COOL, a Context Outlooker, and Its Application to Question Answering and Other Natural Language Processing Tasks

Fangyi Zhu, See-Kiong Ng and Stéphane Bressan
National University of Singapore
zhufy1030@gmail.com, {seekiong, steph}@nus.edu.sg

Abstract
Vision outlooker improves the performance of vision transformers, which implements a self-attention mechanism by adding an outlook attention, a form of local attention.

In natural language processing, as has been the case in computer vision and other domains, transformer-based models constitute the state-of-the-art for most processing tasks. In this domain, too, many authors have argued and demonstrated the importance of local context.

We present an outlook attention mechanism, COOL, for natural language processing. COOL, added on top of the self-attention layers of a transformer-based model, encodes local syntactic context considering word proximity and more pairwise constraints than dynamic convolution used by existing approaches.

A comparative empirical performance evaluation of an implementation of COOL with different transformer-based models confirms the opportunity for improvement over a baseline using the original models alone for various natural language processing tasks, including question answering. The proposed approach achieves competitive performance with existing state-of-the-art methods on some tasks.

1 Introduction
Transformer neural networks, based solely on the self-attention mechanism [Vaswani et al., 2017], have been shown to be remarkably successful in handling various tasks in natural language processing (NLP) [Lan et al., 2019; He et al., 2021] and other domains [Dosovitskiy et al., 2021; Sun et al., 2019; Dong et al., 2018]. Self-attention, in a nutshell, considers every token in the input sequence when encoding a current token. This strategy allows the learning and capture of cross-passage long-range dependencies.

The interrogation naturally arises of whether self-attention is detrimental to local information and of how, if the case, this flaw can be remedied. Several authors have considered and evaluated various ways of incorporating local context information back into a self-attention mechanism for various tasks in various domains [Wu et al., 2019; Gulati et al., 2020; Yuan et al., 2021; Xiao et al., 2021].

Recently, Yuan et al. [2021] proposed Vision Outlooker (VOLO) for visual recognition. VOLO successfully improves the performance of vision transformers by adding a lightweight outlook attention mechanism, a form of local attention.

We present and evaluate the outlook attention mechanism for NLP, named Context Outlooker (COOL). COOL implements a context outlook attention mechanism that encodes the local syntactic context considering word proximity and can be inserted into most transformer-based models. Compared to the operations for local augmentation used by existing works, outlook attention considers more pair-wise constraints as it processes multiple local windows containing an anchor token. We consider three possible modes for the composition of global and local encoding, in which the global encoding precedes, Global-to-Local, follows, Local-to-Global, or is parallel to, Global-and-Local, the local encoding.

Comparative empirical performance evaluation of COOL with different transformer-based approaches confirms the opportunity for improvement over a baseline using the original models for eleven NLP tasks, including question answering, question generation, sentiment analysis, etc.
2 Related Work

2.1 Transformer-based Language Models

Vaswani et al. [2017] first proposed the Transformer architecture, based solely on an attention mechanism, for neural machine translation (NMT). Since Bidirectional Encoder Representation from Transformer (BERT) [Devlin et al., 2019] outstandingly tackled eleven NLP tasks, fine-tuning a transformer-based language model has become the mainstream approach for most NLP tasks. The transformer-based structure has been explored for other domains as well and achieved competitive performance with classical convolutional neural networks (CNN)- or recurrent neural network (RNN)-based models, such as Vision Transformer (ViT) for computer vision (CV) [Dosovitskiy et al., 2021], Bidirectional Encoder Representations from Transformers for Semantic Speech Recognition (ASR) [Sun et al., 2021], and Speech-Transformer for automatic speech recognition (ASR) [Dong et al., 2018].

