XLNet: Generalized Autoregressive Pretraining for Language Understanding

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell,
Ruslan Salakhutdinov, Quoc V. Le

Presented by

Dawei Zhou & Lecheng Zheng

Background

- Unsupervised representation learning has been successful
  - Pretrain on large-scale unlabeled text corpora → finetuning
  - CoVe (2017) → ELMo(2018) → GPT(2018) → BERT (2018)

- Pretraining methods: Autoregressive v.s Autoencoding

$$\max_{\theta} \quad \log p_\theta(\mathbf{x}) = \sum_{t=1}^{T} \log p_\theta(x_t \mid \mathbf{x}_{<t})$$

Autoregressive

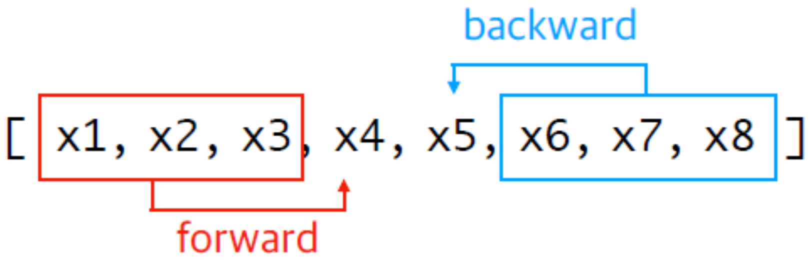
$$\max_{\theta} \quad \log p_\theta(\bar{\mathbf{x}} \mid \hat{\mathbf{x}}) \approx \sum_{t=1}^{T} m_t \log p_\theta(x_t \mid \hat{\mathbf{x}})$$

Autoencoding

Autoregressive v.s Autoencoding

- AR model (GPT)

$$\max_{\theta} \log p_{\theta}(x) = \sum_{t=1}^{T} \log p_{\theta}(x_t \mid x_{<t})$$

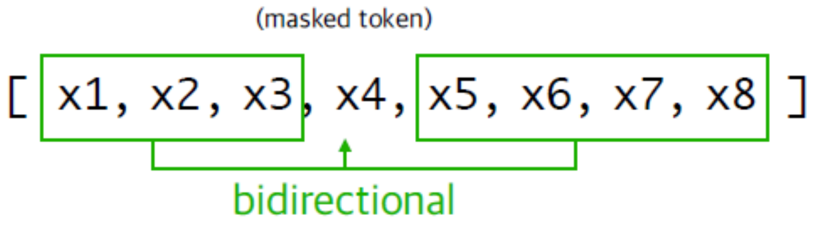


Good at text generation

Lack of bidirectional context information

- AE model (BERT)

$$\max_{\theta} \log p_{\theta}(\bar{x} \mid \hat{x}) \approx \sum_{t=1}^{T} m_t \log p_{\theta}(x_t \mid \hat{x})$$



Good at language understanding

Artificial symbols like [MASK]

AR Language Model

- AR performs pretraining by maximizing the likelihood under the forward autoregressive factorization

$$\max_{\theta} \quad \log p_\theta(\mathbf{x}) = \sum_{t=1}^{T} \log p_\theta(x_t \mid \mathbf{x}_{<t}) = \sum_{t=1}^{T} \log \frac{\exp\left(h_\theta(\mathbf{x}_{1:t-1})^\top e(x_t)\right)}{\sum_{x'} \exp\left(h_\theta(\mathbf{x}_{1:t-1})^\top e(x')\right)}$$

  - $h_\theta(x_{1:t-1})$: context representation produced by neural models.
  - $e(x)$: embedding of x.

AE Language Model (BERT)

- BERT first constructs a corrupted version $\hat{x}$ by randomly setting a portion of tokens in x to a special symbol [MASK].
- Then BERT reconstruct masked tokens $\bar{x}$ from $\hat{x}$:

$$\max_{\theta} \quad \log p_\theta(\bar{\mathbf{x}} \mid \hat{\mathbf{x}}) \approx \sum_{t=1}^{T} m_t \log p_\theta(x_t \mid \hat{\mathbf{x}}) = \sum_{t=1}^{T} m_t \log \frac{\exp\left(H_\theta(\hat{\mathbf{x}})_t^\top e(x_t)\right)}{\sum_{x'} \exp\left(H_\theta(\hat{\mathbf{x}})_t^\top e(x')\right)}$$

  - Hidden vectors $H_\theta(\mathbf{x}) = [H_\theta(\mathbf{x})_1, H_\theta(\mathbf{x})_2, \cdots, H_\theta(\mathbf{x})_T]$
  - Masks m_t

