CogLTX: Applying BERT to Long Texts

Ming Ding
Tsinghua University
dml8@mails.tsinghua.edu.cn

Chang Zhou
Alibaba Group
ericzhou.zc@alibaba-inc.com

Hongxia Yang
Alibaba Group
yang.yhx@alibaba-inc.com

Jie Tang
Tsinghua University
jietang@tsinghua.edu.cn

Abstract

BERT is incapable of processing long texts due to its quadratically increasing memory and time consumption. The most natural ways to address this problem, such as slicing the text by a sliding window or simplifying transformers, suffer from insufficient long-range attentions or need customized CUDA kernels. The maximum length limit in BERT reminds us the limited capacity (5~9 chunks) of the working memory of humans — then how do human beings Cognize Long Texts? Founded on the cognitive theory stemming from Baddeley [2], the proposed CogLTX framework identifies key sentences by training a judge model, concatenates them for reasoning, and enables multi-step reasoning via rehearsal and decay. Since relevance annotations are usually unavailable, we propose to use interventions to create supervision. As a general algorithm, CogLTX outperforms or gets comparable results to SOTA models on various downstream tasks with memory overheads independent of the length of text.

1 Introduction

Pretrained language models, pioneered by BERT [12], have emerged as silver bullets for many NLP tasks, such as question answering [38] and text classification [22]. Researchers and engineers breezily build state-of-the-art applications following the standard finetuning paradigm, while might end up in disappointment to find some texts longer than the length limit of BERT (usually 512 tokens). This situation may be rare for normalized benchmarks, for example SQuAD [38] and GLUE [47], but very common for more complex tasks [53] or real-world textual data.

A straightforward solution for long texts is sliding window [50], processing continuous 512-token spans by BERT. This method sacrifices the possibility that the distant tokens “pay attention” to each other, which becomes the bottleneck for BERT to show its efficacy in complex tasks (for example Figure 1). Since the problem roots in the high $O(L^2)$ time and space complexity in transformers [46] ($L$ is the length of the text).

Figure 1: An example from HotpotQA (distractor setting, concatenated). The key sentences to answer the question are the first and last ones, more than 512 tokens away from each other. They never appear in the same BERT input window in the sliding window method, hence we fail to answer the question.

1Codes are available at https://github.com/Sleepychord/CogLTX

text), another line of research attempts to simplify the structure of transformers [20] [37] [8] [42], but currently few of them have been successfully applied to BERT [35] [4].

The maximum length limit in BERT naturally reminds us the limited capacity of Working Memory [2], a human cognitive system storing information for logical reasoning and decision-making. Experiments [27] [9] [51] already showed that the working memory could only hold 5~9 items/words during reading, so how do humans actually understand long texts?

“The central executive – the core of the (working memory) system that is responsible for coordinating (multi-modal) information”, and “functions like a limited-capacity attentional system capable of selecting and operating control processes and strategies”, as Baddeley [2] pointed out in his 1992 classic. Later research detailed that the contents in the working memory decay over time [3], unless are kept via rehearsal [3], i.e. paying attention to and refreshing the information in the mind. Then the overlooked information is constantly updated with relevant items from long-term memory by retrieval competition [52], collecting sufficient information for reasoning in the working memory.

The analogy between BERT and working memory inspires us with the CogLTX framework to Cognize Long TeXts like human. The basic philosophy behind CogLTX is rather concise — reasoning over the concatenation of key sentences (Figure 2) — while compact designs are demanded to bridge the gap between the reasoning processes of machine and human.

The critical step in CogLTX is MemRecall, the process to identify relevant text blocks by treating the blocks as episodic memories. MemRecall imitates the working memory on retrieval competition, rehearsal and decay, facilitating multi-step reasoning. Another BERT, termed judge, is introduced to score the relevance of blocks and trained jointly with the original BERT reasoner. Moreover, CogLTX can transform task-oriented labels to relevance annotations by interventions to train judge.

Our experiments demonstrate that CogLTX outperforms or achieves comparable performance with the state-of-the-art results on four tasks, including NewsQA [44], HotpotQA [53], 20NewsGroups [22] and Alibaba, with constant memory consumption regardless of the length of text.