The self-attention mechanism, an operation for cross-passage encoding derived from [Cheng et al., 2016], is the crucial module in a transformer-based architecture. It relates all the elements in a sequence to capture long-range dependencies [Bahdanau et al., 2014]. Nonetheless, there are two drawbacks of long-range encoding: the computation is expensive, especially for long sequences, and the cross-passage encoding might dilute the local information within the surrounding context. To address these issues, recent works from different domains proposed to exploit an adaptive span [Hofstätter et al., 2020; Yang et al., 2018; Sukhbaatar et al., 2019], or augment local encoding, e.g., [Wu et al., 2019; Zhang et al., 2020; Jiang et al., 2020] for NLP, [Liu et al., 2021b; Yuan et al., 2021] for CV, and [Gulati et al., 2020] for ASR.

2.2 Local Attention

The dilution of local information in a standard self-attention mechanism for NLP and the benefit of its reinstatement was first discussed for NMT by Wu et al. [2019]. They presented the dynamic convolution to replace global encoding with the self-attention layers. They demonstrated that only encoding the local context could outperform executing a global encoding via self-attention.

Several works followed that tried to integrate both local, leveraging the dynamic convolution technique, and global encoding, leveraging the self-attention mechanism. Zhang et al. [2020] presented a novel module where a dynamic convolution and a self-attention mechanism were placed in parallel for NMT. Jiang et al. [2020] further connected a self-attention mechanism and a dynamic convolution in series to form a mixed-attention mechanism.

Recently, in the area of CV, Yuan et al. [2021] and Xiao et al. [2021] demonstrated that a significant factor limiting the performance of the transformer-based fashion for visual tasks was the low efficacy of the self-attention mechanism in encoding the local features as well. Moreover, Yuan et al. [2021] proposed VOLO for efficient pixel token aggregation via an outlook attention mechanism, with which a transformer-based model could outperform the classical CNN-based models on visual recognition and segmentation. Compared to the standard dynamic convolution technique, handling a single constraint within the current window only, the outlook attention mechanism deals with more pairwise constraints for the current token encoding. Specifically, it considers all local windows containing the current token.

3 Methodology

We present COOL, an outlook attention mechanism and architecture, that helps emphasise local information in NLP using a transformer-based model.

For the sake of clarity, we present the proposed model for a specific task of extractive question answering.

3.1 Architecture

Yuan et al. [2021] argued the importance of local information for visual tasks and proposed VOLO to overcome the weakening of local information in representations produced by the standard self-attention mechanism. The outlook attention mechanism, the primary component of VOLO, dynamically fine-tunes pixel token features produced by the self-attention mechanism considering local information from surrounding pixel tokens.

We present to generalise this idea into NLP as the context outlooker (COOL). The overall architecture has three major modules, global encoding by a generic transformer-based encoder, local encoding via the proposed context outlooker, and output layers for the downstream tasks. The output layers for an extractive question answering task are for answer prediction.

There are three different modes for integrating global and local encoding illustrated in Figure 1,

- **Global-to-Local (G2L):** first encodes the question and passage by the global encoder. Then, the yielded global embeddings go through the context outlooker to enrich the local context information. Finally, the start and end positions of the answer are predicted in the output layers.
- **Local-to-Global (L2G):** switches the order of global encoding and local augmentation in the first structure. In contrast to G2L, L2G first conducts local encoding, followed by global encoding.
- **Global-and-Local (G&L):** executes global and local encoding parallelly and fuses the integrated global and local embeddings with several linear layers. Finally, predict the answer locations.

Next, we elaborate the workflow in light of the first structure of G2L in Figure 2.

3.2 Global Encoder

Given a question \( q = \{q^1, ..., q^M\} \) and a passage \( p = \{p^1, ..., p^N\} \), \( M, N \) denotes the total number of tokens in the question and passage. The global encoder, as a backbone of our proposed system, can be the encoder in any transformer-based model, e.g., BERT [Devlin et al., 2019], ALBERT [Lan et al., 2019], and RoBERTa [Zhuang et al., 2021] with an
encoder-only architecture, or ProphetNet [Qi et al., 2020] with an encoder-decoder architecture.

\[ h_g = \text{Global}(\{q; p\}), \]  

(1)

where \( h_g \in \mathbb{R}^{L \times H} \) is the obtained global representations, \( L \) is the max length of the input sequence, and \( H \) is the size of hidden units in the last layer of the global encoder.

