer [38] as the backbone network. Following previous works [9], we adopt a base setting and a small setting for our model as Transcormer$_{base}$ with 110M parameters and Transcormer$_{small}$ with 34M parameters, that consists of 12/6 transformer layers and each layer has 768/512 hidden size and 12/8 attention heads. During the pre-training, we use wikipedia plus bookcorpus (16GB) as the training corpus, to be consistent with previous works [9]. Our model is

[2] ELMO pre-trains a left-to-right and a right-to-left LSTM [43] and concatenates the outputs of each last unidirectional LSTM layer for prediction.

| Model | IWSLT De | Es | It | Nl | Pl | Ro | Ru | Tr | WMT De-En |
|---|---|---|---|---|---|---|---|---|---|
| Oracle | 41.80 | 48.69 | 41.89 | 44.38 | 27.90 | 46.01 | 29.60 | 27.25 | 39.17 |
| Baseline | 34.77 | 41.20 | 34.95 | 37.73 | 22.67 | 38.73 | 24.21 | 21.65 | 32.54 |
| CLM (GPT) | 34.96 | 41.39 | 35.14 | 38.08 | 22.91 | 39.03 | 24.62 | 22.14 | 32.88 |
| MLM (BERT) | 35.14 | 41.54 | **35.54** | 38.14 | 23.00 | 39.21 | 24.65 | 22.36 | 33.07 |
| Bi-LM (ELMO) | 35.10 | 41.52 | 35.21 | 38.03 | 23.09 | 39.07 | 24.53 | 21.91 | 32.90 |
| SLM (Transcormer$_{base}$) | **35.24** | **41.86** | 35.52 | **38.45** | **23.29** | **39.34** | **24.69** | **22.41** | **33.10** |
| SLM (Transcormer$_{small}$) | 35.05 | 41.58 | 35.15 | 38.06 | 22.98 | 39.08 | 24.52 | 22.06 | 32.94 |

Table 2: Reranking results on IWSLT and WMT tasks, and all LMs have the same model architecture as Transcormer. The translation direction of all IWSLT tasks is to English and all results are reported in BLEU score. All LMs are pre-trained in the wikipedia + bookcorpus (16GB) with the same optimization. The last row is the oracle score from the generated candidates.

trained at the sentence-level (*i.e.*, one sentence per sample). We choose Adam [44] as the default optimizer with learning rate of $5e-4$, $\beta_1 = 0.9$, $\beta_2 = 0.98$ and $\epsilon = 1e-6$, and weight decay is set as 0.01. The learning rate warms up over the first 10,000 steps and then linearly decays. We set the batch size as 8192 tokens per batch, and the training step is 125,000 steps. We use 32 NVIDIA Tesla 32GB GPUs, with FP16 speedup. The total training needs 5.5 days for Transcormer$_{base}$. For more experimental settings (*e.g.*, dataset and its split sizes), please refer to the Appendix. The code and pre-trained models will be released later.

### 4.2 Experiments on Neural Machine Translation

We choose IWSLT14 dataset [45], which includes multiple small-scale translation tasks from different languages to English, and WMT14 English-German dataset [3] for evaluation. We adopt Transformer [38] as the machine translation model with 6-6 transformer layers to generate multiple candidates for reranking, with a beam size of 10. The hidden size and attention head are set as 512/1024 and 8/16 for IWSLT and WMT tasks respectively. During the reranking, we combine the original score produced by the machine translation model and the LM score with a hyper-parameter $\lambda$, by following previous experiences [6, 20]. The hyper-parameter $\lambda$ is tuned on dev set with a range of $[0.0, 2.0]$, and then select the best $\lambda$ to evaluate the test set. The results are reported in Table 2, in terms of BLEU [46]. Each task will be evaluated by five times based on different pre-trained checkpoints, and report the mean value, with a variance of 0.05. From Table 2, we have the following observations:

- Our Transcormer$_{base}$ can obtain better performance than CLM and Bi-LM [4] in both small-scale IWSLT and large-scale WMT tasks, which indicates the importance of bidirectional context for sentence scoring, and further validate the ability of our SLM in utilizing bidirectional information.
- When compared with MLM, our Transcormer$_{base}$ also achieves comparable results. Considering that the computation of MLM for scoring is linear to the input length and needs $n$ inference passes, our SLM show higher efficiency with only $3\times$ computations in a single pass to maintain three streams.

Overall, all of the results reveal that our model is more effective in using contextual information for probability estimation and more efficient with only a single forward pass, especially for long sentences.

### 4.3 Experiments on Automatic Speech Recognition

We choose LibrSpeech [47] to evaluate the performance of our model for reranking on ASR task. We train a Conformer model [48] on LibrSpeech, which has 12 encoder layers and 6 decoder layers with

[3] Here we only evaluate German→English direction as our model is trained on English domain.

[4] Here, we replace LSTM as Transformer in Bi-LM to keep the consistence in architecture.

| Model | dev-clean | dev-other | test-clean | test-other |
|---|---|---|---|---|
| Baseline | 2.80 | 6.90 | 3.06 | 7.05 |
| CLM (GPT) | 2.47 | 6.13 | 2.73 | 6.33 |
| MLM (BERT) | 2.30 | 5.65 | 2.59 | 5.90 |
| Bi-LM (ELMO) | 2.41 | 5.92 | 2.63 | 6.12 |
| SLM (Transcormer$_{base}$) | **2.23** | **5.54** | **2.49** | **5.72** |
| SLM (Transcormer$_{small}$) | 2.48 | 5.95 | 2.62 | 6.20 |
| Oracle | 1.45 | 4.23 | 1.59 | 4.19 |

Table 3: Reranking results on LibrSpeech dataset. All results are reported in WER.