2 Background

Challenge of long texts. The direct and superficial obstacle for long texts is that the pretrained max position embedding is usually 512 in BERT [12]. However, even if the embeddings for larger positions are provided, the memory consumption is unaffordable because all the activations are stored for back-propagation during training. For instance, a 1,500-token text needs about 14.6GB memory for back-propagation during training. For instance, a 1,500-token text needs about 14.6GB memory (512 · L/512) = O(512L) space, which in practice is still too large to train a BERT-large on a 2,500-token text on RTX 2080ti with batch size of 1. Besides, these late-aggregation methods mainly optimizes classification, while other tasks, e.g., span extraction, have L BERT outputs, need O(L^2) space for self-attention aggregation.
3 Method

3.1 The CogLTX methodology

This basic assumption of CogLTX is that “for most NLP tasks, a few key sentences in the text store sufficient and necessary information to fulfill the task”. More specifically, we assume there exists a short text \( z \) composed by some sentences from the long text \( x \), satisfying

\[
\text{reasoner}(x^+) \approx \text{reasoner}(z^+), \tag{1}
\]

where \( x^+ \) and \( z^+ \) are inputs for the \( \text{reasoner} \) BERT w.r.t. the texts \( x \) and \( z \) as illustrated in Figure 2.

We split each long text \( x \) into blocks \([x_0 \ldots x_{r-1}]\) by dynamic programming (see the Appendix), which restricts the block length to a maximum of \( B \). \( B = 63 \) if the BERT length limit \( L = 512 \). The key short text \( z \) should be composed by some blocks in \( x \), i.e. \( z = [x_{z_0} \ldots x_{z_{n-1}}] \), satisfying \( \text{len}(z^+) \leq L \) and \( z_0 < \ldots < z_{n-1} \). We denote \( x_{z_i} \) by \( z_i \). All blocks in \( z \) are automatically sorted to maintain the original relative ordering in \( x \).

The key-blocks assumption is strongly related to latent variable models, which are usually solved by EM [11] or variational bayes [19]. However, these methods estimate the distribution of \( z \) and require multiple-sampling, thus not efficient enough for BERTs. We take the essence of them into the design of CogLTX, and discuss the connections in §3.3.

Two ingredients are essential in CogLTX, MemRecall and the joint training of two BERTs. As demonstrated in Figure 2, MemRecall is the algorithm utilizing the judge model to retrieve key blocks, which are fed into the reasoner to accomplish the task during inference.
3.2 MemRecall

As the brain recalls past episodes relevant to current information in working memory, the MemRecall aims to extract key blocks \( z \) from the long text \( x \) (see Figure 5).

**Input.** Although the aim is to extract key blocks, specific settings differ in the three types of tasks. In Figure 2(a) (c), the question \( Q \) or sub-sequence \( x[i] \) serves as query to retrieve relevant blocks. However, queries are absent in (b), and the relevance is only implicitly defined by the training data. For instance, sentences containing “Donald Trump” or “basketball” are more relevant for news topic classification than time reporting sentences. So how to seamlessly unify the cases?

MemRecall answers by accepting an initial \( z^+ \) as an additional input besides \( x \). \( z^+ \) is the short “key text” maintained during MemRecall to simulate working memory. The query in tasks (a)(c) becomes the initial information in \( z^+ \) to provoke recalling. Then a judge model learns to predict task-specific relevance with the help of \( z^+ \).

**Model.** The only model used by MemRecall is the judge mentioned above, a BERT to score the relevance for each token. Suppose \( z^+ = [[CLS] \ Q \ [SEP] z_0 [SEP] ... z_{n-1}] \),

\[
\text{judge}(z^+) = \text{sigmoid}(\text{MLP}(\text{BERT}(z^+))) \in (0, 1)^{\text{len}(z^+)}. \tag{2}
\]

The score of a block \( z_i \), denoted as \( \text{judge}(z^+)[z_i] \), is the average of the scores of tokens in the block.

**Procedure.** MemRecall begins with a retrieval competition. Each block \( x_i \) is assigned a coarse relevance score \( \text{judge}([z^+ [SEP] x_i]) [x_i] \). The “winner” blocks with the highest scores are inserted into \( z \) as much as \( \text{len}(z^+) \leq L \). The superiority over vector space models [40] lies in that \( x \) fully interacts with current \( z^+ \) via transformers, avoiding information loss during embedding.