### 3.3 Context Outlooker

Context outlooker, the pivotal component in our proposed approach, consists of a convolutional block and a context outlook block.

#### Convolutional Block

The convolution operation is effective and stable for capturing local information [Kalchbrenner et al., 2014]. Several works have proved that inserting convolutional layers can help transformers in applications to different domains [Wu et al., 2019; Xiao et al., 2021; Karpov et al., 2020; Liu et al., 2021a; Gulati et al., 2020]. We thus try to insert a convolutional block for local augment, and further compare the convolution technique to our proposed module.

We stack the convolutional block, consisting of several convolutional layers, on top of the global encoder,

\[ h_c = \text{ReLU}(\text{ConvBlock}(h_g)), \]  

(2)

\[ h_c = [h_c^1, ..., h_c^d, ..., h_c^D], \]  

(3)

\[ h_c = \text{Concat}(\text{AdaptivePool}(h_c)). \]  

(4)

In Eq. (3), \( h_c^d \) is the output of the \( d \)-th convolutional layer with ReLU non-linear activation operation [Glorot et al., 2011], and \( D \) is the number of convolutional layers in the block. The dimension of \( h_c^d \) is \( L^d \times F^d \). \( L^d \) depends on the kernel size and padding size of the convolutional layer, and \( F^d \) is the number of filters in the convolutional layer. To align different \( L^d \) to original length \( L \), we perform adaptive pooling after each convolutional layer interpreted in Eq.(4). Finally, the outputs from all convolutional layers are concatenated along the second dimension to produce \( h_c \in \mathbb{R}^{L \times F} \).

\[ F = \sum_{1 \leq d < D} F^d. \]  

(5)

Unfold extracts sliding local blocks from a batched input tensor. Fold, the reverse operation of unfold, combines an array of sliding local blocks into the original large containing tensor.

#### Context Outlook Block

The context outlook block, consisting of several context outlook layers, follows by the convolutional block.

In analogy with the feature map of an image, each row in the representation matrix \( \hat{h}_c \) is the representation of a token. To maintain the integrality of representation of each token, we adapt the local window size in the visual outlook attention mechanism to \( K \times F \) to handle text tokens instead of \( K \times K \) for processing image pixel tokens.

Similar to the computation of \( Q \), \( K \), \( V \) matrices in the self-attention mechanism, the outlook attention mechanism first produces value matrix \( v \in \mathbb{R}^{L \times F} \) via a linear layer with \( \hat{h}_c \) from previous convolutional block,

\[ v = W_v \hat{h}_c + b_v, \]  

(6)

where \( W_v \) and \( b_v \) separately denote the parameters and bias needed to be learned in the linear layer.

Hence, for the \( i \)-th token, the value matrices of its neighbour tokens in a local window with the size of \( K \times F \) are,

\[ v^\Delta_i = v_{i+r-\lfloor \frac{K}{2} \rfloor}, 0 \leq r < K, \]  

(7)

\( v^\Delta_i \in \mathbb{R}^{F \times K} \), and \( \lfloor \cdot \rfloor \) is the round down formula.

In visual outlook attention, Yuan et al. [2021] proposed to produce the aggregating weights directly from the features of the centric token to dispense the expensive dot product computation in self-attention, as each spatial location in an image feature map can represent its close neighbours. Likewise, the embedding of a token in a sentence is able to express its neighbour tokens as well. Therefore, following their work, we calculate attention matrix \( a \in \mathbb{R}^{L \times A} \) with \( \hat{h}_c \) via another linear layer,

\[ a = W_a \hat{h}_c + b_a, \]  

(8)

here \( A = K \times K \times F \). \( W_a \) and \( b_a \) are learned parameters and bias. Then, the attention weights within different local windows are obtained by reshaping \( a \) from \( \mathbb{R}^{L \times A} \) to \( \mathbb{R}^{L \times K \times (K \times F)} \), \( a^\Delta_i \in \mathbb{R}^{1 \times 1 \times (K \times F)} \) represents the obtained attention weights of \( i \)-th token within a local window.