512 hidden size and 8 attention heads, and the beam size is set as 10. In addition, we use SpecAug [49] as a data augmentation technology to further improve the accuracy of ASR system. We use word error rate (WER) to evaluate the performance of ASR tasks. We follow the same tuning technique used in NMT tasks for hyper-parameter $\lambda$, but with a larger range as $[0.0, 5.0]$. The results are reported in Table 3. From Table 3, we find that our model can give nearly 20% relative improvements over the baseline and also outperform other LMs, including CLM, MLM and Bi-LM. The results on ASR task further demonstrates the generalization and effectiveness of our SLM in sentence scoring.

# 5 Analyses

In this section, we conduct some method analyses on our proposed SLM and CLM/MLM. Besides, we also provide more analyses in Appendix.

## 5.1 Comparable Computation between CLM and SLM

As aforementioned, our SLM needs $3\times$ computations when compared with CLM. To make a fair comparison, we also pre-train a Transcormer$_{small}$ with 34M parameters in total, which consists 6 transformer layers and each layer has 512 hidden size and 8 attention heads. Hence, our Transcormer$_{small}$ has approximately $\frac{1}{3}$ parameters of Transcormer$_{base}$, and has the similar computations as GPT$_{base}$. We conduct experiments on three NMT tasks and an ASR task (LibriSpeech dataset) for comparisons and the results are listed in Table 2 and 3 (*i.e.*, "Transcormer$_{small}$ row"). We can find that even under the same computation, our model still outperforms CLM and this result further validates the necessity of using bidirectional context for sentence scoring. Besides, considering that Transcormer$_{small}$ has fewer parameters, our model is also friendly to the device deployments (*e.g.*, CPU).

| Model | Cost | PPL | dev-clean | dev-other | test-clean | test-other |
|---|---|---|---|---|---|---|
| Baseline | - | - | 2.80 | 6.90 | 3.06 | 7.05 |
| MLM ($k=1$) | $\times n$ | 4.26 | 2.30 | 5.65 | 2.59 | 5.90 |
| MLM ($k=2$) | $\times \lceil n/2 \rceil$ | 8.41 | 2.41 | 5.87 | 2.70 | 6.20 |
| MLM ($k=3$) | $\times \lceil n/3 \rceil$ | 11.58 | 2.60 | 5.95 | 2.87 | 6.41 |
| MLM ($k=\frac{n}{3}$) | $\times 3$ | - | 2.75 | 6.71 | 2.98 | 6.93 |
| MLM ($k=\frac{n}{2}$) | $\times 2$ | - | 2.80 | 6.80 | 3.01 | 6.99 |
| SLM | $\times 3$ | **3.85** | **2.23** | **5.54** | **2.49** | **5.72** |

Table 4: Comparisons of sampling different $k$ tokens for prediction on MLM. We choose ASR reranking task on LibriSpeech dataset to evaluate the results and also report PPL on a subset of sentences with same length ($n = 20$).

## 5.2 Varying Numbers of Forward Passes in MLM

As mentioned above, MLM needs to forward $n$ passes as each time only mask one token. So what will happen if we allow each pass to mask more tokens? Therefore, we design experiments that enforces MLM to forward $k$ tokens for prediction so that it only needs $\lceil n/k \rceil$ passes, and investigate

the effect of different $k$. For a certain $k$, we randomly split the sentence as $\lceil n/k \rceil$ sets, and each time masks one subset for prediction. The comparisons are listed in Table 4. We find that using larger $k$ will severely harm the performance, even if it can reduce the number of inference passes. When $k$ is set as $\frac{n}{3}$, which is equal to the cost of our SLM, it can hardly give any improvements over the baseline. We think that masking more tokens in the sentence will make it more difficult to estimate the token probability at a time. These comparisons also highlight the efficiency and effectiveness of our SLM for sentence scoring.

### 5.3 Sentence Scoring Quality at Each Position

Following previous experiences [20], we count the cross-entropy loss of each position in sentences for each LM, to better analyze the effectiveness of bidirectional context to estimate token probability. Specifically, we sample a subset of sentences as $\mathcal{S}$, and each sentence has the same token number $n$ (here $n$ is 20). For each position $i$, we count the average cross-entropy over the sampled subset $\mathcal{S}$ based on the output probability of each LM. The results are displayed in Figure 3. We can find that: 1) For CLM, the cross-entropy loss is higher at the first several positions and gradually decreases for the subsequent positions but is still higher than MLM and SLM, which indicates that only using undirectional information is not enough to measure the sentence score precisely. 2) SLM can almost obtain similar loss as MLM at each position. Considering SLM just needs a single pass while MLM needs $n$ passes, this phenomenon further validates the superiority and efficiency of SLM in scoring sentences.

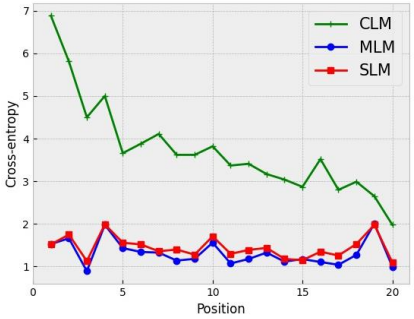



Figure 3: The average cross-entropy loss of each LM at each position. MLM uses $n$ passes to predict each position.