The following rehearsal-decay period assigns each \( z_i \) a fine relevance score \( \text{judge}(z^+)[z_i] \). Only the highest scored blocks are then kept in \( z^+ \), just like the rehearsal-decay phenomenon in working memory. The motivation of fine scores is that the relative sizes of coarse scores are not accurate enough without interaction and comparison between blocks, similar to the motivation of reranking [7].

MemRecall in nature enables multi-step reasoning by repeating the procedure with new \( z^+ \). The importance of iterative retrieval is highlighted by CogQA [13], as the answer sentence fails to be directly retrieved by the question in multi-hop reading comprehension. It is worth noting that blocks reserved from last step can also decay, if they are proved not relevant enough (with low scores) by more information from new blocks in \( z^+ \), which is neglected by previous multi-step reasoning methods [13][4][10].

3.3 Training

The diversity of downstreaming tasks pose challenges for training (finetuning) the BERTs in CogLTX. The solutions under different settings are summarized in Algorithm 1.

**Supervised training for judge.** The span extraction tasks (Figure 2(a)) in nature suggest the answer block as relevant. Even multi-hop datasets, e.g. HotpotQA [53], usually annotate supporting sentences. In these cases, the judge is naturally trained in a supervised way:

\[
\text{loss}_{\text{judge}}(z) = \text{CrossEntropy}(\text{judge}(z^+), \text{relv}_\text{label}(z^+)), \tag{3}
\]

\[
\text{relv}_\text{label}(z^+) = \begin{cases} 1, 1, ..., 1, & \text{for query} \\ 0, 0, ..., 0, & \text{z_0 is irrelevant} \\ 1, 1, ..., 1, & \text{z_1 is relevant} \end{cases} \in [0, 1]^{\text{len}(z^+)}, \tag{4}
\]

where the training sample \( z \) is either a sequence of continuous blocks \( z_{\text{rand}} \) sampled from \( x \) (corresponding to the data distribution of retrieval competition), or a mixture of all relevant and randomly selected irrelevant blocks \( z_{\text{relv}} \) (approximating the data distribution of rehearsal).

**Supervised training for reasoner.** The challenge for reasoner is to keep the consistency of data distributions during training and inference, which is a cardinal principle of supervised learning. Ideally, the inputs of reasoner should also be generated by MemRecall during training, but not all relevant blocks are guaranteed to be retrieved. For instance in question answering, if the answer block
We exhibit an example of unsupervised training on the 20News text classification dataset in Figure 4. 

Algorithm 1: The Training Algorithm of CogLTX

Input: Training set $D = \{(x_n, y_n), \ldots, (x_n, y_n)\}$. Hyperparameters: num_epoch, mode, $t_{up}$, $t_{down}$.

1. If mode is unsupervised then
   2. Initialize the relevance labels in $D$ by Bm25 or Glove if possible. // see Appendix for details.
   3. For epoch from 1 to num_epoch do
      4. For $x$, $y$ in $D$ do
         5. Extract a short ($len \leq L$) span from $x$ at random as $z_{rand}$.
         6. $loss_{rand} = CrossEntropy(judge(z_{rand}^{+}), relv_label(z_{rand}^{+}))$.
         7. Aggregate all relevant blocks and randomly chosen irrelevant blocks as $z_{relv}$ ($len \leq L$).
         8. $loss_{relv} = CrossEntropy(judge(z_{relv}^{+}), relv_label(z_{relv}^{+}))$.
         9. Update judge by descending $\nabla_{\phi}(loss_{rand} + loss_{relv})$. // $\phi$ is the parameters of judge.
      10. For $x$, $y$ in $D$ do
          11. Aggregate all relevant blocks in $x$ as $z$.
          12. For irrelevant block $x_i$ in $x$ do
              13. $\text{score}[x_i] = \text{judge}(z, x_i^{+})[x_i]$ // can be replaced by cached scores during training judge.
          14. Fill $z$ up to length $L$ with highest scoring blocks. // corresponding to the $z$ from MemRecall.
          15. $loss = CrossEntropy(reasoner(z^{+}), y)$.
          16. Update reasoner by descending $\nabla_{\theta}loss$. // $\theta$ is the parameters of reasoner.
      17. If mode is unsupervised and epoch > 1 then
         18. For block $z_i$ in $z$ do
             19. $loss_{-z_i} = CrossEntropy(reasoner(z_{-z_i}^{+}), y)$ // gradient-free, much faster than Line 15.
             20. If $loss_{-z_i} - loss > t_{up}$ then Label $z_i$ as relevant;
             21. If $loss_{-z_i} - loss < t_{down}$ then Label $z_i$ as irrelevant;