Thereewith, normalise the attention weights in each local window with a softmax function and aggregate the neighbours within the window with the acquired attention weights,

\[ v^\Delta_i = \text{softmax}(a^\Delta_i)v^\Delta_i, \]  

(9)
Afterward, sum up the weighted values at the same position from different windows,

\[ \hat{v}_i = \sum_{0 \leq r < K} \hat{v}_i^r \Delta_{i+r} \frac{1}{K}, \quad (10) \]

where \( \hat{v}_i \) is the acquired representation for \( i \)-th token with rich local context information, and \( \hat{v} \) is the representation matrix for the whole sequence.

Finally, conduct a residual operation [He et al., 2016] that sum the encoded \( \hat{v} \) with \( h_e \), and go through several linear layers for further representation learning, followed by performing another residual operation,

\[
\begin{align*}
    h_f &= \hat{v} + \hat{h}_e, \quad (11) \\
    \hat{h}_f &= W_f h_f + b_f, \quad (12) \\
    h'_f &= \hat{h}_f + h_f. \quad (13)
\end{align*}
\]

### 3.4 Output Layers

We stack linear layers to perform answer prediction with the obtained \( h'_f \),

\[
\begin{align*}
    y_s &= \text{softmax}(W_s h'_f + b_s), \quad (14) \\
    y_e &= \text{softmax}(W_e h'_f + b_e), \quad (15)
\end{align*}
\]

\( W_s, W_e \) and \( b_s, b_e \) are parameters and bias needed to be learned in the linear layers. \( y_s, y_e \) are the probability distributions for the start position and the end position.

The training objective is to minimize the sum of the negative log probabilities at the ground truth start position \( a_s \) and end position \( a_e \),

\[ \mathcal{L} = -\log(y_{a_s}^s + y_{a_e}^e). \quad (16) \]

### 4 Experiments

We evaluate the proposed system for extractive question answering on the Stanford Question Answering Dataset (SQuAD), i.e., SQuAD 2.0 [Rajpurkar et al., 2018] and SQuAD 1.1 [Rajpurkar et al., 2016], with no additional training data. We use the standard metrics for question answering, F1 measure and exact match (EM).

We conduct extensive experiments with other NLP tasks and other language to evaluate our proposed approach’s versatility and significance.

### 4.1 Setup

We leverage the encoder in BERT-base\(^2\), ALBERT-base\(^3\), RoBERTa-base\(^4\), and ProphetNet\(^5\) for text generation as the global encoder. 2-padding is used to maintain the sequence length.

In the context outlook block, the kernel size of context outlook layers is \( 3 \times 300 \), depending on the number of filters in the convolutional layers. While without a convolutional block, it is set to \( 3 \times H \) according to the global encoder.

During training, we use AdamW [Loshchilov and Hutter, 2019] optimizer. We set a learning rate to 2e-5 for the global encoder, and 1e-4 for other layers. We train the model with a mini-batch size of 48 for about 5 epochs. The main codes are uploaded as supplementary materials.

<table>
<thead>
<tr>
<th>Architecture</th>
<th>Outlook Layers</th>
<th>F1</th>
<th>EM</th>
</tr>
</thead>
<tbody>
<tr>
<td>BERT</td>
<td>-</td>
<td>75.41</td>
<td>71.78</td>
</tr>
<tr>
<td>G2L w/o Conv.</td>
<td>3</td>
<td>76.79</td>
<td>73.36</td>
</tr>
<tr>
<td>G2L w/ Conv.</td>
<td>2</td>
<td>77.11</td>
<td>74.02</td>
</tr>
<tr>
<td>L2G w/o Conv.</td>
<td>3</td>
<td>74.80</td>
<td>72.02</td>
</tr>
<tr>
<td>L2G w/ Conv.</td>
<td>2</td>
<td>50.06</td>
<td>49.98</td>
</tr>
<tr>
<td>G&amp;L w/o Conv.</td>
<td>3</td>
<td>75.58</td>
<td>72.78</td>
</tr>
<tr>
<td>G&amp;L w/ Conv.</td>
<td>2</td>
<td>75.98</td>
<td>72.86</td>
</tr>
</tbody>
</table>

Table 1: SQuAD 2.0 Dev results of different integration modes.