## 6 Conclusion

In this paper, we propose Transcormer, a Transformer with a novel sliding language modeling for sentence scoring. Specifically, our SLM is able to produce the probability of each token over the whole sentence within a single forward pass, and utilizes bidirectional context for prediction, and thus inherents the advantages of CLM and MLM and avoids their deficiencies. To the best of our knowledge, the proposed Transcormer is the first pre-trained language model tailored for sentence scoring. Experimental results on multiple datasets demonstrate the effectiveness of our Transcormer in computing sentence score for reranking tasks.

Besides, we summarize some potential directions of our Transcormer and SLM as the future works:

- Currently our Transcormer is only conducted on the English domain under the base setting, due to limited computation. We expect to develop large-scale Transcormer and use different language domains or multilingual data for training in the future.
- We design sliding language modeling for sentence scoring, and our experiments are mainly on reranking task. However, based on the characteristics of our SLM, we believe our model can also be used for other scenarios (*e.g.*, error correction [50, 51, 52], data selection), and we will explore the specific fine-tuning techniques when applying our SLM on different downstream tasks.
- Besides, our Transcormer mainly pre-trains SLM on an encoder framework. However, our SLM is not limited to model structure. For example, SLM can be easily extended to encoder-decoder framework [17] based on paired data. Therefore, we also expect to explore the possibility of using SLM on different frameworks.
- Although our paper mainly focuses on text data, we want to highlight that SLM can also be extended to other different modalities with sequential characteristic (*e.g.*, image, speech and time series data). Consequently, how to apply SLM to other modalities is also a valuable topic in the future.

# References


[1] Çaglar Gülçehre, Orhan Firat, Kelvin Xu, Kyunghyun Cho, Loïc Barrault, Huei-Chi Lin, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. On using monolingual corpora in neural machine translation. *CoRR*, abs/1503.03535, 2015.

[2] William Chan, Navdeep Jaitly, Quoc V. Le, and Oriol Vinyals. Listen, attend and spell: A neural network for large vocabulary conversational speech recognition. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2016, Shanghai, China, March 20-25, 2016*, pages 4960–4964. IEEE, 2016.

[3] Alex Warstadt, Alicia Parrish, Haokun Liu, Anhad Mohananey, Wei Peng, Sheng-Fu Wang, and Samuel R. Bowman. Blimp: The benchmark of linguistic minimal pairs for english. *Trans. Assoc. Comput. Linguistics*, 8:377–392, 2020.

[4] Nathan Ng, Kyra Yee, Alexei Baevski, Myle Ott, Michael Auli, and Sergey Edunov. Facebook fair's WMT19 news translation task submission. In *Proceedings of the Fourth Conference on Machine Translation, WMT 2019, Florence, Italy, August 1-2, 2019 - Volume 2: Shared Task Papers, Day 1*, pages 314–319. Association for Computational Linguistics, 2019.

[5] Yingce Xia, Xu Tan, Fei Tian, Fei Gao, Di He, Weicong Chen, Yang Fan, Linyuan Gong, Yichong Leng, Renqian Luo, Yiren Wang, Lijun Wu, Jinhua Zhu, Tao Qin, and Tie-Yan Liu. Microsoft research asia's systems for WMT19. In *Proceedings of the Fourth Conference on Machine Translation, WMT 2019, Florence, Italy, August 1-2, 2019 - Volume 2: Shared Task Papers, Day 1*, pages 424–433. Association for Computational Linguistics, 2019.

[6] Kyra Yee, Yann N. Dauphin, and Michael Auli. Simple and effective noisy channel modeling for neural machine translation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 5695–5700. Association for Computational Linguistics, 2019.

[7] Yuchen Liu, Long Zhou, Yining Wang, Yang Zhao, Jiajun Zhang, and Chengqing Zong. A comparable study on model averaging, ensembling and reranking in NMT. In *Natural Language Processing and Chinese Computing - 7th CCF International Conference, NLPCC 2018, Hohhot, China, August 26-30, 2018, Proceedings, Part II*, volume 11109 of *Lecture Notes in Computer Science*, pages 299–308. Springer, 2018.

[8] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. 2018.

[9] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*, pages 4171–4186, 2019.

[10] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

[11] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, pages 1–16, virtual, 2020. Curran Associates, Inc.

[12] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692, 2019.

[13] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. ALBERT: A lite BERT for self-supervised learning of language representations. In *8th International Conference on Learning Representations, ICLR 2020*, Addis Ababa, Ethiopia, 2020. OpenReview.net.

[14] Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. Unified language model pre-training for natural language understanding and generation. In *NeurIPS*, pages 13042–13054, Vancouver Convention Centre, Vancouver, Canada, 2019. Curran Associates, Inc.

[15] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. In *NeurIPS*, volume 32, pages 5753–5763, Vancouver Convention Centre, Vancouver, Canada, 2019.

[16] Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. Mpnet: Masked and permuted pre-training for language understanding. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020*, virtual, 2020. Curran Associates Inc.

[17] Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. MASS: Masked sequence to sequence pre-training for language generation. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97, pages 5926–5936, Long Beach Convention Center, Long Beach, 2019. PMLR.

[18] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *ACL*, pages 7871–7880, Online, 2020. Association for Computational Linguistics.

[19] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*, 2019.

[20] Julian Salazar, Davis Liang, Toan Q. Nguyen, and Katrin Kirchhoff. Masked language model scoring. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 2699–2712. Association for Computational Linguistics, 2020.