In practice, we split $t$ into $t_{up}$ and $t_{down}$, leaving a buffer zone to prevent frequent changes of labels. We exhibit an example of unsupervised training on the 20News text classification dataset in Figure 4.

Unsupervised training for judge. Unfortunately, many tasks (Figure 2 (b)(c)) do not provide relevance labels. Since CogLTX assumes all relevant blocks necessary, we infer the relevance labels by interventions: test whether a block is dispensable by removing it from $z$.

Suppose that $z$ is the “oracle relevant blocks”, according to our assumption,

$$loss_{reasoner}(z_{-z_i}) - loss_{reasoner}(z) > t, \forall z_i \in z, \quad (necessity)$$  \hspace{1cm} (5)
$$loss_{reasoner}(z_{-z_i}) - loss_{reasoner}(z) \approx 0, \forall x_i \notin z, \quad (sufficiency)$$  \hspace{1cm} (6)

where $z_{-z_i}$ is the result of removing $z_i$ from $z$, and $t$ is a threshold. After every iteration in training reasoner, we ablate each block in $z$, adjust its relevance label according to the increase of loss. Insignificant increase reveals the block as irrelevant, which will probably not “win the retrieval competition” again to train the reasoner in the next epoch, because it will be labeled as irrelevant to train the judge in the next epoch. Then real relevant blocks might enter $z$ next epoch and be detected. In practice, we split $t$ into $t_{up}$ and $t_{down}$, leaving a buffer zone to prevent frequent changes of labels.

Connections to latent variable models. Unsupervised CogLTX can be viewed as a generalization of (conditional) latent variable model $p(y|x; \theta) \propto p(z|x)p(y|z; \theta)$. EM [11] infers the distribution of $z$ as posterior $p(z|x; \theta)$ in E-step, while variational bayes methods [19, 39] use an estimation-friendly $q(z|x, y)$. However, in CogLTX $z$ has a discrete distribution with up to $C_n^m$ possible values, where $n$, $m$ are the number of blocks and the capacity of $z$ respectively. In some cases, sampling for hundreds of times to train BERTs might be required [19], whose expensive time consumption force us turn to point estimation for $z$ e.g. our intervention-based method.

The intervention solution, maintains an $z$ estimation for each $x$, and is essentially a local search specific to CogLTX. $z$ is optimized by comparing nearby values(results after replacing irrelevant blocks) rather than Bayesian rules. The judge fits an inductive discriminative model to help infer $z$. 

\footnote{analogous to K-means, which can be seen as EM for Gaussian mixture model with infinitesimal variances. Then the posterior of $z$, mixture belonging, degenerates into the nearest cluster (the deterministic MLE value).}
4 Experiments

We conducted experiments on four long-text datasets with different tasks. The token-wise (Figure 2 (c)) tasks are not included because they mostly barely need information from adjacent sentences, and are finally transformed into multiple sequence-level samples. The boxplot in Figure 5 illustrates the statistics of the text length in the datasets.

In all experiments, the judge and reasoner are fine-tuned by Adam [18] with learning rate $4 \times 10^{-5}$ and $10^{-4}$ respectively. The learning rates warmup over the first 10% steps, and then linearly decay to 1/10 of the max learning rates. The common hyperparameters are batch size = 32, strides = $[3, 5]$, $t_{up} = 0$ and $t_{down} = -0.05$.

In this section, we separately introduce each task with related results, analysis and ablation studies.

4.1 Reading comprehension

Dataset and settings. Given a question and a paragraph, the task is to predict the answer span in the paragraph. We evaluate the performance of CogLTX on NewsQA [44], which contains 119,633 human-generated questions posed on 12,744 long news articles.