The input representations, for instance, \( H \) is 768 while using BERT-base as global encoder. 2-padding is used to maintain the sequence length.

In the context outlook block, the kernel size of context outlook layers is \( 3 \times 300 \), depending on the number of filters in the convolutional layers. While without a convolutional block, it is set to \( 3 \times H \) according to the global encoder.

During training, we use AdamW [Loshchilov and Hutter, 2019] optimizer. We set a learning rate to 2e-5 for the global encoder, and 1e-4 for other layers. We train the model with a mini-batch size of 48 for about 5 epochs. The main codes are uploaded as supplementary materials.

### 4.2 Extractive Question Answering

**Choice of architecture.** We first compare the three modes of global and local encoding integration: G2L, L2G, and G&L.

Table 1 reports the results of the respective architectures with BERT as a baseline. The table shows the most effective mode is G2L. G&L achieves a minor improvement over the baseline, yet not exceeding that of the first structure. A possible reason is it’s difficult to precisely capture the relationships between global and local representations. We find L2G architecture does not work well. The local encoding operation perturbs the following global encoding procedure, particularly while integrating the convolution operations with different kernel sizes. Specifically, directly combining the information within different n-grams confuses the model. In summary, it should better not replace the input to the pre-trained model. We nevertheless acknowledge that early convolution might benefit a scenario as discussed in [Xiao et al., 2021] for visual tasks. In the following experiments, we adopt the G2L mode.

**Overall results.** Table 2 shows the comparisons of COOL with and without the convolutional block combined with BERT, ALBERT, RoBERTa and the corresponding baselines for SQuAD 2.0 and SQuAD 1.1. It can be viewed that COOL with BERT achieves an improvement of 2.24 and 1.7 points for EM and F1, respectively, over the original BERT. COOL with ALBERT adds 0.88 EM points and 0.86 F1 points. COOL with RoBERTa outperforms the original in terms of both EM and F1 as well. SQuAD 1.1 does not contain unanswerable questions and is thus less challenging than SQuAD 2.0. Nevertheless, models inserted COOL improve across the board over the original models. These results confirm the benefit of an appropriate consideration of local information.

---

\(^2\)bert-base-cased: https://huggingface.co/bert-base-cased

\(^3\)albert-base-v2: https://huggingface.co/albert-base-v2

\(^4\)roberta-base-squad2: https://huggingface.co/roberta-base

\(^5\)https://github.com/microsoft/ProphetNet
Table 2: SQuAD Dev results for extractive question answering.

<table>
<thead>
<tr>
<th>Model</th>
<th>F1</th>
<th>EM</th>
</tr>
</thead>
<tbody>
<tr>
<td>BERT</td>
<td>75.41</td>
<td>71.78</td>
</tr>
<tr>
<td>COOL(BERT) w/o Conv.</td>
<td>76.79</td>
<td>73.36</td>
</tr>
<tr>
<td>COOL(BERT) w/ Conv.</td>
<td>77.11</td>
<td>74.02</td>
</tr>
<tr>
<td>ALBERT</td>
<td>78.63</td>
<td>75.33</td>
</tr>
<tr>
<td>COOL(ALBERT) w/o Conv.</td>
<td>79.23</td>
<td>75.95</td>
</tr>
<tr>
<td>COOL(ALBERT) w/ Conv.</td>
<td>79.49</td>
<td>76.21</td>
</tr>
<tr>
<td>RoBERTa</td>
<td>82.91</td>
<td>79.87</td>
</tr>
<tr>
<td>COOL(RoBERTa) w/o Conv.</td>
<td>83.39</td>
<td>80.12</td>
</tr>
<tr>
<td>ROBERTa</td>
<td>83.61</td>
<td>80.33</td>
</tr>
</tbody>
</table>

Table 3: Comparisons with deeper models.