[21] Alex Wang and Kyunghyun Cho. BERT has a mouth, and it must speak: BERT as a markov random field language model. *CoRR*, abs/1902.04094, 2019.

[22] Joonbo Shin, Yoonhyung Lee, and Kyomin Jung. Effective sentence scoring method using BERT for speech recognition. In *Proceedings of The 11th Asian Conference on Machine Learning, ACML 2019, 17-19 November 2019, Nagoya, Japan*, volume 101 of *Proceedings of Machine Learning Research*, pages 1081–1093. PMLR, 2019.

[23] Shih-Hsuan Chiu and Berlin Chen. Innovative bert-based reranking language models for speech recognition. In *IEEE Spoken Language Technology Workshop, SLT 2021, Shenzhen, China, January 19-22, 2021*, pages 266–271. IEEE, 2021.

[24] Yue Weng, Sai Sumanth Miryala, Chandra Khatri, Runze Wang, Huaixiu Zheng, Piero Molino, Mahdi Namazifar, Alexandros Papangelis, Hugh Williams, Franziska Bell, and Gökhan Tür. Joint contextual modeling for ASR correction and language understanding. In *2020 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2020, Barcelona, Spain, May 4-8, 2020*, pages 6349–6353. IEEE, 2020.

[25] Shankar Kumar and William J. Byrne. Minimum bayes-risk decoding for statistical machine translation. In *Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics, HLT-NAACL 2004, Boston, Massachusetts, USA, May 2-7, 2004*, pages 169–176, 2004.

[26] Raphael Shu and Hideki Nakayama. Later-stage minimum bayes-risk decoding for neural machine translation. *CoRR*, abs/1704.03169, 2017.

[27] Libin Shen, Anoop Sarkar, and Franz Josef Och. Discriminative reranking for machine translation. In *Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics, HLT-NAACL 2004, Boston, Massachusetts, USA, May 2-7, 2004*, pages 177–184, 2004.

[28] Franz Josef Och. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics, 7-12 July 2003, Sapporo Convention Center, Sapporo, Japan*, pages 160–167, 2003.

[29] Yukun Ma, Erik Cambria, and Benjamin Bigot. ASR hypothesis reranking using prior-informed restricted boltzmann machine. In *Computational Linguistics and Intelligent Text Processing - 18th International Conference, CICLing 2017, Budapest, Hungary, April 17-23, 2017, Revised Selected Papers, Part I*, volume 10761, pages 503–514. Springer, 2017.

[30] Michael Auli and Jianfeng Gao. Decoder integration and expected BLEU training for recurrent neural network language models. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, Volume 2: Short Papers*, pages 136–142. The Association for Computer Linguistics, 2014.

[31] Terumasa Ehara. In *Proceedings of the 32nd Pacific Asia Conference on Language, Information and Computation: 5th Workshop on Asian Translation, WAT@PACLIC 2018, Hong Kong, December 1-3, 2018*. Association for Computational Linguistics, 2018.

[32] Yuguang Wang, Shanbo Cheng, Liyang Jiang, Jiajun Yang, Wei Chen, Muze Li, Lin Shi, Yanfeng Wang, and Hongtao Yang. Sogou neural machine translation systems for WMT17. In *Proceedings of the Second Conference on Machine Translation, WMT 2017, Copenhagen, Denmark, September 7-8, 2017*, pages 410–415. Association for Computational Linguistics, 2017.

[33] Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. Pre-training transformers as energy-based cloze models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 285–294. Association for Computational Linguistics, 2020.

[34] Joongbo Shin, Yoonhyung Lee, Seunghyun Yoon, and Kyomin Jung. Fast and accurate deep bidirectional language representations for unsupervised learning. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 823–835. Association for Computational Linguistics, 2020.

[35] Alexei Baevski, Sergey Edunov, Yinhan Liu, Luke Zettlemoyer, and Michael Auli. Cloze-driven pretraining of self-attention networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 5359–5368. Association for Computational Linguistics, 2019.

[36] Sumanta Bhattacharyya, Amirmohammad Rooshenas, Subhajit Naskar, Simeng Sun, Mohit Iyyer, and Andrew McCallum. Energy-based reranking: Improving neural machine translation using energy-based models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pages 4528–4537. Association for Computational Linguistics, 2021.

[37] Ann Lee, Michael Auli, and Marc'Aurelio Ranzato. Discriminative reranking for neural machine translation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pages 7250–7264. Association for Computational Linguistics, 2021.

[38] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, pages 5998–6008, Long Beach Convention Center, Long Beach, 2017. Curran Associates, Inc.

[39] Weizhen Qi, Yu Yan, Yeyun Gong, Dayiheng Liu, Nan Duan, Jiusheng Chen, Ruofei Zhang, and Ming Zhou. Prophetnet: Predicting future n-gram for sequence-to-sequence pre-training. In *Findings of the Association for Computational Linguistics: EMNLP 2020, Online Event, 16-20 November 2020*, volume EMNLP 2020 of *Findings of ACL*, pages 2401–2410. Association for Computational Linguistics, 2020.

[40] Dongling Xiao, Han Zhang, Yu-Kun Li, Yu Sun, Hao Tian, Hua Wu, and Haifeng Wang. ERNIE-GEN: an enhanced multi-flow pre-training and fine-tuning framework for natural language generation. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*, pages 3997–4003. ijcai.org, 2020.