Since previous SOTA [43] is not BERT based (due to long texts) in NewsQA, to keep the similar scale of parameters for fair comparison, we finetune the base version of RoBERTa [26] for 4 epochs in CogLTX.

Results. Table 1 show that CogLTX-base outperforms well-established QA models, for example BiDAF [41] (+17.8% $F_1$), previous SOTA DECAPROP [43], which incorporates elaborate self-attention and RNN mechanisms (+4.8% $F_1$), and even RoBERTa-large with sliding window (+4.8% $F_1$). We hypothesize that the first sentence (the lead) and the last sentence (the conclusion) are usually the most informative parts in news articles. CogLTX can aggregate them for reasoning while sliding window cannot.

<table>
<thead>
<tr>
<th>Model</th>
<th>EM</th>
<th>$F_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Match-LSTM [48]</td>
<td>34.9</td>
<td>50.0</td>
</tr>
<tr>
<td>BiDAF [41]</td>
<td>37.1</td>
<td>52.3</td>
</tr>
<tr>
<td>FastQAExt [51]</td>
<td>42.8</td>
<td>56.1</td>
</tr>
<tr>
<td>AMANDA [21]</td>
<td>48.4</td>
<td>63.7</td>
</tr>
<tr>
<td>MINIMAL [28]</td>
<td>50.1</td>
<td>63.2</td>
</tr>
<tr>
<td>DECAPROP [43]</td>
<td>53.1</td>
<td>66.3</td>
</tr>
<tr>
<td>RoBERTa-large [26]</td>
<td>49.6</td>
<td>66.3</td>
</tr>
</tbody>
</table>

CogLTX 55.2 70.1

Figure 5: The boxplot of the text length distribution in the datasets.

Table 1: NewsQA results (%).

---

We use the original version instead of the simplified version in MRQA [15], which removed long texts.
4.2 Multi-hop question answering

**Dataset and settings.** In complex scenarios, the answer is based on multiple paragraphs. Previous methods usually leverage the graph structure between key entities across the paragraphs [13, 36]. However, if we can handle long texts with CogLTX, the problem can be elegantly solved by concatenating all the paragraphs as the input of BERTs.

HotpotQA [53] is a multi-hop QA dataset of 112,779 questions, whose distractor setting provides 2 necessary paragraphs and 8 distractor paragraphs for each question. Both answers and supporting facts are required for evaluation. We treat each sentence as a block in CogLTX, and directly output the 2 blocks with the highest fine scores as supporting facts.

Table 2: Results on HotpotQA distractor (dev). (+hyperlink) means usage of extra hyperlink data in Wikipedia. Models beginning with "−" are ablation studies without the corresponding design.

<table>
<thead>
<tr>
<th>Model</th>
<th>Ans EM</th>
<th>Ans $F_1$</th>
<th>Sup EM</th>
<th>Sup $F_1$</th>
<th>Joint EM</th>
<th>Joint $F_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline [53]</td>
<td>45.60</td>
<td>59.02</td>
<td>20.32</td>
<td>64.49</td>
<td>10.83</td>
<td>40.16</td>
</tr>
<tr>
<td>DecompRC [29]</td>
<td>55.20</td>
<td>69.63</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>QFE [30]</td>
<td>53.86</td>
<td>68.06</td>
<td>57.75</td>
<td>84.49</td>
<td>34.63</td>
<td>59.61</td>
</tr>
<tr>
<td>DFGN [56]</td>
<td>56.31</td>
<td>69.69</td>
<td>51.50</td>
<td>81.62</td>
<td>33.62</td>
<td>59.82</td>
</tr>
<tr>
<td>SAE [45]</td>
<td>60.36</td>
<td>73.58</td>
<td>56.93</td>
<td>84.63</td>
<td>38.81</td>
<td>64.96</td>
</tr>
<tr>
<td>SAE-large</td>
<td>66.92</td>
<td>79.62</td>
<td>61.53</td>
<td>86.86</td>
<td>45.36</td>
<td>71.45</td>
</tr>
<tr>
<td>HGN [14] (+hyperlink)</td>
<td>66.07</td>
<td>79.36</td>
<td>60.33</td>
<td>87.33</td>
<td>43.57</td>
<td>71.03</td>
</tr>
<tr>
<td>HGN-large (+hyperlink)</td>
<td>69.22</td>
<td>82.19</td>
<td>62.76</td>
<td>88.47</td>
<td>47.11</td>
<td>74.21</td>
</tr>
</tbody>
</table>