<table>
<thead>
<tr>
<th>Model</th>
<th>F1</th>
<th>EM</th>
</tr>
</thead>
<tbody>
<tr>
<td>BERT</td>
<td>88.33</td>
<td>80.76</td>
</tr>
<tr>
<td>COOL(BERT) w/o Conv.</td>
<td>88.38</td>
<td>81.00</td>
</tr>
<tr>
<td>COOL(BERT) w/ Conv.</td>
<td>88.44</td>
<td>80.91</td>
</tr>
<tr>
<td>ALBERT</td>
<td>87.81</td>
<td>79.50</td>
</tr>
<tr>
<td>COOL(ALBERT) w/o Conv.</td>
<td>87.81</td>
<td>79.93</td>
</tr>
<tr>
<td>COOL(ALBERT) w/ Conv.</td>
<td>87.90</td>
<td>80.15</td>
</tr>
<tr>
<td>RoBERTa</td>
<td>90.40</td>
<td>83.00</td>
</tr>
<tr>
<td>COOL(RoBERTa) w/o Conv.</td>
<td>90.65</td>
<td>83.36</td>
</tr>
<tr>
<td>COOL(RoBERTa) w/ Conv.</td>
<td>90.66</td>
<td>83.42</td>
</tr>
</tbody>
</table>

Table 4: Comparisons with other techniques for local augmentation.

<table>
<thead>
<tr>
<th>Model</th>
<th>F1</th>
<th>EM</th>
</tr>
</thead>
<tbody>
<tr>
<td>BERT</td>
<td>75.41</td>
<td>71.78</td>
</tr>
<tr>
<td>BERT-Deeper(3 layers)</td>
<td>75.28</td>
<td>71.81</td>
</tr>
<tr>
<td>BERT-Deeper(5 layers)</td>
<td>74.94</td>
<td>71.28</td>
</tr>
<tr>
<td>COOL(BERT) w/o Conv.</td>
<td>76.79</td>
<td>73.36</td>
</tr>
<tr>
<td>COOL(BERT) w/ Conv.</td>
<td>77.11</td>
<td>74.02</td>
</tr>
</tbody>
</table>

Illustrative examples. We first present and discuss illustrative examples of the situations in which models with COOL predict the correct or better answers while the baseline models do not and reverse. The examples indicate the question (Q), the paragraph or the significant excerpt thereof (P), the answers by the baseline models, and the answers by COOL with the corresponding baseline models. The key portions of the paragraph pertinent to the discussion are highlighted in bold, and correct answers are indicated with underlining.

In Example 1, the baseline models fail to predict the correct answer. After inserting COOL, COOL(BERT) and COOL(RoBERTa) precisely recognize the relationship between ‘USC’ and ‘Trojans’ respectively and thus give the correct answer. Although COOL(ALBERT) still fails, it attempts to recognize the relationship between the state ‘UCLA’ and the team’s name ‘Bruins’. We further visualize the self-attention weights in Figure 4. We can observe the most important tokens to ‘USC’ are itself, ‘UCLA’, ‘both’, in that order. For ‘UCLA’, the important tokens are itself, ‘USC’, ‘the’, ‘Bruins’, ‘and’, and ‘both’. Likewise, the important words to ‘Bruins’ are
itself, ‘both’, ‘and’, ‘Trojan’, and to ‘Trojan’ are itself, its subtoken ‘#s’, and ‘Bruins’. It explains that the standard self-attention operation via similarity score calculation can not distinguish these tokens and understand their relationships. It thus predicts a long span as the answer. Whereas augmenting the local context information by COOL helps to address this disadvantage and benefits understanding the relations between close neighbours.

Example 1.

Q: What is the name of the team from USC?

P: The UCLA Bruins and the USC Trojans both field teams.

BERT: UCLA Bruins and the COOL(BERT): Trojans

ALBERT: UCLA Bruins and the USC Trojans

ROBERTa: USC Trojans

Moreover, we showcase interesting Example 2, the consideration of the local context allows COOL to capture logical relations where the baseline models fail. It is reasonable as most logical formulas are within local content.