[41] Jungo Kasai, James Cross, Marjan Ghazvininejad, and Jiatao Gu. Non-autoregressive machine translation with disentangled context transformer. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 5144–5155. PMLR, 2020.

[42] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pages 2227–2237. Association for Computational Linguistics, 2018.

[43] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, 1997.

[44] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[45] Mauro Cettolo, Jan Niehues, Sebastian Stüker, Luisa Bentivogli, and Marcello Federico. Report on the 11th IWSLT evaluation campaign. In *Proceedings of the 11th International Workshop on Spoken Language Translation: Evaluation Campaign*, pages 2–17, Lake Tahoe, California, dec 2014.

[46] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, July 6-12, 2002, Philadelphia, PA, USA*, pages 311–318. ACL, 2002.

[47] Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. Librispeech: An ASR corpus based on public domain audio books. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2015, South Brisbane, Queensland, Australia, April 19-24, 2015*, pages 5206–5210. IEEE, 2015.

[48] Anmol Gulati, James Qin, Chung-Cheng Chiu, Niki Parmar, Yu Zhang, Jiahui Yu, Wei Han, Shibo Wang, Zhengdong Zhang, Yonghui Wu, et al. Conformer: Convolution-augmented transformer for speech recognition. In *INTERSPEECH*, 2020.

[49] Daniel S Park, William Chan, Yu Zhang, Chung-Cheng Chiu, Barret Zoph, Ekin D Cubuk, and Quoc V Le. Specaugment: A simple data augmentation method for automatic speech recognition. In *INTERSPEECH*, 2019.

[50] Yichong Leng, Xu Tan, Linchen Zhu, Jin Xu, Renqian Luo, Linquan Liu, Tao Qin, Xiangyang Li, Edward Lin, and Tie-Yan Liu. Fastcorrect: Fast error correction with edit alignment for automatic speech recognition. *Advances in Neural Information Processing Systems*, 34, 2021.

[51] Yichong Leng, Xu Tan, Rui Wang, Linchen Zhu, Jin Xu, Wenjie Liu, Linquan Liu, Xiang-Yang Li, Tao Qin, Edward Lin, et al. Fastcorrect 2: Fast error correction on multiple candidates for automatic speech recognition. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 4328–4337, 2021.

[52] Kaitao Song, Xu Tan, and Jianfeng Lu. Neural machine translation with error correction. In *Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence*, pages 3891–3897, 2021.

[53] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *EMNLP*, pages 1631–1642, oct 2013.

[54] Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *CoRR*, abs/2107.13586, 2021.

# Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] Please see Section refsubsec:nmt and Appendix.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] Please see Section 4.1
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]

(d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]

5. If you used crowdsourcing or conducted research with human subjects...

(a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

# A Appendix

## A.1 Experimental Setup

### A.1.1 Datasets

**IWSLT 2014** is the evaluation campaign of the 11th International Workshop on Spoken Language Translation. It consist of a lot of small-scale translation tasks collected from TED talks, including German (De), Spanish (Es), Italian (It), Dutch (NL), Polish (PL), Romanian (Ro), Russian (Ru), Turkish (Tr) to English. We randomly split each dataset as the training set and dev set with a ratio of 25:1. And each task concatenates TED.tst2010, TED.tst2011, TED.dev2010 and TED.tst2012 as the test set. The statistics of each sub-task is described as:

| | De | Es | It | Nl | Pl | Ro | Ru | Tr |
|---|---|---|---|---|---|---|---|---|
| Train | 160K | 169K | 167K | 153K | 128K | 167K | 153K | 109K |
| Valid | 7.2K | 7.6K | 7.5K | 6.9K | 5.8K | 7.6K | 6.9K | 4.9K |
| Test | 5.5K | 5.5K | 5.5K | 5.3K | 5.4K | 5.5K | 5.5K | 5.4K |

Table 5: Statistical of IWSLT datasets.

**WMT14 English-German** comprises 4.5M bilingual data collected from Europarl v7, Common Crawl corpus and News Commentary. We concatenate newstest2012 and newstest2013 as the valid set, and choose newstest2014 as the test set for WMT14 English-German. Our experiments mainly focuses on German→English.

**LibriSpeech** [47] includes 1000hr speech data, sampled at 16k Hz. LibriSpeech includes four subsets for evaluation, which are dev-clean, dev-other, test-clean and test-other.

| Hyper-parameter | Transcormer$_{base}$ | Transcormer$_{small}$ |
|---|---|---|
| Number of Layers | 12 | 6 |
| Hidden Size | 768 | 512 |
| Filter Size | 3072 | 2048 |
| Attention heads | 12 | 8 |
| Dropout | 0.1 | 0.1 |
| Weight Decay | 0.01 | 0.01 |
| Learning Rate | 5e-4 | 5e-4 |
| Steps | 125K | 125K |
| Batch | 8192 | 8192 |

Table 6: Pre-training hyper-parameters for Transcormer$_{base}$ and Transcormer$_{small}$.