Autoregressive v.s Autoencoding

- AR model (GPT)

$$\max_{\theta} \log p_\theta(x) = \sum_{t=1}^{T} \log p_\theta(x_t \mid x_{<t})$$

  - Pros
    - Estimate the joint probability of a text corpus
  - Cons
    - Lack of bidirectional context information

- AE model (BERT)

$$\max_{\theta} \log p_\theta(\bar{x} \mid \hat{x}) \approx \sum_{t=1}^{T} m_t \log p_\theta(x_t \mid \hat{x})$$

  - Pros
    - Utilize the bidirectional context information
  - Cons
    - Artificial symbols like [MASK]
    - Assume the predicted (masked) tokens are independent given unmasked ones
    - Lacks long-term dependency

Brain Storm (20 mins)

- Do you have any ideas to address the aforementioned issues in AR and AE models? (10 mins)

- Are any ideas/techniques from AR and AE that we can utilized in your research? (10 mins)

XLNet

- Permutation language modeling (Address cons in AR)

- Two-stream self-attention (Address cons in AE)

Permutation Language Modeling

origin sequence	[1, 2, 3, 4, 5, 6, 7, 8]
Forward AR	[1, 2, 3, 4, 5, 6, 7, 8]
Backward AR	[8, 7, 6, 5, 4, 3, 2, 1]
Permutation AR	[3, 2, 5, 6, 8, 1, 7, 4]

Permutation Language Modeling

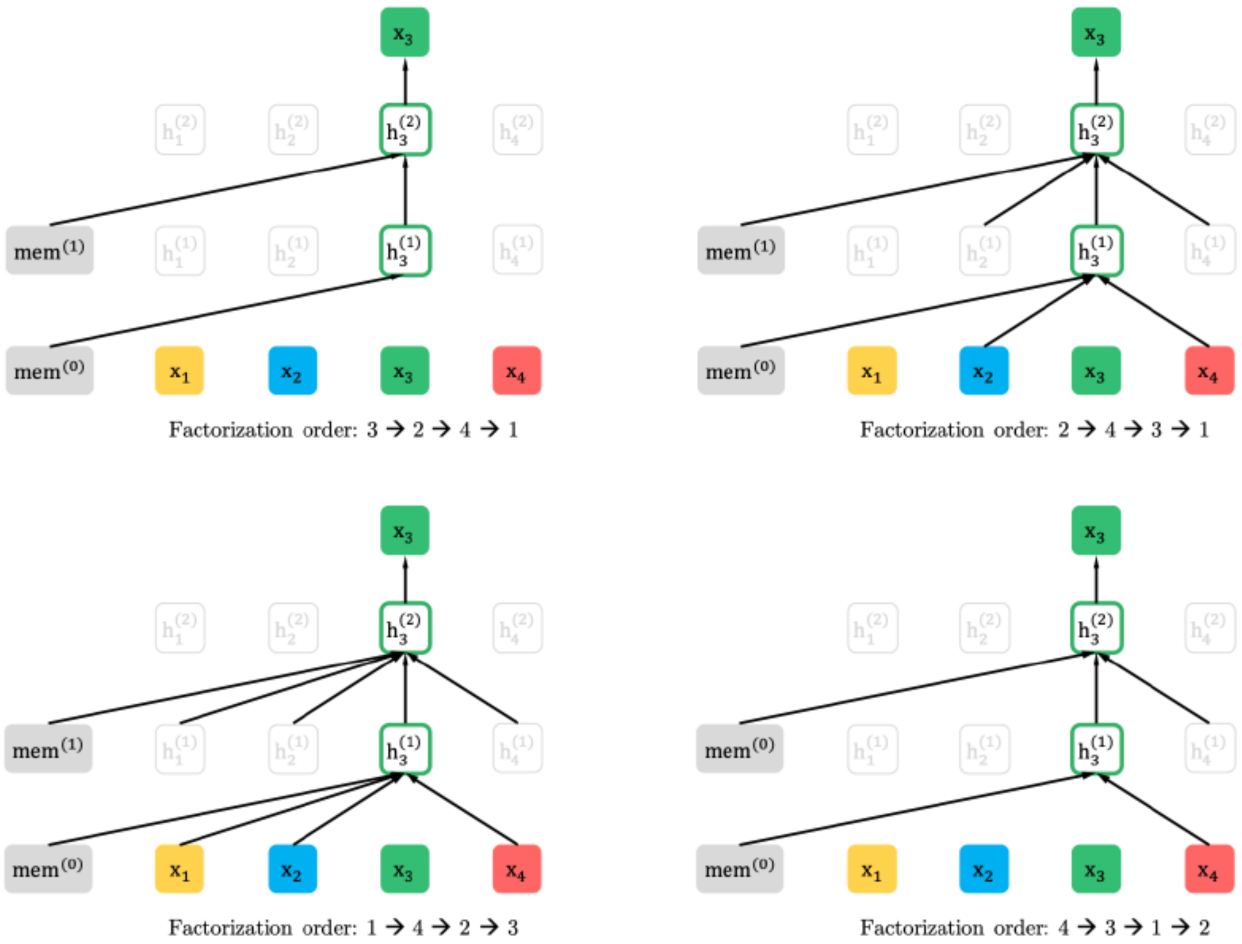
- Permutations of the length T corpus
  - Z_T: all possible permutations of the length-T index sequence.
  - z_t: t-th element.
  - $z_{<t}$: first $t-1$ elements of a permutation z.