Example 2.

Q: What is an expression that can be used to illustrate the suspected inequality of complexity classes?

P: Many known complexity classes are suspected to be unequal. For instance $P \subseteq NP \subseteq PP \subseteq PSPACE$, but it is possible that $P = PSPACE$.

BERT: (no answer)

ALBERT: $P = PSPACE$, \textit{but it is possible that $P = PSPACE$}

ROBERTa: $P = PSPACE$, \textit{but it is possible that $P = PSPACE$}

There are, unfortunately, situations in which too much local attention can be misleading and detrimental, as illustrated by Examples 3. We target to address this challenge of balancing the global and local encoding in future work.

Example 3.

Q: What was the Anglo-Norman language’s final form?

P: The Anglo-Norman language was eventually absorbed into the Anglo-Saxon language of their subjects (see Old English) and influenced it, helping (along with the Norse language of the earlier Anglo-Norse settlers and the Latin used by the church) in the development of Middle English. It in turn evolved into Modern English.

BERT: Modern English

ALBERT: (no answer)

ROBERTa: Modern English
such as tense [Tian and Lo, 2015] . NER and POS tagging e.g., noun, verb, adjective, and other grammatical categories tagging assigns to each word in the text its part of speech, instruction and person [Yamada et al., 2020]. For instance, names of locations and people from the text entities in a sentence into pre-defined categories. It extracts, Named-entity recognition (NER) locates and classifies named entities in a sentence into pre-defined categories. It extracts, Named-entity recognition and part-of-speech tagging. The Situations With Adversarial Generations (SWAG) dataset contains 113k sentence-pair examples that evaluate grounded commonsense inference [Zellers et al., 2018]. It targets to choose the most plausible continuation among four choices. The results of the comparisons of the baselines with models with COOL are reported in Table 7. According to the results, COOL module is able to improve this task.

Named-entity recognition and part-of-speech tagging. Named-entity recognition (NER) locates and classifies named entities in a sentence into pre-defined categories. It extracts, for instance, names of locations and people from the text and places them under certain categories, such as organization and person [Yamada et al., 2020]. Part-of-speech (POS) tagging assigns to each word in the text its part of speech, e.g., noun, verb, adjective, and other grammatical categories such as tense [Tian and Lo, 2015]. NER and POS tagging are basic NLP tasks. We evaluate the model performance on the CoNLL-2003 dataset [Tjong Kim Sang and De Meulder, 2003]. The comparative results are reported in Table 7. We can observe that COOL benefits the two tasks as well.

Sentiment analysis. Sentiment analysis (SA) determines whether a sentence is positive, negative, or neutral. The common datasets are the large movie review dataset (IMDB) [Maas et al., 2011] and the Stanford Sentiment Treebank (SST-2) from GLUE. The comparisons are also presented in Table 7. For this task, the results are mixed, suggesting the possible lesser importance of the fine-grained element brought by involving the local context.

4.4 Other Language

To verify the proposed context outlooker is generalisable for other languages. We try to apply it to Malay, an important low-resource language in Southeast Asia. We evaluate the approach on Malay SQuAD 2.0 which is translated from English SQuAD 2.0 by Husein6. Compared with the original BERT-Bahasa7 in Table 8, the proposed approach also brings improvement to the Malay language. It further proves the effectiveness and generalisability of the proposed module.

---

6https://github.com/huseinzol05/Malay-Dataset
7https://github.com/huseinzol05/malaya

5 Conclusion

We presented COOL, an outlook attention mechanism for natural language processing that can be added to a transformer-based network for any natural language processing task.

A comparative empirical performance evaluation of an implementation of COOL with different models confirms that COOL, in its Global-to-Local mode, yields a practical performance improvement over a baseline using the original model alone for question answering. The performance and examples suggest that COOL is indeed able to acquire a more expressive representation of local context information.