| Model | Overall | ANA. AGR | ARG. STR | BINDING | CTRL. RAIS. | D-N AGR | ELLIPSIS | FILLER GAP | IRREGULAR | ISLAND | NPI | QUANTIFIERS | S-V AGR |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| GPT-2 (345M) | 82.6 | **99.4** | 83.4 | 77.8 | 83.0 | 96.3 | 86.3 | 81.3 | 94.9 | 71.7 | 74.7 | **74.1** | 88.3 |
| BERT (base) | 84.2 | 97.0 | 80.0 | 82.3 | 79.6 | **97.6** | 89.4 | 83.1 | 96.5 | 73.6 | 84.7 | 71.2 | **92.4** |
| BERT (large) | 84.8 | 97.2 | 80.7 | 82.0 | 82.7 | **97.6** | 86.4 | 84.3 | 92.8 | 77.0 | 83.4 | 72.8 | 91.9 |
| RoBERTa (base) | 85.4 | 97.3 | 83.5 | 77.8 | 81.9 | 97.0 | **91.4** | **90.1** | 96.2 | 80.7 | 81.0 | 69.8 | 91.9 |
| RoBERTa (large) | **86.5** | 97.8 | **84.6** | 79.1 | **84.1** | 96.8 | 90.8 | 88.9 | **96.8** | **83.4** | 85.5 | 70.2 | 91.4 |
| Transcormer (base) | 84.6 | 98.1 | 80.7 | 83.2 | 80.2 | 96.0 | 90.7 | 84.1 | 95.5 | 74.3 | 85.7 | 73.4 | 91.3 |
| + BERT Init, 20K steps | 85.0 | 98.1 | 80.0 | **84.8** | 79.2 | 96.2 | 89.3 | 84.6 | 96.1 | 76.6 | **87.8** | **75.0** | 91.7 |

Table 7: Results on BLiMP. The results on GPT-2, BERT and RoBERTa are taken from [20]. The "+ BERT Init, 20K steps" means the Transcormer model uses BERT model for initialization and then trains in SLM with 20K steps (nearly 5 epochs).

### A.1.2 Hyper-parameter Setup

The pre-training hyper-parameters of Transcormer are described in Table 6.

## A.2 More Details about Related Works

As mentioned in Section 2.1, some works [21, 20, 33, 34] tried to alleviate the efficient problem in MLM model caused by N-passes. Specifically, [21] proposed to calculate pseudo log-likelihood score via stochastic estimation, that is randomly sampling K tokens and computing the probability of these K tokens via masked prediction as the final sentence probability. It can reduce the time complexity from $O(|\mathbf{x}|)$ to $O(K)$ but will harm model performance. [20] also suggested a distillation strategy to cover this problem, that requires model to compute sentence score via N-passes first (*i.e.*, teacher model) and then distills it to the output vector of the $[\text{CLS}]$ token (*i.e.*, student model) during the pre-training. However, this paradigm also will under-perform regular LMs in their experiments [20]. [34] designed a model named DLM to produce token-wise probability via a single inference pass. To fulfill this target, DLM only feeds word embeddings as the key/value for each Transformer layer, rather than the previous layer. Such design allows the query stream to capture the whole sentence information but without any contextualized semantics. Electric [33] is a model built upon a Two-Cloze Tower [35], based on noise contrastive estimation. More in details, Electric trained a left-to-right Transformer and a right-to-left Transformer and then concatenated together at the final layer to predict each token. Just as discussed in Section 3.3, this model learns forward and backward context individually and only fuse semantics at the final layer while the query stream of our SLM is able to fuse bidirectional context iteratively. Overall, our SLM can make full use of bidirectional information over all Transformer layers and predict token-wise probabilities simultaneously.

Besides, there also remain some works [36, 37] which use discriminative language modeling to approximately estimate sentence scores. [36] borrowed the idea of the energy-based model into sentence reranking. [37] proposed a discriminative language model that minimizes the KL-divergence between the target distribution and the output distribution. These methods can be considered as the discriminative language modeling which directly predicts a single value to be the sentence score. Discriminative language model usually needs target datasets for fine-tuning, while our proposed language model is independent to downstream tasks. We think discriminative language models are complementary to our works and we leave this combination as the future works.

## A.3 Results

### A.3.1 Results on BLiMP

Following previous experiences [20], we also conduct experiments on Benchmark of Linguistic Minimal Pairs (BLiMP) [3], which includes 67K minimal pairs that contrast in grammatical acceptability and isolate specific phenomena in syntax, morphology or semantics. BLiMP provides an

| Model | Domain | Dev | | Test | |
|---|---|---|---|---|---|
| | | clean | other | clean | other |
| Baseline [22] | - | 7.17 | 19.79 | 7.26 | 20.37 |
| Electric [33] | wikibooks | - | - | 5.65 | 17.42 |
| T-TA [34] | Libri | 4.98 | 16.09 | 5.11 | 16.91 |
| GPT-2 (117M) | openwebtext | 5.39 | 16.81 | 5.64 | 17.60 |
| GPT-2 (345M) | openwebtext | 5.15 | 16.48 | 5.30 | 17.26 |
| BERT (base) | wikibooks | 5.17 | 16.44 | 5.41 | 17.41 |
| RoBERTa (base) | RoBERTa [12] | 5.03 | 16.16 | 5.25 | 17.18 |
| Transcormer (base) | wikibooks | 5.09 | 16.30 | 5.28 | 17.31 |
| + BERT Init, 20K steps | wikibooks | 5.10 | 16.27 | 5.19 | 17.20 |
| Transcormer (base , 20K steps) | Libri | 4.61 | 15.51 | 4.73 | 16.45 |

Table 8: WERs on LibriSpeech after rescoring. We evaluate the results on the dataset shared by [22], and the results of baseline, GPT and BERT are taken from [22]. RoBERTa adopts a large-scale pre-training corpus with 160GB.

unsupervised setting that uses language models to evaluate sentences and the acceptable sentence can be assigned by a lower log-likelihood score. We compare our model with GPT-2, BERT and RoBERTa, and the results are reported in Table 7. We can find that our Transcormer can easily beat GPT model in BLiMP, which further validates the effectiveness of our SLM in using bidirectional context for sentence scoring. And when compared with BERT model, our Transcormer can also match or outperform BERT performance slightly. Considering that our model only needs a single pass to produce the probability of all tokens, which also manifests the efficiency of our SLM in sentence scoring.