- Objective:

$$\max_{\theta} \quad \mathbb{E}_{\mathbf{z} \sim \mathcal{Z}_T} \left[\sum_{t=1}^{T} \log p_\theta(x_{z_t} \mid \mathbf{x}_{\mathbf{z}_{<t}}) \right]$$

  - In expectation, x_t has seen possible element in the sequence.
  - Avoid the independence assumption.

Permutation Language Modeling



Permutation

[3, 2, 4, 1]

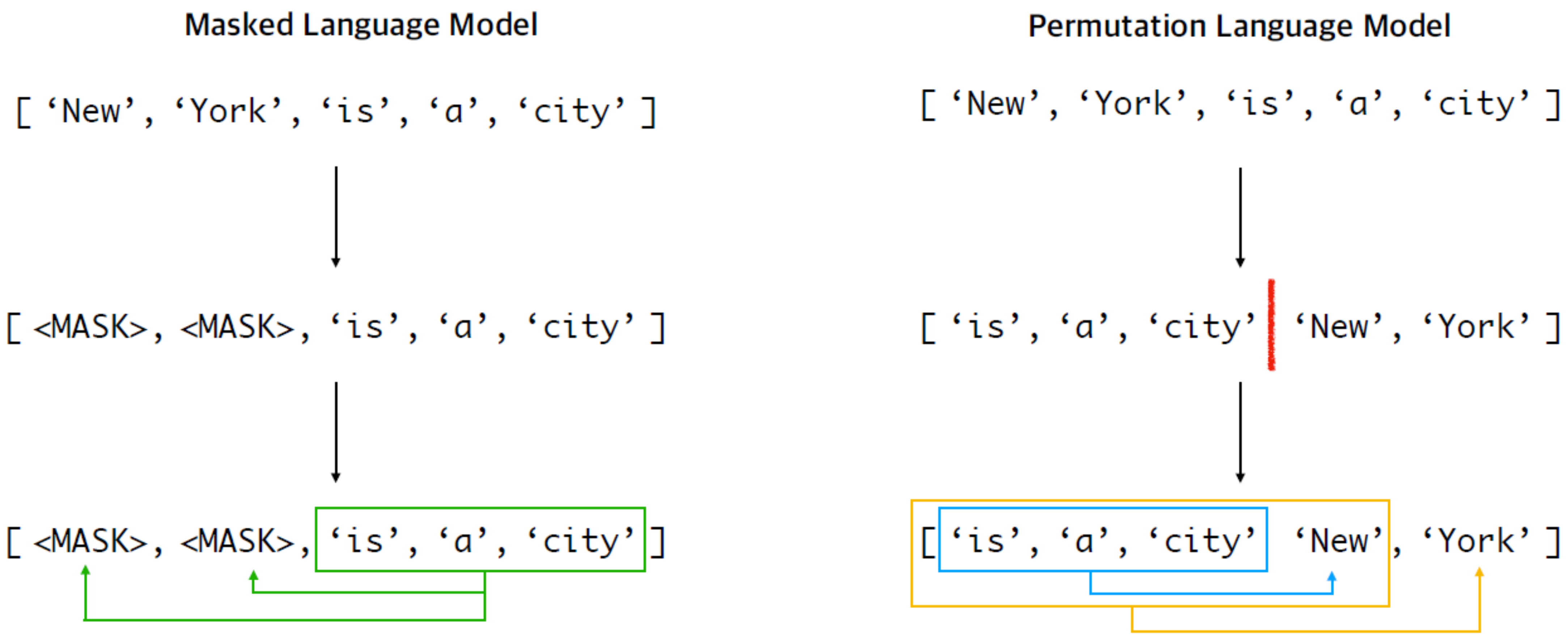
[2, 4, 3, 1]