**BERT (sliding window) variants**

<table>
<thead>
<tr>
<th>Model</th>
<th>Ans EM</th>
<th>Ans $F_1$</th>
<th>Sup EM</th>
<th>Sup $F_1$</th>
<th>Joint EM</th>
<th>Joint $F_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>BERT Plus</td>
<td>55.84</td>
<td>69.76</td>
<td>42.88</td>
<td>80.74</td>
<td>27.13</td>
<td>58.23</td>
</tr>
<tr>
<td>LQR-net + BERT</td>
<td>57.20</td>
<td>70.66</td>
<td>50.20</td>
<td>82.42</td>
<td>31.18</td>
<td>59.99</td>
</tr>
<tr>
<td>GRN + BERT</td>
<td>55.12</td>
<td>68.98</td>
<td>52.55</td>
<td>84.06</td>
<td>32.88</td>
<td>60.31</td>
</tr>
<tr>
<td>EPS + BERT</td>
<td>60.13</td>
<td>73.31</td>
<td>52.55</td>
<td>83.20</td>
<td>35.40</td>
<td>63.41</td>
</tr>
<tr>
<td>LQR-net 2 + BERT</td>
<td>60.20</td>
<td>73.78</td>
<td>56.21</td>
<td>84.09</td>
<td>36.56</td>
<td>63.68</td>
</tr>
<tr>
<td>P-BERT</td>
<td>61.18</td>
<td>74.16</td>
<td>51.38</td>
<td>82.76</td>
<td>35.42</td>
<td>63.79</td>
</tr>
<tr>
<td>EPS + BERT(large)</td>
<td>63.29</td>
<td>76.36</td>
<td>58.25</td>
<td>85.60</td>
<td>41.39</td>
<td>67.92</td>
</tr>
<tr>
<td>CogLTX</td>
<td>65.09</td>
<td>78.72</td>
<td>56.15</td>
<td>85.78</td>
<td>39.12</td>
<td>69.21</td>
</tr>
</tbody>
</table>

|                                   |        |           |        |           |          |             |
|                                   | − multi-step reasoning | 62.00  | 75.39  | 51.74   | 83.10   | 35.85  | 65.35       |
|                                   | − rehearsal & decay   | 61.44  | 74.99  | 7.74    | 47.37   | 5.36   | 37.74       |
|                                   | − train-test matching | 63.20  | 77.21  | 52.57   | 84.21   | 36.11  | 66.90       |

**Results.** Table 2 shows that CogLTX outperforms most of previous methods and all 7 BERT variants solutions on the leaderboard. These solutions basically follow the framework of aggregating the results from sliding windows by extra neural networks, leading to bounded performances attributed to insufficient interaction across paragraphs.

The SOTA model HGN [14] leverages extra hyperlink data in Wikipedia, based on which the dataset is constructed. The thought of SAE [45] is similar to CogLTX but less general. It scores paragraphs by an attention layer over BERTs, selects the highest scoring 2 paragraphs and feeds them into BERT together. The supporting facts are determined by another elaborate graph attention model. With the well-directed designs, SAE fits HotpotQA better than CogLTX (2.2% Joint $F_1$), but does not solve the memory problem for longer paragraphs. CogLTX directly solves the multi-hop QA problem as ordinary QA, gets SOTA-comparable results and explains the supporting facts without extra efforts.

**Ablation studies.** We also summarize the ablation studies in Table 2, indicating that (1) multi-step reasoning does work (+3.9% Joint $F_1$) but not essential, probably because many questions themselves in HotpotQA are relevant enough with the second-hop sentences to retrieve them. (2) The metrics on supporting facts drop dramatically (-35.7% Sup $F_1$) without rehearsal for fine scores, because the relevance scores of top sentences are not comparable without attending to each other.

https://hotpotqa.github.io
As mentioned in §3.3, the discrepancy of data distribution during training and test impairs the performance (-2.3% Joint F1) if reasoner is trained by randomly selected blocks.