Besides, we also report RoBERTa results for reference. Due to the limitation of resources, our model currently cannot match RoBERTa since RoBERTa costs more computations than BERT so it can extract better semantics for prediction. But we believe our Transcormer can also achieve the better performance by training our model with the same computations.

### A.3.2 Comparison with other works

As aforementioned, previous works [34, 33] have tried some strategies to calculate the probabilities of all tokens simultaneously to avoid the limitations in N-passes. To validate the advantages of our model in using bidirectional context, we also conduct experiments to make a comparison with these methods. For the sake of fairness, all of our experiments are deployed on the same datasets used in [22] and the results are shown in Table 8. From Table 8, we find that our Transcormer can outperform these models. These improvements also demonstrate the superiority of our model in utilizing bidirectional context to predict sentence probability.

## A.4 Accelerating SLM Training

Although SLM is more advantageous than CLM in using bidirectional context and demonstrates higher efficiency than MLM for sentence scoring, directly training SLM from scratch is still time-consuming since it needs $3\times$ computations to maintain query and forward/backward streams. So is it possible to accelerate the training of SLM? We note that both SLM and MLM adopt the masked token plus its position to predict its target based on the context while the main difference in context is that MLM adopts one bidirectional context and SLM adopts forward and backward contexts. We think that MLM-based model with bidirectional context should also be able to produce good unidirectional context and SLM does not modify model structure, so that we can use the MLM-based model (e.g., BERT) for initialization to accelerate SLM training. Specifically, we continue to train BERT model with addition 5 epochs (nearly 20K steps) and the results can be found in Table 7 and Table 8. We can find that only needs 5 epochs, our Transcormer with BERT model initialization can match the

performance that is trained from scratch, which means using BERT initialization can accelerate our SLM training.

### A.5 Latency

To better manifest the efficiency of our SLM in rescoring when compared with MLM, we further test the inference latency between SLM and MLM at both GPU and CPU under different lengths of the input sequence. Specifically, we measure the inference latency of each model at a batch size of 1 to validate the efficiency, and all results are reported in Table 9. From Table 9, we can find that MLM (BERT) produces very high latency when compared with SLM (Transcormer), especially in CPU devices. Even using a smaller BERT model, it still cannot avoid the inherent issues (*i.e.*, requires $n\times$ computations) in MLM for scoring, which needs $20\times$ and $166\times$ additional computations over Transcormer (SLM) in GPU and CPU. These results also demonstrate the efficiency of our model in using bidirectional context for calculating sentence scores.

| Model | #Params | # Sent = 10 | | # Sent = 100 | | # Sent = 500 | |
|---|---|---|---|---|---|---|---|
| | | GPU | CPU | GPU | CPU | GPU | CPU |
| BERT (small) | 34M | 53ms | 12.87s | 502ms | 317s | 3658ms | 1,650s |
| BERT (base) | 110M | 76ms | 27.06s | 750ms | 703s | 7890ms | 3,210s |
| BERT (large) | 340M | 135ms | 59.91s | 1390ms | 1676s | 19770ms | 7,433s |
| Transcormer (small) | 34M | 59ms | 3.45s | 71ms | 7.64s | 183ms | 9.90s |
| Transcormer (base) | 110M | 67ms | 6.29s | 97ms | 17.85s | 246ms | 20.03s |

Table 9: Inference latency between BERT (MLM) and Transcormer (SLM) at different sequence lengths. "# Sent" means the length of input sequence for evaluation. GPU is evaluated at NVIDIA Tesla V100-SXM2-16GB and CPU is at Intel (R) Xeon (R) Platinum 8168 CPU @ 2.70GHz. The units of numbers in "GPU" and "CPU" columns are at millisecond (ms) and second (s).

### A.6 Analyses

#### A.6.1 Pre-training Strategy

In our experiment setup, we use sentence-level data as the input for pre-training. To better analyze the effect of using different data processing for pre-training, we conduct experiments by using stream-level data (concatenate multiple sentences as a fixed-length, *e.g.*, 512) to make a comparison. We apply two strategies on NMT & ASR tasks, and then evaluate the average (top-1) accuracy of each token in a sentences with different lengths based on the output probability for SLM. The results are reported in Table 10. We can find that using stream-level data for pre-training can not achieve good accuracy when sentence length is too short. We guess that is because using stream-level data causes model can not fit short sentence since it always pre-trains under the longer sentences (*i.e.*, 512). Considering that our downstream scenarios mainly consist of single sentence, which is usually too short, directly using stream-level data for pre-training can not achieve promising performance. As a result, we recommend to use sentence-level data for pre-training, and we also expect to explore more effective pre-training strategies in the future.

| Model | IWSLT | | WMT | LibriSpeech | | # Sent Len | | |
|---|---|---|---|---|---|---|---|---|
| | De | Es | De-En | dev-clean | dev-other | 20 | 250 | 500 |
| Transcormer | 35.24 | 41.86 | 33.10 | 2.23 | 5.54 | 60.0% | 73.0% | 78.8% |
| Using stream-level | 34.84 | 41.38 | 32.70 | 2.56 | 6.31 | 20.0% | 55.0% | 78.5% |

Table 10: Comparisons between sentence-level and stream-level pre-training. The translation direction of all IWSLT tasks is to English. We sample some sentences from wikipedia with the same length (*e.g.*, 20, 250, 500) to evaluate the token accuracy of SLM in sentences (*i.e.*, obtain the top-1 accuracy of each token and calculate the sentence accuracy by averaging the accuracy of all tokens).