[1, 4, 2, 3]

[4, 3, 1, 2]

$$\max_{\theta} \quad \mathbb{E}_{\mathbf{z} \sim \mathcal{Z}_T} \left[\sum_{t=1}^{T} \log p_\theta(x_{z_t} \mid \mathbf{x}_{\mathbf{z}_{<t}}) \right]$$

Permutation Language Modeling



Two-Stream Self-Attention

- Issue of naïve implementation with standard Transformer

Two-Stream Self-Attention

- Issue of naïve implementation with standard Transformer
  - The representation $h_\theta(x_{z<t})$ does not depend on which position it will predict.
  - The same distribution is predicted regardless of the target position.

- Re-parameterize the next-token distribution

$$p_\theta(X_{z_t} = x \mid \mathbf{x}_{z_{<t}}) = \frac{\exp\left(e(x)^\top g_\theta(\mathbf{x}_{\mathbf{z}_{<t}}, z_t)\right)}{\sum_{x'} \exp\left(e(x')^\top g_\theta(\mathbf{x}_{\mathbf{z}_{<t}}, z_t)\right)}$$

Two-Stream Self-Attention

- Intuitions
  - The content representation $g_{\theta(x_{z<t})}$ should only use the position z_t, and not the content x_z.

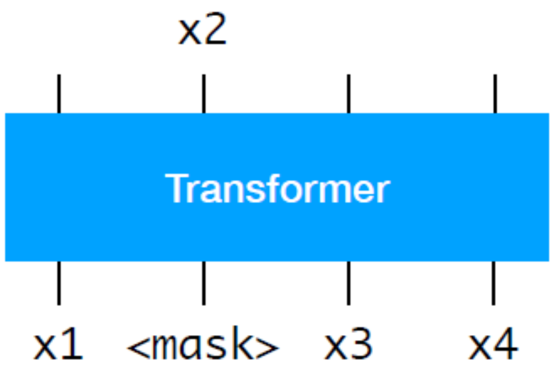
  - To predict the other tokens x_{z_j} (j > t) should also encode the content x_z to provide full contextual information.

Two-Stream Self-Attention

GPT

[x1, x2, x3, x4] → [x5]

BERT



Permutation LM

[x2, x1, x4, x3] → [x5]

→ [x6] ???

→ [x7]

Two-Stream Self Attention

not used when fine-tuning

Query Stream $g_{z_t}^{(m)} \leftarrow \text{Attention}(\mathrm{Q} = g_{z_t}^{(m-1)}, \mathrm{KV} = \mathbf{h}_{\mathbf{z}_{<t}}^{(m-1)}; \theta),$

Content Stream $h_{z_t}^{(m)} \leftarrow \text{Attention}(\mathrm{Q} = h_{z_t}^{(m-1)}, \mathrm{KV} = \mathbf{h}_{\mathbf{z}_{\le t}}^{(m-1)}; \theta),$