4.3 Text classification

**Dataset and settings.** As one of the most general tasks in NLP, text classification is essential to analyze the topic, sentiment, intent, etc. We conduct experiments on the classic 20NewsGroups [22], which contains 18,846 documents from 20 classes. We finetune RoBERTa for 6 epochs in CogLTX.

**Results.** Table 3 demonstrates that CogLTX, whose relevance labels are initialized by Glove [34], outperforms the other baselines, including previous attempts to aggregate the [CLS] pooling results from the sliding window [33]. Moreover, MLP or LSTM based aggregation cannot be trained end-to-end either on long texts.

**Ablation studies.** (1) Since the text lengths in 20 NewsGroups vary greatly (see Figure 5), we further test the performance only on the texts longer than 512 tokens (15%), which is even above the global result. (2) The initialization based on Glove provides good relevance labels, but the lack of adjustments by interventions still leads to 2.2% decrease in accuracy. (3) The Bm25 initialization is based on common words, which only initializes 14.2% training samples due to the short label names, e.g., sports.baseball. The relevant sentences are inferred by interventions and the gradually trained reasoner, achieving an accuracy of 86.1%.

4.4 Multi-label classification

**Dataset and settings.** In many practical problems, each text can belong to multiple classes at the same time. The multi-label classification is usually transformed into binary classification by training an individual classifier for each label. Owing to the large capacity of BERT, we share the model for all the labels by prepending the label name at the beginning of the documents as input, i.e., [CLS] label [SEP] doc, for binary classification. Alibaba is a dataset of 30,000 articles extracted from an industry scenario in a large e-commerce platform. Each article advertises for several items from 67 categories. The detection of mentioned categories are perfectly modeled as multi-label classification. To accelerate the experiment, we respectively sampled 80,000 and 20,000 label-article pairs for training and testing. For this task, we finetune RoBERTa for 10 epochs in CogLTX.

**Results.** Table 4 shows that CogLTX outperforms common strong baselines. The word embeddings used by TextCNN [17] and Bi-LSTM are from RoBERTa for fair comparison. Even CogLTX-tiny (7.5M parameters) outperforms TextCNN. However, the max-pooling results of RoBERTa-large sliding window are worse than CogLTX (7.3% Macro-F1). We hypothesize this is due to the tendency to assign higher probabilities to very long texts in max-pooling, highlighting the efficacy of CogLTX.

4.5 Memory and time consumption

**Memory.** The memory consumption of CogLTX is constant during training, superior to the O(L^2) complexity of vanilla BERT. We also compare longformer [4], whose space complexity is roughly O(L) if the number of global attention tokens is small relative to L and independent of L. The detailed comparison is summarized in Figure 6 (Left).
**Time.** To accelerate the training of *reasoner*, we can cache the scores of blocks during training *judge* and then each epoch only needs $2 \times$ time of single-BERT training. As the numbers of epochs until convergence are similar for CogLTX and sliding window in training, the main concern of CogLTX is the speed of inference. Figure 6 (Right) shows the time to process a 100,000-sample synthetic dataset with different text lengths. CogLTX, with $O(n)$ time complexity, is faster than vanilla BERT after $L > 2,048$ and approaches the speed of sliding window as the text length $L$ grows.

![Graph showing memory and time consumption with varying text length.](image)

**Figure 6:** Memory and Time consumption with varying text length. The data about memory are measured with batch size $= 1$ on a Tesla V100. The batch size is 8 in the measurement of inference time consumption, and CogLTX does 1-step reasoning.

## 5 Conclusion and discussion

We present CogLTX, a cognition inspired framework to apply BERT to long texts. CogLTX only needs fixed memory during training and enables attentions between faraway sentences. Similar ideas were investigated on document-level in DrQA [6] and ORQA [23], and there are also previous works to extract important sentences in unsupervised ways, e.g. based on the metadata about structure [24]. Experiments on 4 different large datasets show its competitive performance. CogLTX is expected to become a general and strong baseline in many complex NLP tasks.

CogLTX defines a pipeline for long text understanding under the “key sentences” assumption. Extremely hard sequence-level tasks might violate it, thus efficient variational bayes methods (estimating the distribution of $z$) with affordable computation still worth investigations. Besides, CogLTX has a drawback to miss antecedents right before the blocks, which is alleviated by prepending the entity name to each sentence in our HotpotQA experiments, and could be solved by position-aware retrieval competition or coreference resolution in the future.