### A.6.2 Domain Adaption

Following previous experiences [20], we also study the effect of using in-domain data for pre-training. For NMT tasks, we randomly sample 20GB monolingual data from NewsCrawl data to build the pre-training corpus for pre-training. And for ASR tasks, as LibriSpeech includes 4GB in-domain data, we direct use this data as our pre-training corpus to handle ASR tasks. The results of NMT and ASR tasks are reported in Table 11 and Table 12. We can find that using in-domain data for pre-training is useful to improve the downstream tasks.

| | IWSLT | | | | | | | | WMT |
|---|---|---|---|---|---|---|---|---|---|
| Model | De | Es | It | Nl | Pl | Ro | Ru | Tr | De-En |
| Oracle | 41.80 | 48.69 | 41.89 | 44.38 | 27.90 | 46.01 | 29.60 | 27.25 | 39.17 |
| Baseline | 34.77 | 41.20 | 34.95 | 37.73 | 22.67 | 38.73 | 24.21 | 21.65 | 32.54 |
| SLM (Transcormer) | 35.24 | 41.86 | 35.52 | 38.45 | 23.29 | 39.34 | 24.69 | 22.41 | 33.10 |
| + in-domain data | **35.74** | **42.39** | **35.97** | **39.06** | **23.91** | **39.70** | **24.95** | **23.05** | **33.51** |

Table 11: Domain adaption on NMT tasks. The translation direction of all IWSLT tasks is to English. All results are reported in BLEU.

| Model | dev-clean | dev-other | test-clear | test-other |
|---|---|---|---|---|
| Oracle | 1.45 | 4.23 | 1.59 | 4.19 |
| Baseline | 2.80 | 6.90 | 3.06 | 7.05 |
| SLM (Transcormer) | 2.23 | 5.54 | 2.49 | 5.72 |
| + in-domain data | **2.01** | **5.12** | **2.12** | **5.23** |

Table 12: Domain adaption on LibrSpeech dataset. All results are reported in WER.

### A.7 Can SLM be used for Language Understanding Tasks?

As aforementioned in Section 3.3, both SLM and MLM can learn bidirectional context to predict token-wise probability, so why we cannot directly use SLM for language understanding tasks? First, a difference between MLM and SLM is that the query and content streams in the MLM-based model are shared together to enjoy the benefits of the bidirectional context, while our SLM maintains query and forward/backward contents. Previous state-of-the-art works [9, 12, 15, 13] have proven that BERT-style models can achieve state-of-the-art performance in NLU tasks due to the benefits of bidirectional context. Although the query stream is able to enjoy completely bidirectional context like MLM due to the design of the triple-stream self-attention mechanism, the content of SLM is decomposed as the forward and backward streams respectively to collect undirectional semantics, which may be weaker than MLM-based model in NLU tasks if using bidirectional context for fine-tuning. In our internal experiments, we have tried to directly fine-tune SLM on SST-2 [53] like BERT (*i.e.*, using bidirectional context). Due to the mismatch (*i.e.*, forward/backward context v.s. bidirection context) between pre-training and fine-tuning, our SLM only achieves 91.9% accuracy while the standard BERT can obtain 92.8%. This phenomenon may also validate our hypothesis that SLM is not the optimal method for solving NLU tasks in a BERT-style fine-tuning paradigm.

However, there still remain some alternative possibilities for SLM to adapt NLU scenarios. A possible solution is that converts language understanding tasks into ranking classification tasks. For example, assuming the task is sentiment classification and we have a review sentence, so we can construct two sentences: “the sentiment of [review] is positive" and “the sentiment of [review] is negative". Based on the constructed sentences, we can use our model on each sentence to calculate their sentence scores and the sentence with ground truth should have a lower log-probability score. It can be considered as a variant of prompt-based learning [11, 54]. We have simply built two different patterns to formulate sentiment classification, and conduct experiments on SST-2 in a zero-shot setting (*i.e.*,

| Prompt Pattern | | SST-2 |
|---|---|---|
| Accepted Sentence | Unaccepted Sentence | (zero-shot) |
| The sentiment of [**Review**] is [**GT Label**]. | The sentiment of [**Review**] is [**Wrong Label**]. | 58.0% |
| [**Review**] is a [**GT Label**] sentiment. | [**Review**] is a [**Wrong Label**] sentiment. | 71.0% |

Table 13: Examples of Ranking classification for solving sentiment classification. "[**Review**]" means the input sentence and "[**GT Label**]" means the real label of the input sentence and "[**Wrong Label**]" means other incorrect labels. The last column is the zero-shot results on SST.

without fine-tuning). The results are shown in Table 13. We find that using patterns like "[**Review**] is a [**GT Label**] sentiment." can obtain an accuracy of 71.0% on SST-2 in a zero-shot setting (BERT fine-tuning cannot be used in a zero-shot scenario). These experiments also demonstrate that we can transform NLU tasks into ranking tasks and then apply our model for ranking, which also indicates the potential of our model in solving NLU or other NLP tasks. We will also continue to explore more techniques to refine this strategy to improve performance in the future.