We also presented empirical results for other natural language processing tasks: question generation, multiple-choice question answering, natural language inference, sentiment analysis, named-entity recognition, and part-of-speech tagging not only illustrate its portability and versatility of COOL over different models and for different tasks but also confirms the positive contribution of COOL to the effectiveness of the resolution of these tasks and other languages.

### Table 7: SWAG Test accuracy results, CoNLL-2003 NER and POS F1 results, and SST and IMDB accuracy results.

<table>
<thead>
<tr>
<th>Model</th>
<th>SWAG</th>
<th>NER</th>
<th>POS</th>
<th>SST</th>
<th>IMDB</th>
</tr>
</thead>
<tbody>
<tr>
<td>BERT</td>
<td>79.7</td>
<td>90.8</td>
<td>93.4</td>
<td>92.5</td>
<td>93.4</td>
</tr>
<tr>
<td>COOL(BERT)</td>
<td>79.9</td>
<td>91.4</td>
<td>93.7</td>
<td>92.2</td>
<td>93.6</td>
</tr>
<tr>
<td>RoBERTa</td>
<td>83.3</td>
<td>91.9</td>
<td>92.8</td>
<td>94.8</td>
<td></td>
</tr>
<tr>
<td>COOL(RoBERTa)</td>
<td>83.4</td>
<td>95.3</td>
<td>93.5</td>
<td>95.2</td>
<td>91.2</td>
</tr>
</tbody>
</table>

### Table 8: Malay SQuAD 2.0 Dev results for Malay extractive question answering.

<table>
<thead>
<tr>
<th>Model</th>
<th>EM</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>BERT</td>
<td>57.17</td>
<td>61.49</td>
</tr>
<tr>
<td>+ Local Graph (RGCN)</td>
<td>74.83</td>
<td>71.52</td>
</tr>
<tr>
<td>+ Local Graph (GAT)</td>
<td>75.79</td>
<td>72.90</td>
</tr>
<tr>
<td>+ Local Graph (HGT)</td>
<td>75.96</td>
<td>72.67</td>
</tr>
</tbody>
</table>

### Table 9: Comparisons with GNNs on SQuAD 2.0.

<table>
<thead>
<tr>
<th>Model</th>
<th>F1</th>
<th>EM</th>
</tr>
</thead>
<tbody>
<tr>
<td>BERT</td>
<td>75.41</td>
<td>71.78</td>
</tr>
<tr>
<td>+ Local Graph (RGCN)</td>
<td>74.83</td>
<td>71.52</td>
</tr>
<tr>
<td>+ Local Graph (GAT)</td>
<td>75.79</td>
<td>72.90</td>
</tr>
<tr>
<td>+ Local Graph (HGT)</td>
<td>75.96</td>
<td>72.67</td>
</tr>
</tbody>
</table>

4.5 Comparison With Augmentation by a Local Graph

We consider whether we can perform local augmentation by building a local graph on top of the fully connected graph processed by the self-attention mechanism. Thus, we try to build a local graph with the outputs from a backbone where only close neighbours within the predefined range are connected and adopt graph neural networks (GNNs) to update the vertex embeddings. In Table 9, we report the preliminary results by utilising different GNNs, i.e., relational graph convolution network (RGCN) [Michael Sejr et al., 2018], graph attention network (GAT) [Veličković et al., 2017], and heterogeneous graph transformer (HGT) [Hu et al., 2020]. The model with RGCN underperforms the baseline as it treats all neighbours equally, which might weaken the discrimination of different neighbours. The models with GAT or HGT, considering weighted effects from different neighbours, outperform the baselines that prove the impact of local augmentation. Nevertheless, the results do not exceed the model with a context outlooker. We consider the reason is the GNNs handle the direct neighbours only. However, the context outlooker considers more constraints from indirect neighbours.
Acknowledgments

This research/project is supported by the National Research Foundation, Singapore under its Industry Alignment Fund – Pre-positioning (IAF-PP) Funding Initiative. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of National Research Foundation, Singapore.

References


[Maas et al., 2011] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In the Association for Computational Linguistics, 2011.