**Acknowledgements**

The work is supported by NSFC for Distinguished Young Scholar (61825602), NSFC (61836013), and a research fund supported by Alibaba. The authors would like to thank Danqi Chen and Zhilin Yang for their insightful discussion, and responsible reviewers of NeurIPS 2020 for their valuable suggestions.
Broader Impact

Positive impact. The proposed method for understanding longer texts is inspired by the theory of working memory in the human brain. After the success of pretraining language models that learn from extremely large corpus, it still remains mysterious how human being can memorize, understand, and conduct efficient yet effective reasoning process within a small memory budget, given a very few examples. Exploring such methods in fact may help design more elegant mechanism, or architecture that connects sub-models to solve complex tasks that require rich context and information. From a societal perspective, the proposed method can be also applied to many applications, e.g., legal document analysis, public opinion monitoring and searching.

Negative impact. With the help of such methods, social platforms may get better understanding about their users by analysing their daily posts. Longer texts understanding specifically provide more accurate and coherent interpretation of who they are, which is a privacy threat.

References


6 Appendix

6.1 MemRecall

As described in § 3, the MemRecall is the process to extract the key blocks. We also need “strides” as input to indicate how many new blocks will be kept in each step. Details are showed as follows:

Algorithm 2: MemRecall

Input: Long text \( x = [x_0, \ldots, x_{n-1}] \), strides \( = [stride_0, \ldots, stride_{m-1}] \). // strides for m-step reasoning.

1. \( z = [] \) // Initialize key blocks sequence as empty.
2. for \( i \) from 0 to \( m - 1 \) do
3.     for block \( x_i \) in \( x \) do
4.         score \( [x_i] = \text{judge}(z \oplus x_i) \) // relevance scores between \( x_i \) and query based on current \( z \).
5.     Fill \( z \) up to length \( L \) with highest scoring blocks. // Recall without interaction between candidate blocks.
6.     score \( [z_0, z_1, \ldots] = \text{judge}(z) \) // relevance scores after cross-attention and comparison.
7.     Retain \( \sum_{j=1}^{m} \text{stride}_j \) highest scoring blocks in \( z \). // rehearsal & decay, keep \( \text{stride}_i \) more blocks.
8. Return \( z \).

6.2 Block Split

We predefine a cost for each punctuation and a basic cost for “hard truncation”. The dynamic programming algorithm are showed as follows:

Algorithm 3: Block Split

Input: Long text \( x \), Punctuation costs \( \text{cost} \), basic cost \( c \), max block size \( B \)

1. Initialize \( f[0] \ldots f[B-1] \) as 0.
2. Initialize \( \text{from}[0] \ldots \text{from}[B-1] \) as -1.
3. for \( i \) from \( B \) to \( \text{len}(x) - 1 \) do
4.     \( f[i] = +\infty \).
5. for \( j \) from \( i - B \) to \( i - 1 \) do
6.     if word is punctuation then \( v = \text{cost[\text{word}]} + f[j] \);
7.     else \( v = c + f[j] \);
8.     if \( v < f[i] \) then \( f[i] = v \), \( \text{from}[i] = j \).
9. \( t = \text{len}(x) - 1 \), blocks = [].
10. while \( t \geq 0 \) do
11.     prepend \( x[\text{from}[t] + 1 \ldots t] \) to blocks.
12.     \( t = \text{from}[t] \).
13. Return blocks.

6.3 Initialization

In unsupervised training of CogLTX, an elaborate rather than random initialization for relevance labels could accelerate the convergence by large. Both query and textual label (e.g. label names for classification) can be used for initialization.

BM25 is a famous TF-IDF-like information retrieval method. Each block is scored based on the common words with query or textual label. However, the semantic relevance are neglected. For example, BM25 fails to find the relevance between label name “sports” with “baseball player”.

Glove is a group of pretrained word representation. Suppose the query or textual label is \( q \), and we score the relevance of a block \( b \) by averaging the \( \text{len}(q) \times \text{len}(b) \) inner products of their words. Top two blocks are initialized as relevant.