Similar to the role of standard hidden states in Transformer

Two-Stream Self-Attention

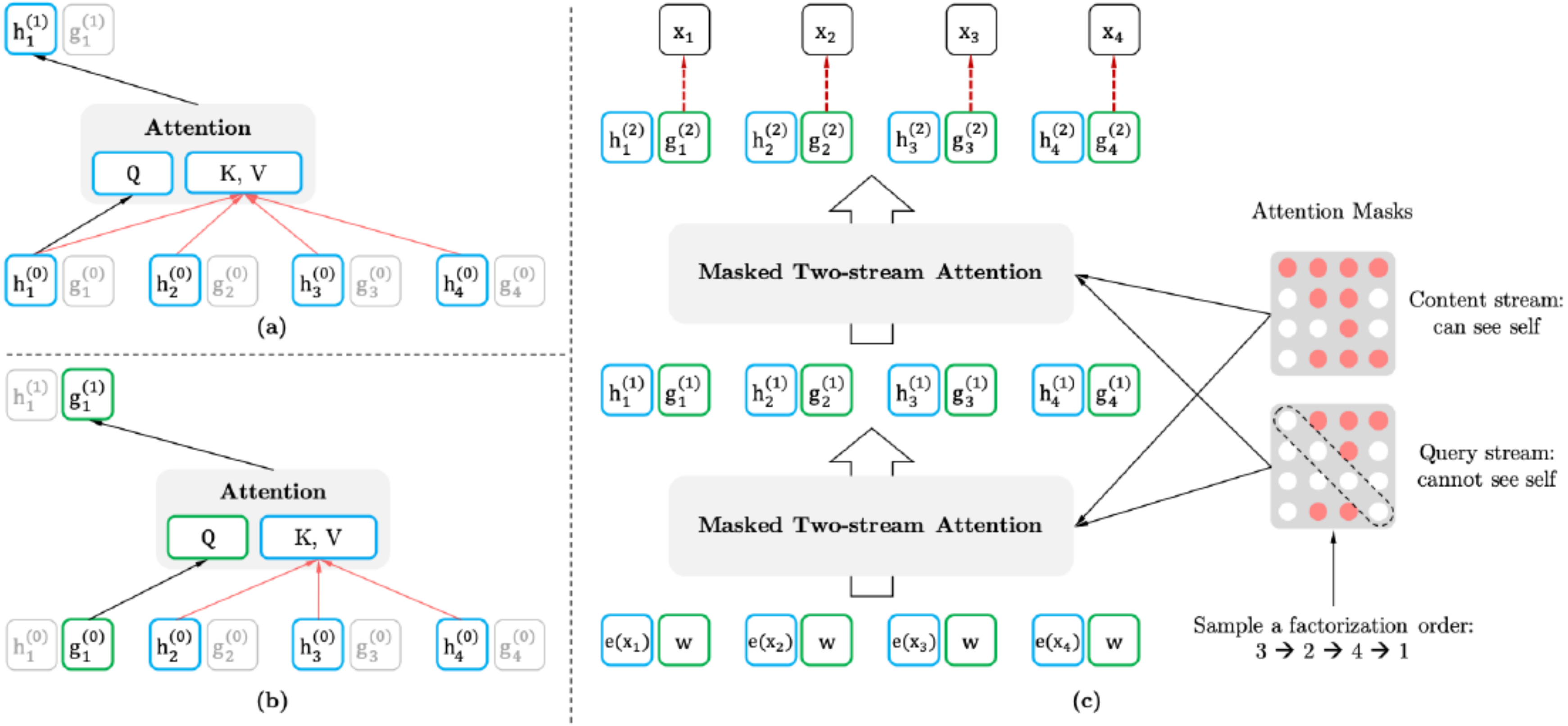


Figure 2: (a): Content stream attention, which is the same as the standard self-attention. (b): Query stream attention, which does not have access information about the content x_{z_t}. (c): Overview of the permutation language modeling training with two-stream attention.

Experiments

- Pretraining datasets
  - BERT: BookCorpus + English Wikipedia
  - XLNet: BookCorpus + English Wikipedia + Giga5 + ClueWeb + Common Crawl

- Model size
  - XLNet similar to BERT

- Training time
  - 512 TPU v3 (batch size 2048) → 2.5 days

Experiments

- Text classification

Model	IMDB	Yelp-2	Yelp-5	DBpedia	AG	Amazon-2	Amazon-5
CNN [14]	-	2.90	32.39	0.84	6.57	3.79	36.24
DPCNN [14]	-	2.64	30.58	0.88	6.87	3.32	34.81
Mixed VAT [30, 20]	4.32	-	-	0.70	4.95	-	-
ULMFiT [13]	4.6	2.16	29.98	0.80	5.01	-	-
BERT [35]	4.51	1.89	29.32	0.64	-	2.63	34.17
XLNet	**3.79**	**1.55**	**27.80**	**0.62**	**4.49**	**2.40**	**32.26**

Table 3: Comparison with state-of-the-art error rates on the test sets of several text classification datasets. All BERT and XLNet results are obtained with a 24-layer architecture with similar model sizes (aka BERT-Large).

Experiments

- Reading comprehension task

RACE	Accuracy	Middle	High
GPT [25]	59.0	62.9	57.4
BERT [22]	72.0	76.6	70.1
BERT+OCN* [28]	73.5	78.4	71.5
BERT+DCMN* [39]	74.1	79.5	71.8
XLNet	**81.75**	**85.45**	**80.21**

Table 1: Comparison with state-of-the-art results on the test set of RACE, a reading comprehension task. $*$ indicates using ensembles. "Middle" and "High" in RACE are two subsets representing middle and high school difficulty levels. All BERT and XLNet results are obtained with a 24-layer architecture with similar model sizes (aka BERT-Large). Our single model outperforms the best ensemble by 7.6 points in accuracy.


hello
literacy
Q&A